LOGO
UM PASSO CRIATIVO

A proposta deste livro é a de trabalhar a linguagem LOGO de forma abrangente, explorando todo o seu potencial, principalmente no que se refere à manipulação de palavras e listas.

Destina-se a alunos de nível médio e a você que busca uma linguagem simples para iniciar-se no mundo da informática.

Os programas são apresentados de forma gradativa, com o objetivo de fornecer-lhe elementos necessários a um contínuo aprofundamento neste interessante universo.

KÁTIA • REGINA

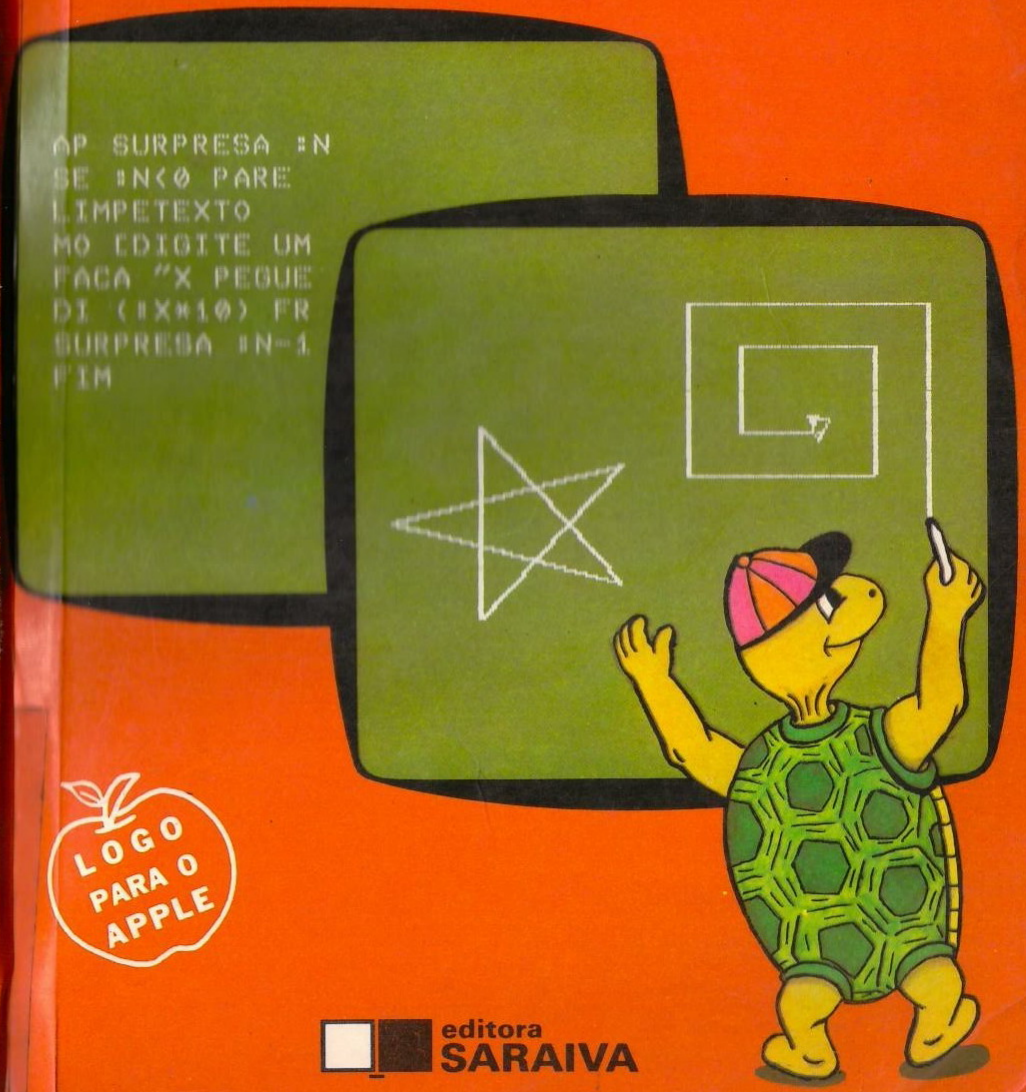
LOGO

UM PASSO CRIATIVO

SARAIVA

KÁTIA • REGINA

LOGO
UM PASSO CRIATIVO



LOGO PARA O APPLE

editora SARAIVA

KÁTIA SIMÕES DE QUEIROZ
REGINA CÉLIA ANDRADE E SILVA DE SOUZA

LOGO
UM PASSO CRIATIVO

1ª edição — 1988

LOGO PARA O APPLE

editora SARAIVA

Dados de Catalogação na Publicação (CIP) Internacional
(Câmara Brasileira do Livro, SP, Brasil)

Q44L

Queiroz, Kátia Simões de.
LOGO : um passo criativo / Kátia Simões de Queiroz, Regina Célia Andrade e Silva de Souza. — 1. ed. — São Paulo : Saraiva, 1988.

"LOGO para o Apple".

1. Computadores - Estudo e ensino 2. Computadores e crianças 3. LOGO (Linguagem de programação para computadores) I. Souza, Regina Célia Andrade e Silva de. II. Título.

88-0127

CDD-001.6424
-001.64024054
-001.6407

Índices para catálogo sistemático:

1. Computadores : Estudo e ensino : Processamento eletrônico de dados 001.6407
2. Computadores e crianças 001.64024054
3. Crianças e computadores 001.64024054
4. LOGO : Linguagem de programação : Computadores : Processamento de dados 001.6424

Supervisão editorial: José Lino Fruet
Copy-desk: Renato Alberto Colombo Jr.
Programação visual: Fátima Gilberti
Produção gráfica: João Batista Ribeiro F.º
Capa: Paulo Sérgio Alessi Manzi
Diagramação: Ana Mônica de Mello
Arte-final: Cesar Montagna de Oliveira, Francisco Augusto da Costa Filho, Marcelo Tadeu Gonçalves, Roseli Rita Andrade Jimenez, Vagner Castro dos Santos, Walter Reinoso.
Revisão: Ana Lúcia C. França, Ana C. Dionisio, Isabel Rebelo, Liege Marucci, Rosa M. Mangueira, Sandra Valenzuela, Teresa Cristina J. Costa
Composição: Susi Novais Merola
Revisão de fotolito: Liberato Verdile Junior

editora SARAIVA

Av. Marquês de São Vicente, 1697 — CEP 01139 — Barra Funda — Tel.: PABX (011) 826-8422
São Paulo-SP

Distribuidores Regionais

Belém: 222-9034
Belo Horizonte: 461-9962
Brasília: 226-3722
Campo Grande: 382-3682
Cuiabá: 322-5481
Curitiba: 234-2622
Fortaleza: 231-7881
Goiânia: 225-2882
Maceió: 221-9559
Manaus: 234-4664
Natal: 222-2569
Porto Alegre: 43-2986
Recife: 231-1764
Ribeirão Preto: 634-0546
Rio de Janeiro: 201-7149
Salvador: 244-0139
São Luís: 222-0107
Vitória: 227-5670

APRESENTAÇÃO

Este livro pretende dar um outro enfoque à linguagem LOGO: o manejo de listas e palavras, usando primitivas bastante específicas.

Embora seja dirigido a alunos de nível médio, pode ser útil a todos os jovens e adultos que buscam uma linguagem para iniciar-se no mundo da informática.

Queremos que ele seja uma ferramenta de estímulo para que você continue se aprofundando nesta linguagem, visto que alguns programas aqui apresentados possuem uma elaboração suficientemente elevada para que sirvam de exemplo.

Esperamos que, no decorrer dos capítulos, muitos desafios surjam, para que você possa desenvolver sua criatividade, um dos elementos fundamentais da filosofia LOGO.

A versão utilizada é a MLOGO para computadores compatíveis com a linha APPLE. Assim, ao se trabalhar com outro tipo de computador, deve-se fazer uma tradução dos comandos para a versão que você for utilizar.

No Apêndice deste livro, você encontrará uma tabela de conversão dos comandos utilizados, do MLOGO para o LOGO ITAUTEC.

as autoras

KÁTIA SIMÕES DE QUEIROZ
- Bacharelado em Matemática — Universidade Mackenzie.
- Coordenadora do Departamento Educacional da Domus Informática.

REGINA CÉLIA ANDRADE E SILVA DE SOUZA
- Pedagogia — FFCL de Moema.
- Coordenadora da Área de Informática da Escola Experimental Pueri Domus.

Gostaríamos de agradecer:

À **Domus Informática**, pela disponibilidade de material e equipamentos para a elaboração deste livro.

ENDEREÇO PARA CURSOS E INFORMAÇÕES:
DOMUS INFORMÁTICA LTDA.
R. Verbo Divino, 993-A
Fone: 521-2155, ramais 123 ou 155

ÍNDICE

CAPÍTULO 1

INTRODUÇÃO

Você que já leu o livro ***LOGO: O Primeiro Passo*** deve estar lembrado de que a linguagem LOGO possui um conjunto de comandos e funções básicas chamadas **primitivas**, utilizadas para a construção de **procedimentos**.

Cada procedimento criado passa a ser incorporado ao vocabulário já existente, podendo ser usado da mesma forma que as primitivas.

Vale recordar, ainda, que a linguagem LOGO é organizada em módulos, ou seja, um projeto LOGO não precisa ser descrito como um único programa, e pode ser dividido em diversos procedimentos definidos separadamente. Esta é uma forte característica da linguagem LOGO.

Se no decorrer de seu trabalho, você tiver necessidade de gravar seus projetos, proceda da mesma maneira vista no livro ***LOGO: O Primeiro Passo***. Caso não se recorde, no Apêndice deste livro, você encontrará um resumo de comandos intitulado **Comandos para Operação de Memória e Disco**.

E agora, mãos a obra!

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
ESCREVA		Permite que ingressemos no modo de texto posicionando o cursor no canto superior esquerdo da tela.
PALAVRA? n		Responde **VERD** se n for uma palavra e **FALSO** caso contrário.
NUMERO? n		Responde **VERD** se n for um número e **FALSO** caso contrário.
LISTA? n		Responde **VERD** se n for uma lista e **FALSO** caso contrário.

APRESENTANDO PALAVRAS E LISTAS

Vejamos uma lista de compras:

ARROZ, SAL, PIMENTA, OVOS e BATATA

Em LOGO, também podemos utilizar informações do tipo **LISTA** (como na lista de compras). Chamamos de LISTA, o conjunto de palavras, separadas umas das outras por um espaço em branco. Esta LISTA pode conter um único elemento (LISTA UNITÁRIA) ou mesmo nenhum (LISTA VAZIA). É bom, ainda, percebermos que a ordem em que os elementos estão dispostos na lista é muito importante. Exemplo:

Na lista de compras temos:

1º elemento ⟶ ARROZ
2º elemento ⟶ SAL
3º elemento ⟶ PIMENTA
4º elemento ⟶ OVOS
5º e último elemento ⟶ BATATA

Para que uma lista seja definida, devemos colocar seus elementos entre colchetes ([ ]).

EXPERIMENTE!!!

Você que já trabalhou com a parte gráfica do LOGO deve recordar que para desenharmos utilizávamos o comando **DESENHE** . Agora, como trabalharemos com textos, devemos ingressar no **modo de texto**. Para isso digitaremos **ESCREVA** .

O comando **ESCREVA** faz com que o computador apresente uma tela em branco, com o cursor posicionado no canto superior esquerdo.

Note que a tartaruga não aparece, pois ela só é utilizada para desenharmos.

Então vamos lá.

Digite:

```
                [ARROZ SAL PIMENTA OVOS BATATA]
Obteremos ⟶ IGUAL:[ARROZ SAL PIMENTA OVOS
              BATATA]
```

Observação: A mensagem **IGUAL:** significa que nós não especificamos o que deveria ser feito com a lista utilizada.

```
                [PATO RATO GATO]
Obteremos ⟶ IGUAL:[PATO RATO GATO]

                [MARMELADA] —— lista unitária
Obteremos ⟶ IGUAL:[MARMELADA]

                [] ——— lista vazia
Obteremos ⟶ IGUAL:[]
```

OBSERVE!

É também considerado lista, o conjunto de ordens atribuído ao comando **REPITA:** . Exemplo:

```
REPITA 36 [FR 10 ES 10]
          └──────────┘
              lista
```

Por sua vez, cada PALAVRA (elemento de LISTA) é um conjunto de caracteres que não inclui o espaço em branco. PALAVRA pode possuir um único caractere (PALAVRA UNITÁRIA) ou nenhum (PALAVRA VAZIA). Neste caso, tam-

bém é importante a ordem em que os elementos estão dispostos. Exemplo:

Na primeira palavra da lista de compras:

1º caractere → A
2º caractere → R
3º caractere → R
4º caractere → O
5º e último caractere → Z

Para que uma PALAVRA seja definida, devemos indicar seus elementos precedidos de aspas (").

LEMBRE-SE!

Caractere é toda letra, número ou sinal gráfico existente no teclado do nosso micro.

EXPERIMENTE!!!

Digite:

```
"CASA
```
Obteremos →
```
IGUAL:CASA
```

```
"C.;E2
```
Obteremos →
```
IGUAL:C.;E2
```

```
"A
```
→ palavra unitária

Obteremos →
```
IGUAL:A
```

→ palavra vazia

Obteremos →
```
IGUAL:
```


OBSERVE!

Ao gravarmos um arquivo no disquete, o nome deste arquivo é uma PALAVRA. Veja:

```
GRAVE "POLIGONOS
```

Nomes de procedimentos, primitivas do LOGO e nomes de variáveis também são PALAVRAS.

NÚMERO É PALAVRA?

EXPERIMENTE!!!

Digite:

```
NUMERO? "123
```
Obteremos →
```
IGUAL:VERD
```

```
NUMERO? 123
```
Obteremos →
```
IGUAL:VERD
```

```
PALAVRA? "123
```
Obteremos →
```
IGUAL:VERD
```

```
PALAVRA? 123
```
Obteremos →
```
IGUAL:VERD
```

```
NUMERO? [123]
```
Obteremos →
```
IGUAL:FALSO
```

```
LISTA? [123]
```
Obteremos →
```
IGUAL:VERD
```

As primitivas **NUMERO?** , **PALAVRA?** e **LISTA?** nos informam qual a natureza do elemento determinado.

NUMERO? nnn nos responde **VERD** se **nnn** for um número e **FALSO** , caso contrário.

PALAVRA? nnn nos responde **VERD** se **nnn** for uma palavra e **FALSO** , caso contrário.

LISTA? nnn nos responde **VERD** se **nnn** for uma lista e **FALSO** , caso contrário.

Baseados nestas observações, podemos concluir que NÚMERO é considerado PALAVRA, pois todo número é uma seqüência de caracteres que não contém espaços em branco. Veja:

1893 ⟶ 4 caracteres

- 1º caractere ⟶ 1
- 2º caractere ⟶ 8
- 3º caractere ⟶ 9
- 4º caractere ⟶ 3

e, no caso do número, podemos ou não apresentá-lo entre aspas (").

Assim, todo número é palavra, mas nem toda palavra é número.

Daí, podemos partir para duas definições a respeito das seqüências de caracteres que formam as palavras:

a. **Seqüência alfa-numérica** é toda seqüência composta por qualquer caractere, incluindo-se os números; porém não exclusivamente.

b. **Seqüência numérica** é toda seqüência de caracteres composta somente por números.

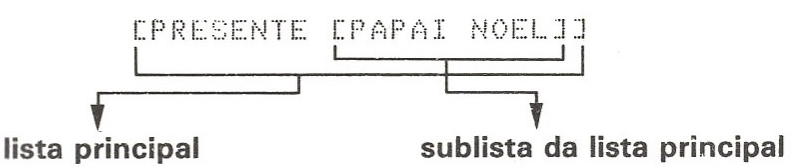
SUGESTÕES

Verifique qual a natureza dos elementos abaixo, respondendo com **VERD** ou **FALSO** às primitivas **PALAVRAS?** , **NUMERO?** e **LISTA?**

1. `PALAVRA? [ABOBORA]`
2. `LISTA? [ABOBORA]`
3. `PALAVRA? ABOBORA`
4. `PALAVRA? "ABOBORA`
5. `NUMERO? "5+3`
6. `NUMERO? 2A`
7. `NUMERO? "7 + "8`
8. `NUMERO? 1+2`
9. `NUMERO? [1+2]`
10. `LISTA? "ARROZ "FEIJAO`
11. `LISTA? [PRESENTE [PAPAI NOEL]]`
12. `LISTA? [CASA 158 TEL 571-8816]`
13. `LISTA? "5+3`

OBSERVE!

Como no exemplo 11, uma lista pode ser elemento de outra lista. Nós a chamaremos de sublista da lista principal.



A lista principal possui dois elementos:

1º **A PALAVRA "PRESENTE**
2º **A LISTA [PAPAI NOEL]**

A sublista da lista principal também possui dois elementos:

1º **A PALAVRA "PAPAI**
2º **A PALAVRA "NOEL**

CAPÍTULO 2

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
MOSTRE nnn	**MO**	Mostra no vídeo o elemento **nnn**, deslocando o cursor para a primeira posição da linha seguinte.
MOSTRAR nnn		Idem ao **MOSTRE** , porém mantém o cursor na mesma linha.
+		Operação de adição
–		Operação de subtração
*		Operação de multiplicação
/		Operação de divisão
SORTEIE n		Responde com um valor qualquer entre **0** e **n–1**.
INT n		Responde com a parte inteira do número n desprezando a parte decimal.
QUOC a b		Responde com o quociente da divisão de **a** por **b**.
RESTO a b		Responde com o resto da divisão de **a** por **b**.
RQD n		Responde com o valor correspondente à raiz quadrada de **n**.

ESCREVENDO

Copie a seguinte frase: "O DIA ESTÁ BOM".

O computador também pode "copiar" no vídeo o que desejarmos.

Para isso, vamos conhecer um novo comando: **MOSTRE (MO)** . Sua função é mostrar (copiar) o elemento especificado como entrada, que pode ser palavra ou lista.

EXPERIMENTE!!!

```
MO [BOM TRABALHO]
BOM TRABALHO
```
Obteremos ⟶

O comando **MOSTRE** imprime na tela a mensagem contida entre colchetes ([ ]).

Se quisermos mostrar uma palavra, devemos utilizar aspas (") em lugar de colchetes. Assim:

```
MO "TRABALHO
TRABALHO
```
Obteremos ⟶

Porém, se digitarmos: **MO [TRABALHO]** , receberemos a mesma resposta. No entanto, como já foi visto no capítulo anterior, a natureza de ambas é diferente: a primeira é a palavra **TRABALHO** e a segunda, a lista unitária **[TRABALHO]**

EXPERIMENTE!!!

```
MO 2+3
5
```
Obteremos ⟶

```
MO "2+3
2+3
```
Obteremos ⟶


```
MO "2 + 3
5
```
Obteremos ⟶

```
MO 2 + "3
5
```
Obteremos ⟶

Podemos, ainda, digitar **MO [ ]** e será mostrada na tela uma lista vazia. Isto pode ser útil quando quisermos pular uma linha entre as partes de um procedimento.

Há outro comando muito parecido com o **MOSTRE** . É o **MOSTRAR** , que tem a mesma função, porém com uma pequena diferença.

OBSERVE!

Digite:

```
MOSTRE "CASA
CASA
?■
```
Obteremos ⟶

o cursor é posicionado no início da linha seguinte.

Agora, digite:

```
MOSTRAR "CASA
CASA?■
```
Obteremos ⟶

o cursor é posicionado imediatamente após o último caractere.

Você deve ter percebido que os comandos **MOSTRE** e **MOSTRAR** admitem um único argumento como entrada.

Digite:

```
MO "AV. MO [SANTO AMARO,] MO 1240
AV.
SANTO AMARO,
1240
?
```

Obteremos →

```
MOSTRAR "BI MOSTRAR "CI MOSTRAR
"CLE MOSTRAR "TA
BICICLETA?
```

Obteremos →

Para que a estes comandos possamos atribuir mais de uma entrada, devemos colocar a ordem, juntamente com seus argumentos, entre parênteses. Exemplo:

```
( MO "AV. [SANTO AMARO,] "1240 )
AV. SANTO AMARO, 1240
?
```

Obteremos →

Observação: É necessário o espaço em branco entre os parênteses e os dados no seu interior:

```
( MOSTRAR "BI "CI "CLE "TA )
BICICLETA?
```

Obteremos →

Vamos criar um procedimento utilizando o **MOSTRE** , atribuindo a ele várias entradas:

```
AP FRASE :SUBST :VERBO :LOCAL
( MO "O :SUBST :VERBO "NO :LOCAL )
FIM
```

Digite:

```
FRASE "GATO "DORME "TELHADO
O GATO DORME NO TELHADO
```

Obteremos →

Observação: Devemos nos lembrar de que os valores dados às variáveis são palavras, logo, deverão vir precedidos de aspas (").

COMANDO OU OPERAÇÃO?

Devemos verificar a diferença existente entre COMANDOS e OPERAÇÕES que podem ser primitivos ou por nós criados (até agora só criamos comandos).

COMANDO indica sempre uma ação. Pode ou não necessitar de dados de entrada. Exemplo:

I. **DESENHE:** indica a ação de limpar a tela gráfica, posicionando a tartaruga no centro. Não necessita de dados.

II. **FRENTE 50:** é a ação de deslocar a tartaruga para a frente. É necessário, como dado, a quantidade de passos que a tartaruga deve se deslocar, no caso 50.

É conveniente notar que não podemos determinar duas ações para o mesmo dado. Por exemplo: **FR VO 50** .

A OPERAÇÃO necessita de dados de entrada e nos dá como resposta o resultado dos dados operados. Exemplo:

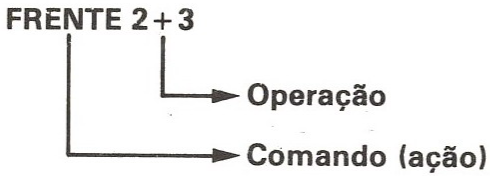
2+3 → + é a operação

2 e **3** são os dados de entrada, que deverão "sofrer" a operação.

5 é o resultado dos dados já operados que obteremos como resposta.

Note que se digitarmos no computador **2+3** , obteremos **IGUAL:5** , ou seja, a operação foi efetuada, porém, fal-

ta uma ação que determine o que deve ser feito com o resultado (**5**). Exemplo:



CALCULANDO

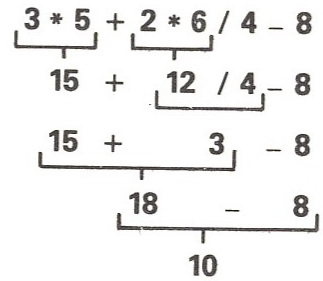
O LOGO também pode ser usado como calculadora. Além de somar, subtrair, multiplicar e dividir, extrai raiz quadrada, sorteia números etc.

Estas operações podem ser usadas com ou sem tartaruga.

PROBLEMA:	**INSTRUÇÃO:**
1. Mostrar na tela o resultado da soma entre 5 e 8.	`MO 5+8`
2. Girar a tartaruga à direita na quantidade de graus resultante da diferença entre 135 e 45.	`DI 135-45`
3. Mostrar na tela o resultado da multiplicação de 4 por 9.	`MO 4*9`
4. Deslocar a tartaruga para a frente na quantidade de passos resultante da divisão de 36 por 6.	`FR 36/6`

Expressões aritméticas são resolvidas através do LOGO, obedecendo a mesma hierarquia de sinais utilizada na Matemática.

Tente resolver:



EXPERIMENTE!!!

Agora, proponha ao LOGO a mesma expressão e confira o resultado: `MO 3*5+2*6/4-8`

Você perceberá que serão resolvidas, primeiramente, as multiplicações e divisões e, a seguir, as somas e subtrações.

Os parênteses podem ser usados quando quisermos alterar essa hierarquia. Assim:

```
MO 3 * (5+2*6/4) - 8
```
Obteremos ⟶ 16

Suponhamos que você queira dividir 20 pela soma de 2 e 3. A resposta será 4. Digite:

```
MO 20/2+3
```
Obteremos ⟶ 13

Isto porque a hierarquia foi respeitada, ou seja, primeiro foi feita a divisão de 20 por 2 e, a seguir, adicionado 3.

Para conseguir o resultado esperado, é necessário a utilização dos parênteses.

```
MO 20/(2+3)
```
Obteremos ⟶ 4

Quando trabalhamos com incrementos na recursão, também utilizamos operações, como no exemplo abaixo:

```
AP LABIRINTO :L
FR :L
DI 90
LABIRINTO :L+5
FIM
```

Dispomos, ainda, de outras operações, além das já vistas. São elas:

I. **SORTEIE:** permite a obtenção de números aleatórios, ou seja, números sorteados ao acaso.

Ao ordenarmos **SORTEIE 100** , obtemos como resposta um número qualquer entre **0** e **99** .

EXPERIMENTE!

Digite:

MO SORTEIE 100 , e veja qual a resposta obtida. Digitando novamente a mesma ordem, a resposta certamente será outra.

SORTEIE só se utiliza de números positivos.

OBSERVE!

Se você quiser um número sorteado entre 1 e 5, deverá digitar:

```
1 + SORTEIE 5
```

Isto significa: Sorteie um número entre 0 e 4, e some 1 a ele.

Digite o procedimento abaixo:

```
AP PONTOS
SC TODATELA
ES SORTEIE 360
VO SORTEIE 30
CL FR 1
PONTOS
FIM
```

Digite **PONTOS** e observe o resultado.

SUGESTÃO

Explore mais a operação **SORTEIE** e veja que resultados interessantes você vai obter.

Por exemplo, escreva um procedimento para desenhar figuras geométricas de tamanhos variados, em diferentes posições na tela.

II. **INT:** converte um valor decimal em um valor inteiro, ignorando a parte decimal. Exemplo:

```
MO INT 108.9632
```

Obteremos ⟶ 108

III. **QUOC:** retorna a parte inteira do quociente da divisão do primeiro valor de entrada pelo segundo. Exemplo:

```
MO QUOC 8 5
```

Obteremos ⟶ 1

IV. **RESTO:** retorna o resto da divisão do primeiro valor de entrada pelo segundo. Exemplo:

```
MO RESTO 8 5
```
Obteremos ⟶ 3

V. **RQD:** retorna a raiz quadrada do valor numérico positivo utilizado como entrada. Exemplo:

```
MO RQD 36
```
Obteremos ⟶ 6

Vamos criar alguns procedimentos que utilizem estas operações:

1.
```
AP NUMEROPAR :N
SE RESTO :N 2 = 0 ENTAO ( MO
[O NUMERO] :N [E' PAR] ) PARE
( MO [O NUMERO] :N [E' IMPAR] )
FIM
```

Digitando:

```
NUMEROPAR 50
```
Obteremos ⟶ O NUMERO 50 E' PAR

2.
```
AP RESTANTE :DIVIDENDO :DIVISOR
MO :DIVIDENDO - (:DIVISOR * QUOC
:DIVIDENDO :DIVISOR)
FIM
```

Digitando:

```
RESTANTE 12 5
```
Obteremos ⟶ 2

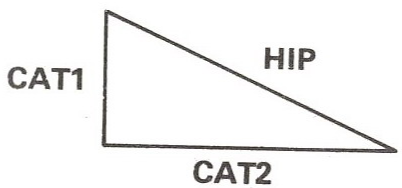
3.
```
AP DECIMAL :X :Y
MO :X/:Y - INT (:X/:Y)
FIM
```

Este procedimento mostra no vídeo a parte decimal do quociente da divisão de 8 por 5.

Digitando:

```
DECIMAL 8 5
```
Obteremos ⟶ .6

4. Você se lembra do teorema de Pitágoras que diz: "O quadrado da hipotenusa é igual à soma dos quadrados dos catetos"?



$$HIP = \sqrt{CAT1^2 + CAT2^2}$$

Dados os dois catetos, vamos fazer um procedimento para calcular a hipotenusa.

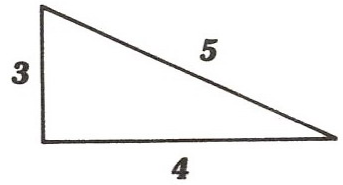
EXPERIMENTE!!!

```
AP HIPOTENUSA :CAT1 :CAT2
MO RQD (:CAT1 * :CAT1 + :CAT2 * :CAT2)
FIM
```

Digitando:

```
HIPOTENUSA 3 4
```
Obteremos ⟶ 5

Isto é, as medidas do triângulo são:



Observe a instrução:

```
MO RQD (:CAT1 * :CAT1 + :CAT2 * :CAT2)
```

ação — **quadrado do cat1** — **quadrado do cat2**

operação raiz quadrada

5. Como calcular o MDC, pelo método das divisões sucessivas? Vejamos o MDC entre 28 e 12.

	2	3
28	12	4
4	0	

→ MDC entre 28 e 12.

Podemos notar que:

1º passo

	QUOCIENTE	
DIVIDENDO	DIVISOR	
RESTO		

2º passo

O divisor passa a ser o novo dividendo e o resto o novo divisor. Chamaremos a estes **DIVIDENDO 2** e **DIVISOR 2** . Então, teremos novamente:

	QUOCIENTE 2	
DIVIDENDO 2	DIVISOR 2	
RESTO 2		

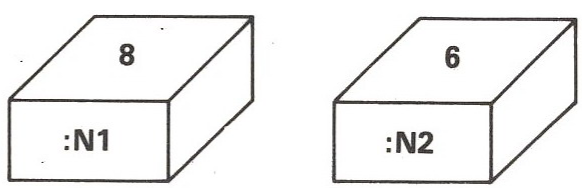
e assim, sucessivamente, até que o resto seja igual a zero. O **MDC** será o valor correspondente ao último divisor.

Portanto, vamos ensinar o procedimento abaixo e, a seguir, acompanhar sua resolução:

```
AP MDC :N1 :N2
SE RESTO :N1 :N2 = 0 MO :N2 PARE
MDC :N2 RESTO :N1 :N2
FIM
```

Digite: `MDC 8 6`

1. **VISUALIZANDO AS VARIÁVEIS E SEUS CONTEÚDOS**



2. **LINHA 1 — NÍVEL 1 ⟶ SE RESTO :N1 :N2 = 0 MO :N2 PARE**

Calcule o resto da divisão de **:N1** por **:N2**

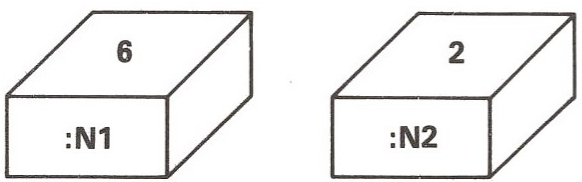
```
8 | 6
2   1
```

RESTO :N1 :N2 é igual a **2**, portanto diferente de **0**.

Sendo assim, passaremos à linha seguinte.

3. **LINHA 2 — NÍVEL 1 ⟶ MDC :N2 RESTO :N1 :N2**

Reinicie o procedimento **MDC** atribuindo:
À variável **:N1** o valor contido em **:N2** .
À variável **:N2** o resto da divisão de **:N1** por **:N2** .
Então, os conteúdos das variáveis **:N1** e **:N2** ficaram:

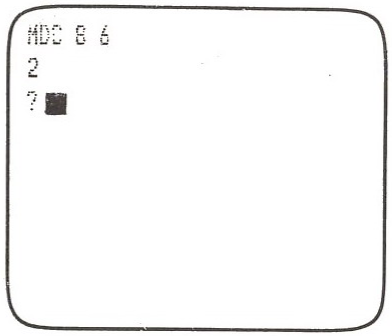


4. **LINHA 1 — NÍVEL 2 ⟶ SE RESTO :N1 :N2 = 0 MO :N2 PARE**

Calcule o resto da divisão de **:N1** por **:N2** .

```
6 | 2
0   3
```

RESTO :N1 :N2 é igual a **0**. Assim, mostre no vídeo o conteúdo de **:N2** (que é **2**) e pare.



SUGESTÕES

Faça procedimentos para:

a. Calcular se um número é divisível por 3, 5, 7, 11 etc.

b. Calcular a área de figuras geométricas, tais como: quadrado, retângulo, triângulo, trapézio.

CONTANDO

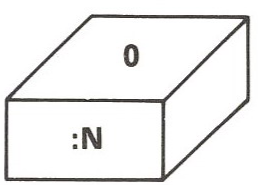
Vamos fazer um contador?

Experimente:

```
AP CONTAR :N
MO :N
CONTAR :N+1
FIM
```

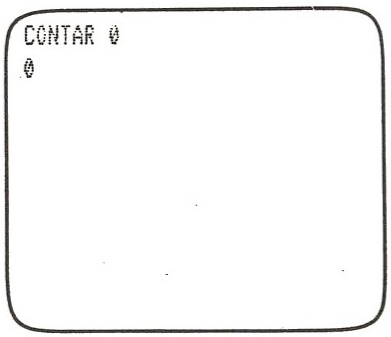
Digite **CONTAR 0** e analise o procedimento:

1. **VISUALIZANDO A VARIÁVEL E SEU CONTEÚDO**



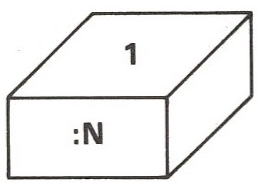
2. **LINHA 1 — NÍVEL 1 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** (que é **0**).



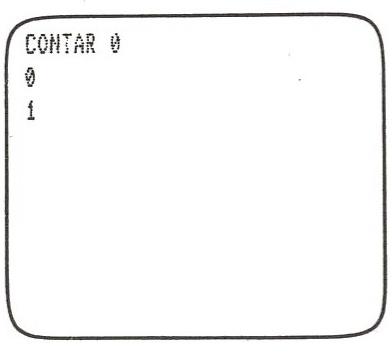
3. **LINHA 2 — NÍVEL 1 ⟶ CONTAR :N+1**

Reinicie o procedimento **CONTAR** atribuindo:
À variável **:N** o seu próprio valor **+ 1**.
Teremos:



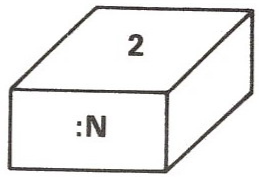
4. **LINHA 1 — NÍVEL 2 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** (que é **1**).



5. **LINHA 2 — NÍVEL 2 ⟶ CONTAR :N+1**

Reinicie o procedimento **CONTAR** , atribuindo:
À variável **:N** seu próprio valor **+1**.
Teremos:



E assim sucessivamente.

Observe que este procedimento conta de um em um, partindo de um valor qualquer, até ser interrompido através das teclas **CTRL** e **G**. Assim, se chamarmos **CONTAR 10** , ele irá contar de um em um a partir do numeral **10**, até que ocorra a interrupção.

Se quisermos contar até um determinado valor, basta estabelecermos uma condição:

```
AP CONTAR :N
SE :N > 10 PARE
MO :N
CONTAR :N+1
FIM
```

Digitando:

```
CONTAR 5
```

Obteremos ⟶

```
5
6
7
8
9
10
```

Neste procedimento, nosso contador vai de um número, por nós determinado, até **10** e pára.

Pode ser interessante que durante a execução de procedimentos haja uma pausa entre uma mensagem e outra, ou entre um desenho e outro. Para tanto, podemos utilizar um procedimento semelhante ao anterior, porém, sem que os valores sejam mostrados no vídeo. Seria como se o computador estivesse contando em voz baixa.

Nosso procedimento ficaria:

```
AP ESPERE :T
SE :T < 0 PARE
ESPERE :T-1
FIM
```

Este procedimento entrará em recursão, partindo do número determinado, até atingir o **0**, porém, os valores não serão mostrados no vídeo, pois omitimos de nosso programa a linha **MO :T** .

Mais adiante, veremos a utilidade prática do procedimento acima.

Podemos, também, construir um contador onde tanto o valor inicial como o valor final sejam determinados por nós, da seguinte forma:

```
AP CONTAR :VI :VF
SE :VI > :VF PARE
MO :VI
CONTAR :VI+1 :VF
FIM
```


EXPERIMENTE!!!

Altere o procedimento anterior, para que o contador conte de dois em dois, três em três, cinco em cinco etc.

Veja dois procedimentos que utilizam os contadores:

1.
```
AP TABUADA :N :C
SE :C >10 PARE
( MO :N "X :C "= :N*:C )
TABUADA :N :C+1
FIM
```

Digite:

```
TABUADA 5 1
```

Obteremos →

```
5 X 1 = 5
5 X 2 = 10
5 X 3 = 15
5 X 4 = 20
5 X 5 = 25
5 X 6 = 30
5 X 7 = 35
5 X 8 = 40
5 X 9 = 45
5 X 10 = 50
```

Para este, usaremos o procedimento **ESPERE :T** , já definido, além de um novo comando: **LIMPETEXTO** , cuja função é limpar a tela de texto, posicionando o cursor na margem esquerda da primeira linha disponível. Lembre-se de que quando trabalhamos na tela gráfica, temos apenas quatro linhas de texto disponíveis.

2.
```
AP ADIVINHE
DESENHE
MO [ ADIVINHE O QUE EU VOU DESENHAR.]
ESPERE 100
LAB 5
LIMPETEXTO
MO [ JA' DESCOBRIU?]
MO [ VOU DESENHAR MAIS UMA PARTE.]
ESPERE 100
FR 30 QUA 20
LIMPETEXTO
MO [ AGORA ESTA' FACIL.]
ESPERE 100
CARA
LIMPETEXTO
MO [ GOSTOU DO MEU CARACOL?]
ESPERE 100
LIMPETEXTO
MO [ SE VOCE QUISER, COMPLETE
O DESENHO.]
FIM
```

Analisando o procedimento acima, podemos notar que alguns subprocedimentos deverão ser ensinados. São eles: **LAB** , **QUA** e **CARA** .

1.
```
AP LAB :L
SE :L > 60 PARE
FR :L
DI 90
LAB :L+5
FIM
```

2.
```
AP QUA :L
REPITA 4 [ FR :L ES 90 ]
FIM
```


3.
```
AP CARA
ES 90 FR 20
DI 90 FR 5 DI 90 FR 8
SC ES 90 FR 5 DI 90
CL QUA 6
SC ES 90 FR 10 ES 45
CL FR 10 VO 10 DI 90
FR 10 ST
FIM
```

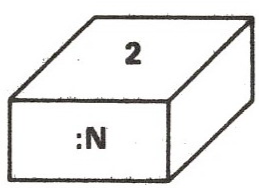
Tomamos o cuidado de apagar as linhas de texto, antes que outras fossem mostradas, bem como com o tempo dado a fim de que elas fossem lidas. Se você achar o tempo muito curto poderá prolongá-lo. Para tanto, basta aumentar o valor atribuído à variável **:T** , no subprocedimento **ESPERE** , da seguinte forma: substitua a linha **ESPERE 100** por **ESPERE 200** ou qualquer outro valor.

Vamos criar um contador que conte em ordem decrescente e crescente ao mesmo tempo:

```
AP CONT :N
SE :N < 0 PARE
MO :N
CONT :N-1
MO :N
FIM
```

Digite **CONT 2** e acompanhe a resolução.

1. VISUALIZANDO A VARIÁVEL E SEU CONTEÚDO

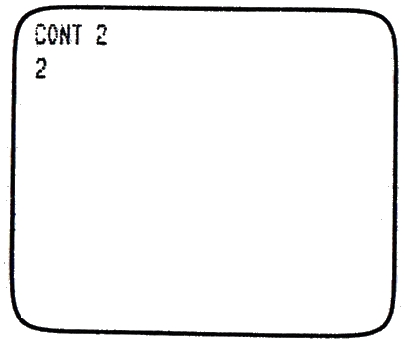


2. **LINHA 1 — NÍVEL 1 ⟶ SE :N 0 PARE**

2 não é menor do que **0**. Portanto, passe à linha seguinte.

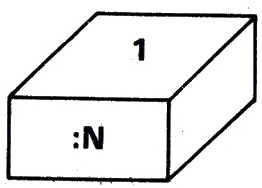
3. **LINHA 2 — NÍVEL 1 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** (que é **2**).



4. **LINHA 3 — NÍVEL 1 ⟶ CONT :N-1**

Reinicie o procedimento **CONT** , atribuindo:
À variável **:N** seu próprio valor **— 1**
Teremos:

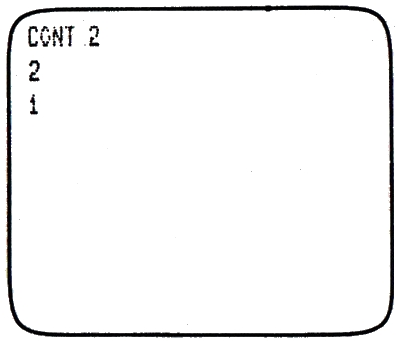


5. **LINHA 1 — NÍVEL 2 ⟶ SE :N 0 PARE**

1 não é menor do que **0**. Portanto, passar à linha seguinte.

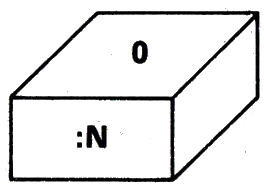
6. **LINHA 2 — NÍVEL 2 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** (que é **1**).



7. **LINHA 3 — NÍVEL 2 ⟶ CONT :N-1**

Reinicie o procedimento **CONT** , atribuindo:
À variável **:N** seu próprio valor **-1**
Teremos:

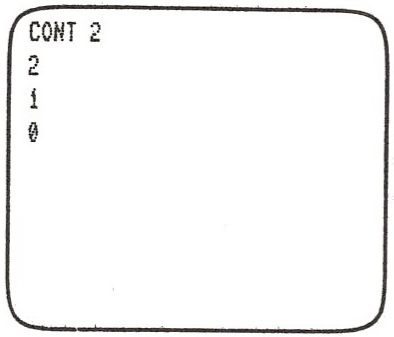


8. **LINHA 1 — NÍVEL 3 ⟶ SE :N 0 PARE**

0 não é menor do que **0**. Portanto, passe à linha seguinte.

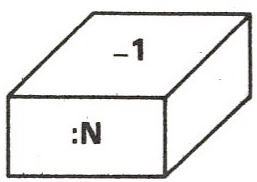
9. **LINHA 2 — NÍVEL 3 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** (que é **0**)



10. **LINHA 3 — NÍVEL 3 ⟶ CONT :N-1**

Reinicie o procedimento **CONT** , atribuindo:
À variável **:N** seu próprio valor **-1**.
Teremos:



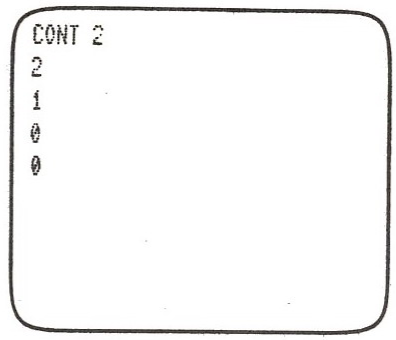
11. **LINHA 1 — NÍVEL 4 ⟶ SE :N 0 PARE**

-1 é menor do que **0**. Devemos então parar.

Chegamos ao final da resolução do **NÍVEL 4** .
Retorne ao **NÍVEL 3** para completá-lo.

12. **LINHA 4 — NIVEL 3 ⟶ MO :N**

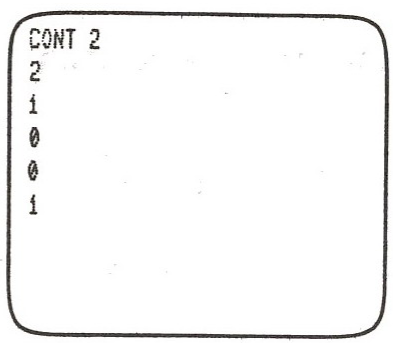
Mostre no vídeo o conteúdo da variável **:N** no **NÍVEL 3** (que é **0**).



A próxima ordem é **FIM** , o que indica o final da resolução do **NÍVEL 3** .
Retornar ao **NÍVEL 2** para completá-lo.

13. **LINHA 4 — NÍVEL 2 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** no **NIVEL 2** (que é **1**).



A próxima ordem é **FIM** , o que indica o final da resolução do **NÍVEL 2** .
Retorne ao **NÍVEL 2** para completá-lo.

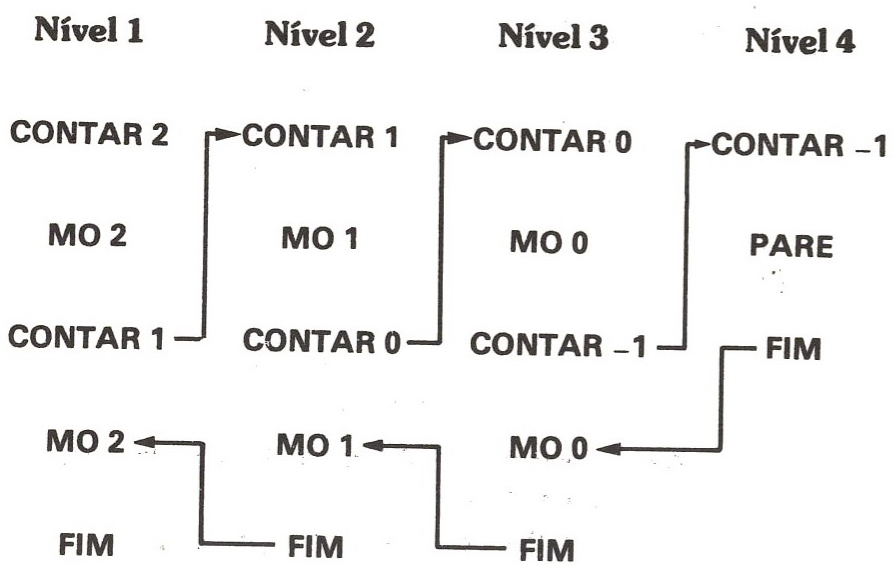
14. **LINHA 4 — NÍVEL 1 ⟶ MO :N**

Mostre no vídeo o conteúdo da variável **:N** no **NÍVEL 1** (que é **2**).

```
CONT 2
2
1
0
0
1
2
?
```

A próxima ordem é **FIM** , o que indica o final da resolução do **NÍVEL 1** e conseqüentemente o final da execução do procedimento **CONT** , tornando a aparecer a interrogação (**?**) e o cursor.

Podemos, então, montar o seguinte esquema:



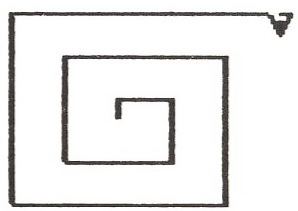
No procedimento **LABIRINTO :L** , vamos incluir algumas ordens após a recursão. Note sua execução antes e depois das inclusões:

```
AP LABIRINTO :L
SE :L > 100 ENTAO PARE
FR :L
DI 90
LABIRINTO :L+10
FIM
```

Digite:

```
LABIRINTO 10
```

Obteremos ⟶



A tartaruga termina a execução na posição em que desenhou o último lado do **LABIRINTO** e girou 90 graus.

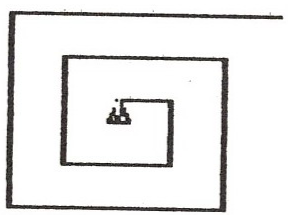
Vamos incluir **ES 90 VO :L** após a recursão:

```
AP LABIRINTO :L
SE :L > 100 ENTAO PARE
FR :L
DI 90
LABIRINTO :L+10
ES 90
VO :L
FIM
```

Digite:

```
LABIRINTO 10
```

Obteremos →



A tartaruga termina a execução na mesma posição em que a iniciou.

CAPÍTULO 3

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
FACA "X n		Cria a variável de nome **X** e conteúdo **n**.

DEFININDO VARIÁVEIS

Em LOGO, há duas maneiras de se atribuir valores a uma variável. A primeira delas, conforme já vimos, consiste em ativar o procedimento com a variável no seu título. É a variável local, que possui este nome por conservar seu valor somente enquanto o procedimento a que pertence estiver sendo executado. Ela deixa de existir assim que a execução se completa. Exemplo:

```
AP TRT :I
REPITA 3 [ FR :L DI 120 ]
FIM
```

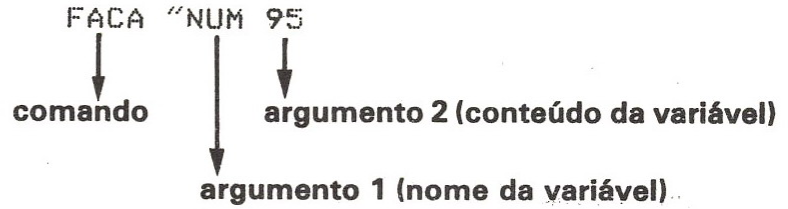
Neste caso, determinamos o valor de **:L** na própria ordem de ativação do procedimento. Assim, **TRI 50** significa que o valor **50** foi atribuído junto ao nome do procedimento.

A segunda maneira envolve uma nova primitiva do LOGO. É o comando **FACA** . Este comando define variáveis globais, que ao contrário das locais, não têm sua validade limitada a nenhum procedimento, estando associadas ao nível principal e conservando seu valor, independente da execução do procedimento.

O comando **FACA** necessita de dois argumentos como entrada:

Argumento 1 ⟶ O nome da variável à qual queremos atribuir determinado valor, nome este que deve ser precedido de aspas (").

Argumento 2 ⟶ O valor em questão. Exemplo:



Onde:

"NUM é o nome da variável e **95** é o conteúdo a ser guardado na variável **:NUM** :

```
FACA "NOME "CARLOS
```

Onde **"NOME** é o nome da variável e **CARLOS** é a palavra a ser guardada na variável **:NOME** :

```
FACA "LISTA [LEAO TIGRE [BICHO
PREGUICA] ONCA]
```

Onde **"LISTA** é o nome da variável e **[LEAO TIGRE [BICHO PREGUICA] ONCA]** é a lista a ser guardada na variável **:LISTA**

Observe que se quisermos definir um nome composto como elemento de uma lista, este deverá vir também em forma de lista (entre colchetes), caso contrário cada elemento que o forma será identificado individualmente. Assim, se **BICHO PREGUICA** não estivesse entre colchetes, formariam dois elementos da lista **:LISTA** .

Se quisermos mostrar no vídeo o conteúdo das variáveis definidas através do comando **FACA** , basta digitarmos:

```
MO :NUM
```
Obteremos ⟶
```
95
```

```
MO :NOME
```
Obteremos ⟶
```
CARLOS
```

```
MO :LISTA
```
Obteremos ⟶
```
LEAO TIGRE [BICHO
PREGUICA] ONCA]
```

Podemos, também, através do comando **FACA** utilizar as regras já vistas para operações aritméticas. Digite:

```
FACA "RESULTADO 4*(3+2)/(2+3)+1
MO :RESULTADO
```
Obteremos ⟶
```
5
```

EXPERIMENTE!!!

Digite, agora:

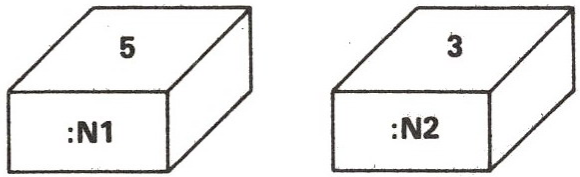
```
FACA "AREA 5*8
FACA "VOLUME :AREA * 3
MO :VOLUME
```
Obteremos ⟶
```
120
```

Vamos então construir procedimentos que utilizem o comando **FACA** :

I.
```
AP MULTIPLICAR :N1 :N2
FACA "MULT :N1 * :N2
MO :MULT
FIM
```

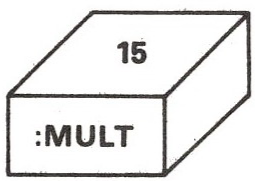

Digite **MULTIPLICAR 5 3** e acompanhe sua execução.

1. **VISUALIZANDO AS VARIÁVEIS E SEUS CONTEÚDOS**



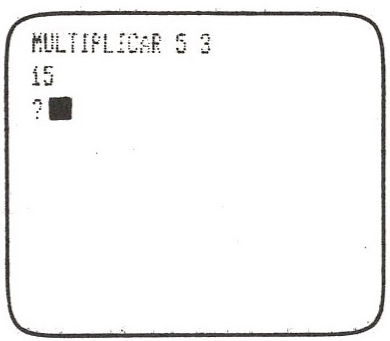
2. **LINHA 1 – NÍVEL 1 ⟶ FACA "MULT :N1 * :N2**

   Crie a variável **:MULT** e atribua a ela o valor resultante do produto de **:N1 por :N2** .
   Teremos:



3. **LINHA 2 – NÍVEL 1 ⟶ MO :MULT**

   Mostre no vídeo o conteúdo da variável **:MULT**



II.
```
AP QUADRADO :N
FACA "Q :N * :N
MO :Q
FIM
```

Digitando:

```
QUADRADO 5
```

Obteremos ⟶ 25

III.
```
AP VIZINHOS :N
FACA "A :N-1
FACA "S :N+1
MO [ ]
( MO [O ANTECESSOR DE] :N "E' :A )
MO [ ]
( MO [O SUCESSOR DE] :N "E' :S )
FIM
```

Digitando:

```
VIZINHOS 10
```

Obteremos ⟶

```
O ANTECESSOR DE 10 E' 9

O SUCESSOR DE 10 E' 11
```

SUGESTÕES

1. Analise o seguinte procedimento:

```
AP DOBRO
FACA "N 1 + SORTEIE 20
MO [ ]
( MO [ O NUMERO SORTEADO FOI ] :N )
MO [ ]
( MO [ E O DOBRO DE ] :N "E' 2 * :N )
FIM
```

2. Construa um procedimento que calcule a soma de dois números sorteados.
3. Acrescente, ao procedimento da sugestão 2, o cálculo do produto dos mesmos números.

CAPÍTULO 4

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
TESTE nnn		Responde **VERD** se nnn for verdadeiro e **FALSO** caso contrário.
SEVERD	**SV**	Determina o caminho que o programa deve seguir se **TESTE** responder **VERD** .
SEFALSO	**SF**	Determina o caminho que o programa deve seguir se **TESTE** responder **FALSO** .
(		Operação lógica **MENOR QUE** .
)		Operação lógica **MAIOR QUE** .
=		Operação lógica **IGUAL A** .
SETODOS a b c		Responde **VERD** somente se todos os dados (a, b, c...) forem verdadeiros.
SEUM a b c		Responde **VERD** se pelo menos um dos dados (a, b, c...) for verdadeiro.

MAIS SOBRE CONDICIONAIS

Quando impomos uma condição para que determinado fato ocorra, contamos sempre com duas alternativas. Veja:

SE meu salário for maior do que Cz$ 5000,00, **ENTAO** comprarei uma camisa, **SENAO** terei que trabalhar mais.

Vamos agora dividir a condição acima.

Tomemos primeiro a afirmativa:

Meu salário é maior do que Cz$ 5000,00.

Podemos tirar as seguintes conclusões:

1. A afirmativa é verdadeira, isto é, meu salário é realmente maior do que Cz$ 5000,00. Então poderei comprar a camisa.
2. A afirmativa é falsa, isto é, meu salário não é maior do que Cz$ 5000,00. Então, terei que trabalhar mais.

Vamos, assim, formular de outro modo a mesma frase:

O meu salário é maior do que Cz$ 5000,00.
SE FOR VERDADEIRA comprarei uma camisa.
SE FOR FALSA terei que trabalhar mais.

Também o LOGO nos permite decidir qual a melhor forma de estabelecermos condições em nossos procedimentos. Note:

```
AP CONTAR :N
SE :N > 10 ENTAO PARE
MO :N
CONTAR :N+1
FIM
```

```
AP CONTAR :N
TESTE :N>10
SEVERD PARE
SEFALSO MO :N
CONTAR :N+1
FIM
```


Em ambos os casos, seguimos os seguintes passos:

a. Especificamos a afirmativa a ser avaliada **(:N > 10)**
b. Determinamos o que deverá ser feito, caso a afirmativa seja verdadeira **(PARE)** .
c. Determinamos o que deverá ser feito, caso a afirmativa seja falsa **(MO :N)**

Observação: As condições **SE...ENTAO...SENAO** devem obrigatoriamente ser escritas numa única linha do procedimento, ao passo que **TESTE** , **SEVERD (SV)** , **SEFALSO (SF)** são três instruções distintas, sendo conveniente escrevê-las em linhas diferentes.

Tomemos como exemplo:

```
AP LABIRINTO :L
SE :L > 50 ENTAO PARE SENAO FR :L DI 90
LABIRINTO :L+5
FIM
```

Substituindo **SE...ENTAO...SENAO** por **TESTE** :

```
AP LABIRINTO :L
TESTE :L > 50
SV PARE
SF FR :L DI 90
LABIRINTO :L+5
FIM
```

Nas estruturas condicionais que acabamos de ver **(SE... ENTAO...SENAO e TESTE, SV, SF)** , devemos atribuir uma expressão que determine o caminho a ser seguido pelo procedimento.

No capítulo 1, definimos operações lógicas como sendo todas aquelas que respondam com **VERD** ou **FALSO** .

As mais usadas por nós são:

I. < : necessita de dois argumentos numéricos. Um deverá ser colocado à sua esquerda e o outro à sua direita. A resposta será **VERD** somente quando o argumento que estiver à esquerda for menor que o da direita.

Como exemplo, digite:

```
2 < 3
IGUAL:VERD
```
Obteremos →

```
MO 7 < 3
FALSO
```
Obteremos →

```
MO 27+3 < 100/2
VERD
```
Obteremos →

```
-2 < 1
IGUAL:VERD
```
Obteremos →

```
-2 < -3
IGUAL:FALSO
```
Obteremos →

II. > : da mesma forma que < , necessita de dois argumentos numéricos, respondendo **VERD** se o primeiro argumento for maior do que o segundo e **FALSO** caso contrário.

Como exemplo, digite:

```
7 > 5
IGUAL:VERD
```
Obteremos →


```
MO 9 > 20
FALSO
```
Obteremos →

```
13+3 > 10*2
IGUAL:FALSO
```
Obteremos →

```
2+3 > 5-2
IGUAL:VERD
```
Obteremos →

III. = : responde **VERD** somente se o primeiro argumento for igual ao segundo. Diferente das operações anteriores, os argumentos não precisam ser necessariamente numéricos, mas sim qualquer elemento de que o LOGO dispõe. Porém a resposta só será **VERD** se os elementos possuírem a mesma natureza.

Como exemplo, digite:

```
"OLA = [OLA]
IGUAL:FALSO
```
Obteremos →

```
"OLA = "OLA
IGUAL:VERD
```
Obteremos →

```
MO [DIA] = [DIA]
VERD
```
Obteremos →

```
MO 12 = "12
VERD
```
Obteremos →

```
MO 12 = [12]
FALSO
```
Obteremos →

Lembre-se que número é um caso particular de palavra, e não de lista, portanto **12** é igual a **"12** , mas **12** é diferente de **[12]** .

OPERAÇÕES LÓGICAS PARTICULARES

Existem duas primitivas (operações lógicas) que operam sobre os resultados de outras operações lógicas.

Como veremos a seguir, pode ser interessante formularmos questões do tipo:

1. **SE** amanhã não chover **E** eu estiver de carro, **ENTÃO** irei ao clube.

Observe a necessidade das duas condições serem satisfeitas para que eu vá ao clube, ou seja:

a. Não pode chover.

b. Eu tenho que estar de carro.

Se uma delas não for satisfeita, eu não irei ao clube.

2. **SE** amanhã não chover **OU** se eu estiver de carro **ENTÃO** irei ao clube.

Neste caso, basta que apenas uma das condições seja satisfeita para eu ir ao clube. Portanto, eu só não irei ao clube se chover **E** se eu não estiver de carro.

As operações lógicas em LOGO correspondentes aos conectivos **E** e **OU** são, respectivamente, **SETODOS** e **SEUM** .

I. **SETODOS:** esta operação necessita de dois ou mais valores lógicos como entrada, respondendo **VERD** somente se todos os argumentos determinados forem verdadeiros. Exemplo:


```
SETODOS 3<4 4<5
```
Obteremos → `IGUAL?VERD`

pois: **3 < 4 → VERD**
4 < 5 → VERD

```
SETODOS -2<-3 3<4
```
Obteremos → `IGUAL?FALSO`

pois: **-2 < -3 → FALSO**
3 < 4 → VERD

```
SETODOS "C = "C "V = "V
```
Obteremos → `IGUAL?VERD`

pois: **"C = "C → VERD**
"V = "V → VERD

Se quisermos determinar mais de dois argumentos, devemos colocá-los juntamente com a operação entre parênteses. Exemplo:

```
( SETODOS 2<3 3<4 7>6 10>3 )
```
Obteremos → `IGUAL?VERD`

pois: **2 < 3 → VERD**
3 < 4 → VERD
7 > 6 → VERD
10 > 3 → VERD

II. **SEUM:** necessita de dois ou mais valores lógicos como entrada, respondendo **VERD** se ao menos um dos argumentos for verdadeiro. Sendo assim, a resposta será **FALSO** somente se todos os argumentos forem falsos. Exemplo:

```
SEUM 3<4 7>10
IGUAL:VERD
```

Obteremos ⟶

pois: **3 < 4 ⟶ VERD**
7 > 10 ⟶ FALSO

```
SEUM 5=6 9<0
IGUAL:FALSO
```

Obteremos ⟶

pois: **5=6 ⟶ FALSO**
9<0 ⟶ FALSO

```
( SEUM 1<2 2>9 5=2 )
IGUAL:VERD
```

Obteremos ⟶

pois: **1 < 2 ⟶ VERD**
2 > 9 ⟶ FALSO
5=2 ⟶ FALSO

```
SEUM "RUA = "RUA "AVE = [AVE]
IGUAL:VERD
```

Obteremos ⟶

pois: **"RUA="RUA ⟶ VERD**
"AVE=[AVE] ⟶ FALSO

Usando as operações que acabamos de ver, faremos procedimentos que resolvam os seguintes problemas:

1. A porcentagem de desconto do salário de determinado funcionário varia segundo a tabela abaixo:

SALÁRIO (em mil cruzados)	**%DESCONTO**
ENTRE 0 E 20	**5**
ENTRE 20 E 40	**10**
ENTRE 40 E 60	**15**
ACIMA DE 60	**20**

Qual é esse desconto?

```
AP SALARIO :S
TESTE SETODOS 0<:S :S<20
SV ( MO "DESCONTO: :S*5/100 ) PARE
SF TESTE SETODOS 20<:S :S<40
SV ( MO "DESCONTO: :S*10/100 ) PARE
SF TESTE SETODOS 40<:S :S<60
SV ( MO "DESCONTO: :S*15/100 ) PARE
SF ( MO "DESCONTO: :S*20/100 )
FIM
```

Digite:

```
SALARIO 30
DESCONTO: 3
```

Obteremos ⟶

2. Qual o quociente e o resto da divisão de dois números?

```
AP DIVISAO :A :B
TESTE SEUM :B<0 :B>0
SV ( MO "QUOCIENTE: QUOC :A :B "RESTO:
RESTO :A :B )
SF MO [ NAO EXISTE DIVISAO POR ZERO ]
FIM
```

Digite:

```
DIVISAO 9 2
```

Obteremos → `QUOCIENTE: 4 RESTO: 1`

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
PALAVRA a b		Responde com a palavra formada pelas palavras **a** e **b**.
SENTENCA a b	**SN**	Responde com a lista formada pelas palavras e/ou listas **a** e **b**.
PRIMEIRO a		Responde com o primeiro elemento da palavra ou lista **a**.
ULTIMO a		Responde com o último elemento da palavra ou lista **a**.
SEMPRIMEIRO a	**SP**	Responde com todos os elementos, exceto o primeiro da palavra ou lista **a**.
SEMULTIMO a	**SU**	Responde com todos os elementos, exceto o último da palavra ou lista **a**.

OPERANDO COM PALAVRAS E LISTAS

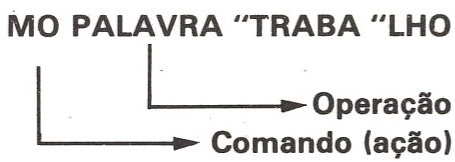
O LOGO contém uma série de primitivas que nos possibilita a manipulação de palavras e/ou listas, examinando-as ou modificando-as. Elas são particularmente úteis para processamento de textos. Estas primitivas são operações e, portanto, respondem com um valor que deverá ser utilizado como argumento de comandos.

Veremos a princípio duas delas:

I. **PALAVRA:** a operação **PALAVRA** necessita de dois dados como entrada. Esses dados devem ser necessariamente palavras, que nos são devolvidos numa única palavra, resultante da junção das primeiras. Assim:

```
MO PALAVRA "TRABA "LHO
```

Obteremos → TRABALHO



Como ocorre?

Para duas entradas há apenas uma saída. Observe que esta saída é sempre uma palavra, sendo conveniente lembrar que, para o LOGO, palavra é uma seqüência ordenada de caracteres que não inclui o espaço em branco.

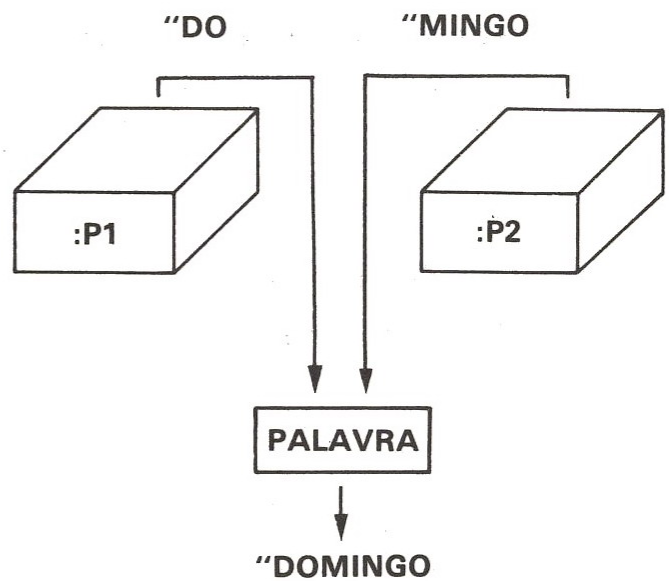
PALAVRA utiliza sempre palavras, nunca listas, como entrada.

Isto também é válido para variáveis, desde que o conteúdo destas sejam palavras. Exemplo:

```
FACA "P1 "DO
FACA "P2 "MINGO
MO PALAVRA :P1 :P2
```

Obteremos → DOMINGO

Como ocorre?



Podemos, ainda, associar o conteúdo de variáveis a outra palavra:

```
MO PALAVRA :P1 "CE
DOCE
```
Obteremos ⟶

```
MO PALAVRA "CE :P1
CEDO
```
Obteremos ⟶

Em certos casos, precisamos determinar mais de duas entradas ao comando **PALAVRA** , a fim de obtermos uma única palavra como resposta.

Para tanto, devemos colocar a operação, juntamente com seus argumentos, entre parênteses, deixando sempre um espaço em branco entre estes e as ordens. Exemplo:

```
MO ( PALAVRA "TRA "BA "LHO )
TRABALHO
```
Obteremos ⟶

Se não procedermos desta forma, ao digitar:

```
MO PALAVRA "TRA "BA "LHO
TRABA
IGUAL:LHO
```
Obteremos ⟶

SUGESTÃO

Digite as instruções que se seguem:

```
FACA "P1 "CA
FACA "P2 "MENTO
FACA "P3 "SAL
FACA "P4 "VA
FACA "P5 "LOR
FACA "P6 "LO
```


Agora, usando o comando **PALAVRA** , forme várias palavras, como no exemplo:

```
MO ( PALAVRA :P1 :P4 :P6 )
CAVALO
```
Obteremos ⟶

II. **SENTENÇA (SN):** a operação **SENTENCA** , tal como **PALAVRA** , necessita de dois argumentos como entrada. Estes argumentos podem ser:

a. Duas listas:

```
MO SN [BOM DIA] [PARA VOCE]
BOM DIA PARA VOCE
```
Obteremos ⟶

b. Uma lista e uma palavra:

```
MO SN [BOM DIA] "FABIO
BOM DIA FABIO
```
Obteremos ⟶

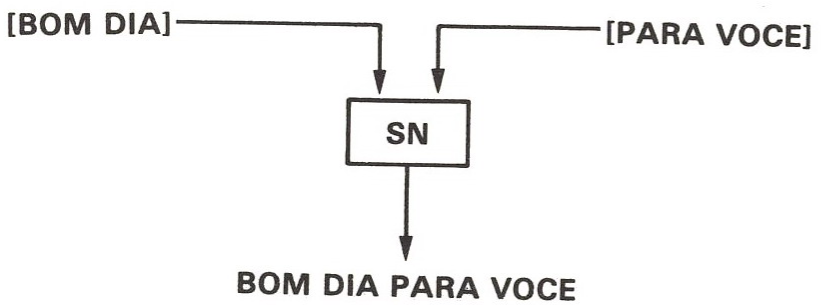
c. Duas palavras:

```
MO SN "BOM "DIA
BOM DIA
```
Obteremos ⟶

devolvendo uma única lista, proveniente da união de ambos.

Seu funcionamento é semelhante ao da operação **PALAVRA** .

Veja o esquema da opção a:



Portanto, também aqui, para duas entradas há apenas uma saída, sendo que neste caso permanecem os espaços entre as palavras.

Para obtermos uma **SENTENCA** resultante de mais de duas entradas, procederemos exatamente como na operação **PALAVRA** .

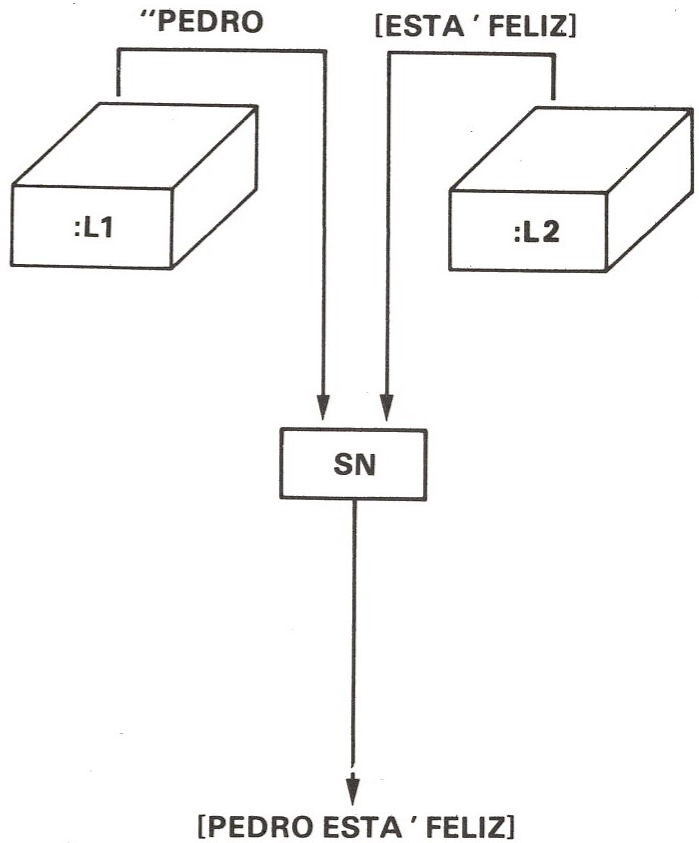
EXPERIMENTE!!!

```
MO ( SN "HOJE [O DIA] [ESTA'
LINDO] )
```
Obteremos → `HOJE O DIA ESTA' LINDO`

Podemos usar variáveis como argumento para **SENTENCA** . Exemplo:

```
FACA "L1 "PEDRO
FACA "L2 [ESTA' FELIZ]
MO SN :L1 :L2
```
Obteremos → `PEDRO ESTA' FELIZ`

Esquematizando:

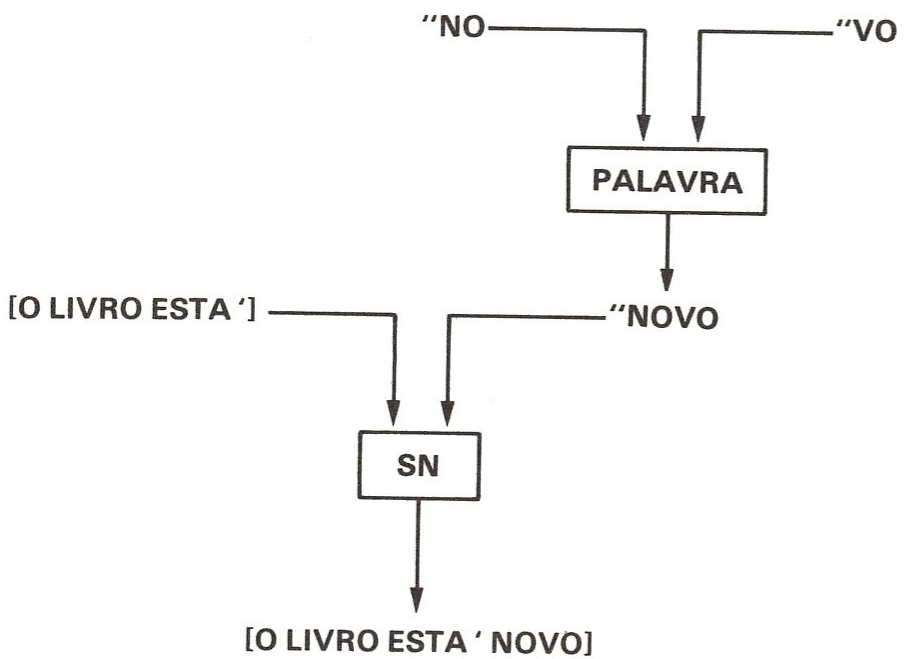


Será possível a construção da seguinte sentença?

```
MO SN [O LIVRO ESTA'] PALAVRA "NO "VO
```

A resposta é afirmativa, pois **SENTENCA** pode ter palavras como entrada.

Esquematizando:



Agora, entraremos em contato com outras quatro operações muito interessantes e fáceis de usar, antes, porém, é conveniente recordarmos que **PALAVRA** e **LISTA** são conjuntos ordenados, ou seja, é importante a ordem em que seus elementos são apresentados.

É diferente **VOA** de **AVO** (embora sejam conjuntos que contenham os mesmos caracteres).

É diferente **[ATE LOGO]** de **[LOGO ATE]**.

Todas estas operações utilizam-se de alguma forma, da posição que os elementos ocupam no conjunto em questão (**PALAVRA** ou **LISTA**).

São elas:

I. **PRIMEIRO:** dá como resposta o primeiro elemento da lista ou da palavra determinada como entrada. Exemplo:

```
MO PRIMEIRO "CASA
C
```
Obteremos→

```
MO PRIMEIRO [ABACAXI PERA BANANA]
ABACAXI
```
Obteremos→

```
FACA "P "BOM
MO PRIMEIRO :P
B
```
Obteremos→

```
FACA "P PRIMEIRO "BOM
MO :P
B
```
Obteremos→

Apesar da resposta ter sido a mesma nos dois últimos exemplos, convém notar que no primeiro, ao dizer **FACA "P "BOM** , determinamos que a palavra .**BOM** seja o conteúdo da variável **:P** , enquanto no segundo **(FACA "P PRIMEIRO "BOM)** o conteúdo de **:P** é apenas a letra **B** , ou seja, o primeiro elemento da palavra **BOM** .

O mesmo ocorre com listas.

Digite:

```
FACA "L [RADIO TV COMPUTADOR]
MO PRIMEIRO :P
RADIO
```
Obteremos→

O conteúdo da variável **:L** é a lista **[RADIO TV COMPUTADOR]**

```
FAÇA "L PRIMEIRO [RADIO
TV COMPUTADOR]
MO :L
RADIO
```

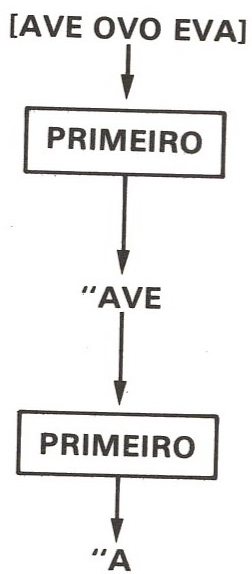
Obteremos →

O conteúdo da variável **:L** , agora, é a palavra **RADIO** .

O que acontece se digitarmos:

```
MO PRIMEIRO PRIMEIRO [AVE OVO EVA]
```

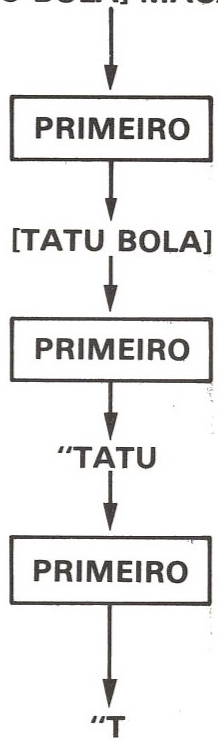
ou seja, mostre no vídeo (ação) o primeiro elemento (operação) do primeiro elemento (operação) da lista **[AVE OVO EVA]** . Veja o esquema:



O que será mostrado no vídeo é a letra **A** .


```
MO PRIMEIRO PRIMEIRO PRIMEIRO
[[TATU BOLA] MACACO]
```

[[TATU BOLA] MACACO]



II. **ULTIMO:** envia como resposta o último elemento da lista ou da palavra determinada como entrada. Exemplo:

```
MO ULTIMO "CASA
A
```

Obteremos →

```
MO ULTIMO 123
3
```

Obteremos →

```
MO ULTIMO [CASA]
CASA
```

Obteremos →

```
MO ULTIMO [ABACAXI [LARANJA LIMA]]
LARANJA LIMA
```

Obteremos →

Lembre-se de que a lista **[ABACAXI [LARANJA LIMA]]** só possui dois elementos: a palavra **ABACAXI** , e a lista **[LARANJA LIMA]** , o último elemento é a lista **LARANJA LIMA** .

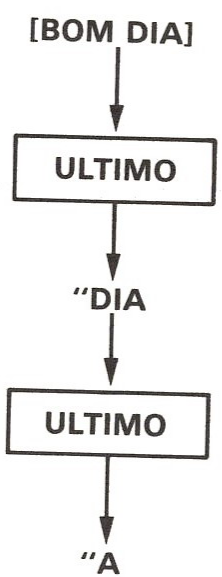
```
FACA "U ULTIMO [BOM DIA]
MO :U
```

Obteremos → `DIA`

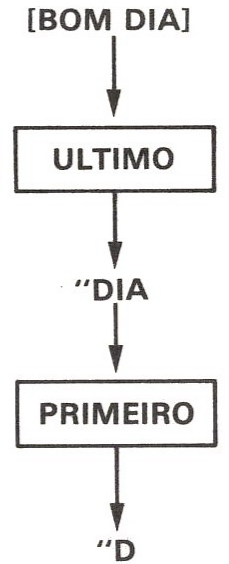
```
FACA "U "BOM
MO ULTIMO :U
```

Obteremos → `M`

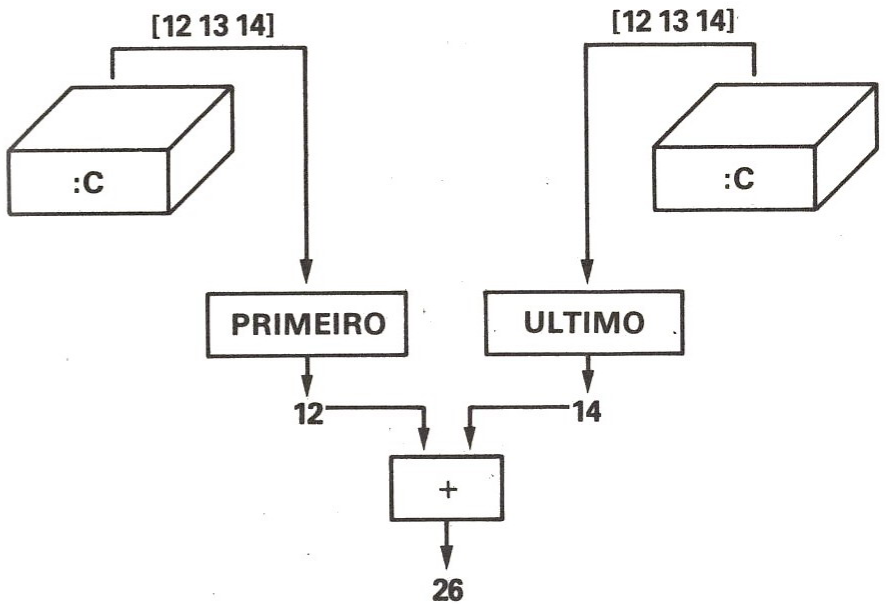
```
MO ULTIMO ULTIMO [BOM DIA]
```




```
MO PRIMEIRO ULTIMO [BOM DIA]
```




```
FACA "C [12 13 14]
MO PRIMEIRO :C + ULTIMO :C
```



III. **SEMPRIMEIRO:** dá como resposta todos os elementos da lista ou da palavra, com exceção do primeiro. Pode ser abreviado por **SP** Exemplo:

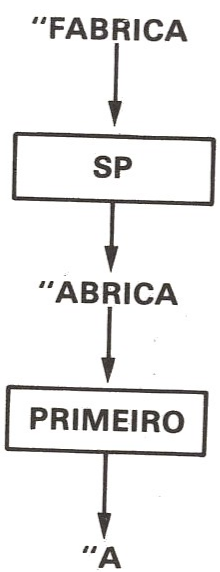
```
MO SP "CANETA
```
Obteremos → ANETA

```
MO SP [A B C D]
```
Obteremos → B C D

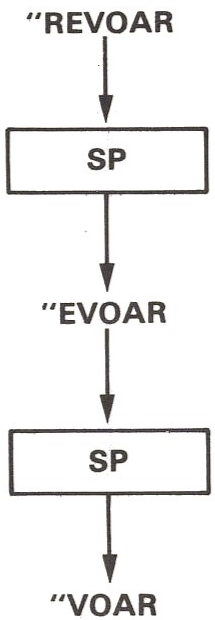
```
MO SP 123
```
Obteremos → 23

```
MO PRIMEIRO SP "FABRICA
```
Obteremos → A

Observe o esquema:




```
MO SP SP "REVOAR
```
Obteremos → VOAR



IV. **SEMULTIMO:** dá como resultado todos os elementos da lista ou palavra, exceto o último. Pode ser abreviado por **SU** . Exemplo:

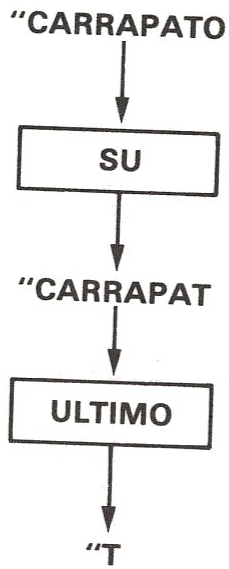
```
MO SU "CAMINHOS
```
Obteremos → CAMINHO

```
MO SU [SAPO RAPOSA BORBOLETA]
```
Obteremos → SAPO RAPOSA

```
MO SU 451
```
Obteremos → 45

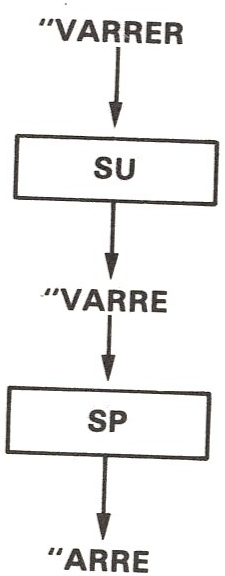
Obteremos →

```
MO ULTIMO SU "CARRAPATO
T
```



Obteremos →

```
MO SP SU "VARRER
ARRE
```



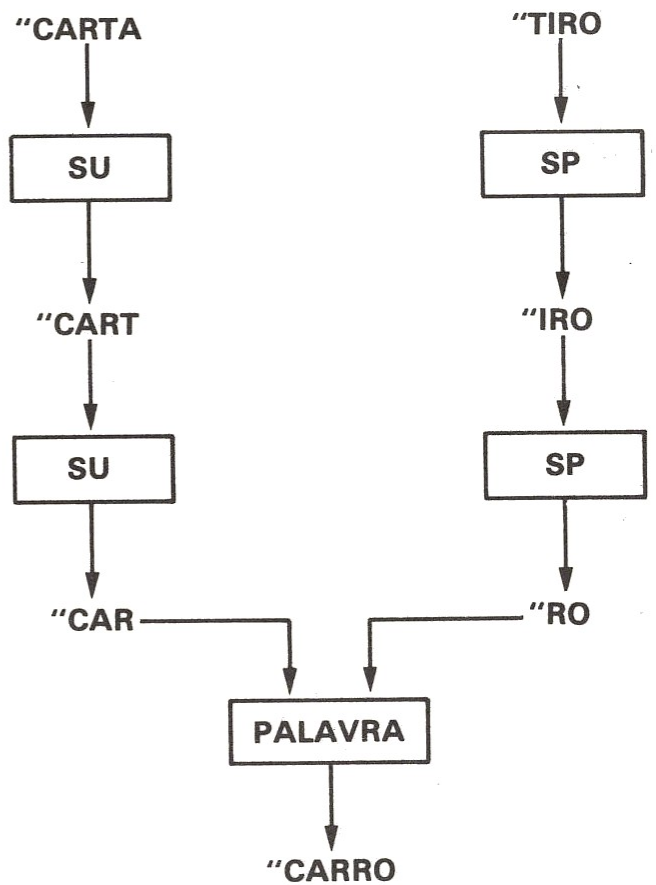
JUNTANDO OPERAÇÕES

Com estas seis operações que acabamos de conhecer, podemos fazer coisas muito interessantes:

```
MO PALAVRA SU SU "CARTA SP SP
"TIRO
```

Obteremos → `CARRO`

OBSERVE!



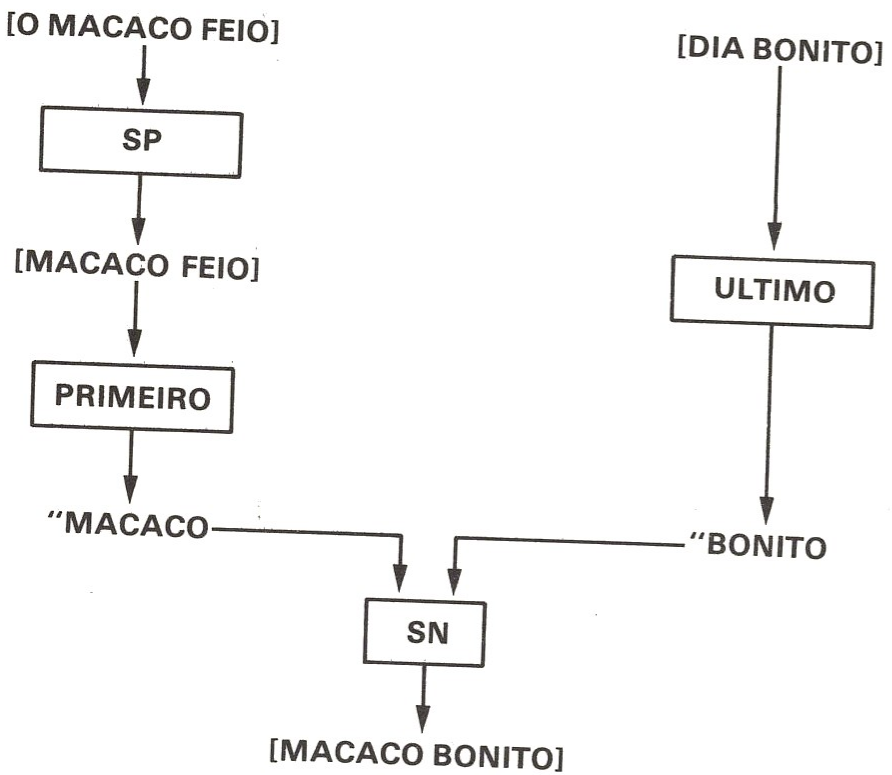
Tente descobrir o resultado das ordens abaixo, depois teste no seu computador:

1. `MO PALAVRA SP SP SP SU SU "DIALOGO SU SU "GOTA`
2. `MO ( PALAVRA PRIMEIRO "NAVE PRIMEIRO SP "CAVIAR SP SP "NEVE )`
3. `MO ( PALAVRA SU SU "TROPA SP SP SU "CAPIM SP SP "LOCAL )`

```
MO SN PRIMEIRO SP [O MACACO
FEIO] ULTIMO [DIA BONITO]
```

Obteremos → `MACACO BONITO`

OBSERVE!

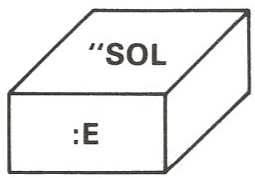


Vamos elaborar alguns procedimentos que utilizem estas operações:

I.
```
AP TRIP :E
SE :E = " PARE
MO :E
TRIP SU :E
FIM
```

Digite **TRIP "SOL** e passemos a acompanhar a resolução:

1. **VISUALIZANDO A VARIÁVEL E SEU CONTEÚDO**



2. **LINHA 1 — NÍVEL 1 → SE :E = " PARE**

   O conteúdo de **:E ("SOL)** não é uma palavra vazia (").

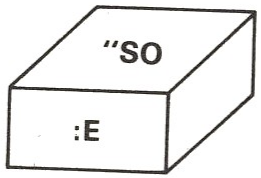
Portanto, passe à linha seguinte.

3. **LINHA 2 — NÍVEL 1 → MO :E**

   Mostre no vídeo o conteúdo da variável **:E** (que é **"SOL**)

4. **LINHA 3 — NÍVEL 1 ⟶ TRIP SU :E**

Reinicie o procedimento **TRIP** , atribuindo à variável **:E** seu próprio conteúdo, sem o último elemento. Teremos:



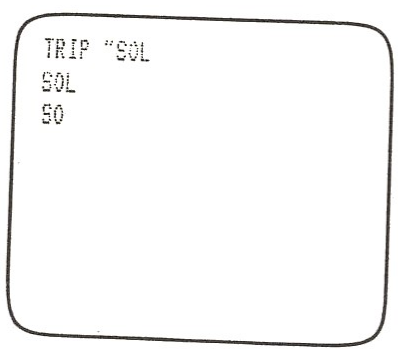
5. **LINHA 1 — NÍVEL 2 ⟶ SE :E = "PARE**

O conteúdo de **:E ("SO)** não é uma palavra vazia (").

Portanto, passe à linha seguinte.

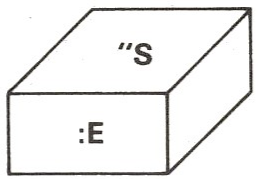
6. **LINHA 2 — NÍVEL 2 ⟶ MO :E**

Mostre no vídeo o conteúdo da variável **:E** (que é **"SO**).



7. **LINHA 3 — NÍVEL 2 ⟶ TRIP SU :E**

Reinicie o procedimento **TRIP** , atribuindo à variável **:E** seu próprio conteúdo sem o último elemento. Teremos:

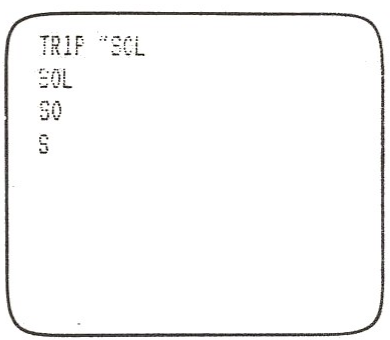


8. **LINHA 1 — NIVEL 3 ⟶ SE :E = " PARE**

O conteúdo de **:E ("S)** não é uma palavra vazia. Portanto, passe à linha seguinte.

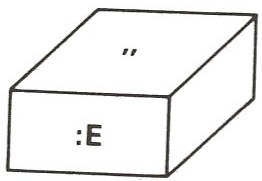
9. **LINHA 2 — NÍVEL 3 ⟶ MO :E**

Mostre no vídeo o conteúdo da variável **:E** (que é **"S**).



10. **LINHA 3 — NÍVEL 3 ⟶ TRIP SU :E**

Reinicie o procedimento **TRIP** , atribuindo à variável **:E** seu próprio conteúdo sem o último elemento. Teremos:



11. **LINHA 1 — NÍVEL 4 ⟶ SE :E = " PARE**

O conteúdo de **:E** é uma palavra vazia ("). Portanto, pare. Assim, chegamos ao final do procedimento **TRIP** , onde retornam ao vídeo a interrogação (**?**) e o cursor.

Note que só podemos determinar para a variável **:E** uma palavra, pois na **linha SE :E =" PARE** , comparamos a variável **:E** com uma **PALAVRA** vazia.

A fim de que possamos triangular listas, devemos apenas substituir a linha **SE :E =" PARE** por **SE :E =[] PARE** , onde iremos comparar se **:E** é uma **LISTA** vazia.

Então o procedimento **TRIL :E** ficará assim:

```
AP TRIL :E
SE :E = [] PARE
MO :E
TRIL SU :E
FIM
```


Digitando:

```
TRIL [O DIA ESTA' BONITO]
```

Obteremos ⟶

```
O DIA ESTA' BONITO
O DIA ESTA'
O DIA
O
```

II. Quantos elementos possui a palavra **"CALOR** e a lista **[GATO PATO RATO]** ?

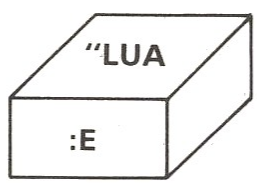
Vamos criar procedimentos que sejam capazes de nos responder às perguntas acima.

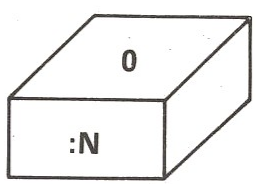
```
AP CONTEP :E :N
SE :E = " MO :N PARE
CONTEP SP :E :N+1
FIM
```

No procedimento acima, devemos determinar como valor à variável **:E** , a palavra que deverá ser contada e a **:N 0** , que determina o valor inicial da variável de contagem.

Digite **CONTEP "LUA 0** e acompanhe a resolução:

1. **VISUALIZANDO AS VARIÁVEIS E SEUS CONTEÚDOS**





2. **LINHA 1 — NÍVEL 1 ⟶ SE :E = " MO :N PARE**

O conteúdo de **:E ("LUA)** não é uma palavra vazia (").

Portanto, passe à linha seguinte.

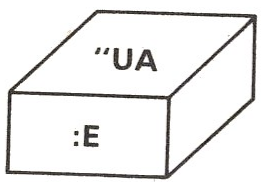
3. **LINHA 2 — NÍVEL 1 ⟶ CONTEP SP :E :N + 1**

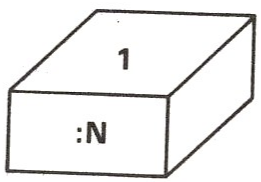
Reinicie o procedimento **CONTEP** , atribuindo:

a. À variável **:E** seu próprio conteúdo sem o primeiro elemento.

b. À variável **:N** seu próprio valor **+ 1**.

Teremos:





4. **LINHA 1 — NÍVEL 2 ⟶ SE :E = " MO :N PARE**

O conteúdo de **:E ("UA)** não é uma palavra vazia (").

Portanto, passe à linha seguinte.

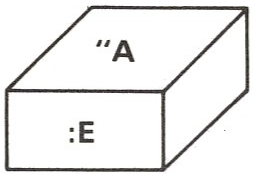
5. **LINHA 2 — NÍVEL 2 ⟶ CONTEP SP :E :N + 1**

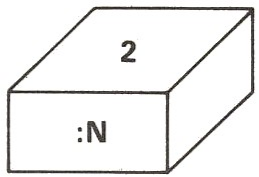
Reinicie o procedimento **CONTEP** , atribuindo:

a. À variável **:E** seu próprio conteúdo sem o primeiro elemento.

b. À variável **:N** seu próprio valor **+ 1**

Teremos:





6. **LINHA 1 — NÍVEL 3 ⟶ SE :E = " PARE**

O conteúdo de **:E ("A)** não é uma palavra vazia (").
Portanto, passe à linha seguinte.

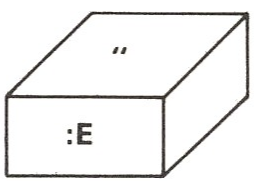
7. **LINHA 2 — NÍVEL 3 ⟶ CONTEP SP :E :N + 1**

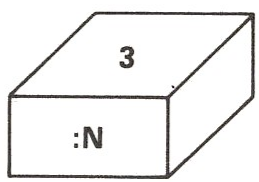
Reinicie o procedimento **CONTEP** , atribuindo:

a. À variável **:E** seu próprio conteúdo sem o último elemento.

b. À variável **:N** seu próprio valor **+ 1**

Teremos:

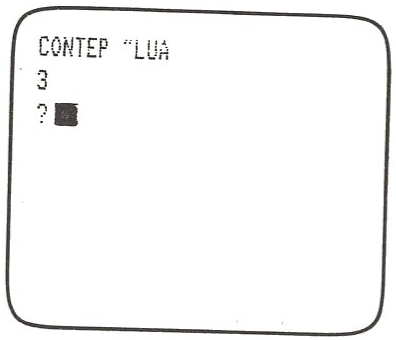




8. **LINHA 1 — NÍVEL 4 ⟶ SE :E = " MO :N PARE**

O conteúdo de **:E (")** é uma palavra vazia (").

Portanto, mostre no vídeo o conteúdo da variável **:N** (que é **3**) e pare.



Assim, chegamos ao final da resolução do procedimento **CONTEP** , onde reaparecem no vídeo a interrogação(**?**) e o cursor.

O mesmo devemos fazer para contar **LISTAS** , substituindo a linha **SE :E =" MO :N PARE** por **SE :E = [ ] MO :N PARE** , então **CONTEL :E :N** ficará assim:

```
AP CONTEL :E :N
SE :N = [] MO :N PARE
CONTEL SP :E :N+1
FIM
```

Digitando:

```
CONTEL [PAI PIA PAIS PIPA] 0
```

Obteremos ⟶ 4

III. Podemos ainda, como exemplo, verificar se a letra **"X** pertence à palavra **"CASA** ou se a palavra **"BRANCO** pertence à lista **[AZUL BRANCO AMARELO]** .

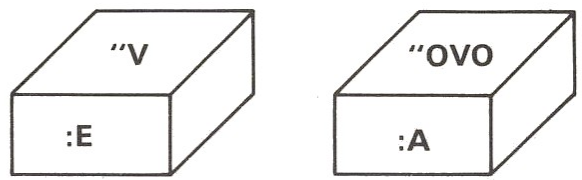
Para isso, veremos mais dois procedimentos, um para **PALAVRAS** e outro para **LISTAS** :

```
AP VERIFP :E :A
SE :A = " ENTAO ( MO :E [NAO PERTENCE A]
:A ) PARE
SE :E = PRIMEIRO :A ENTAO ( MO
:E [PERTENCE A] :A ) PARE
VERIFP :E SP :A
FIM
```

Digite:

```
VERIFP "V "OVO
```

1. **VISUALIZANDO AS VARIÁVEIS E SEUS CONTEÚDOS**



2. **LINHA 1 — NÍVEL 1 ⟶ SE :A = " ENTAO (MO :E [NAO PERTENCE A] :A) PARE**

   O conteúdo de **:A ("OVO)** não é uma palavra vazia (").

   Portanto, passe à linha seguinte.

3. **LINHA 2 — NÍVEL 1 ⟶ SE :E = PRIMEIRO :A ENTAO (MO :E [PERTENCE A] :A) PARE**

   O conteúdo de **:E ("V)** não é igual ao primeiro elemento de **:A ("O)** . Portanto, passe à linha seguinte.

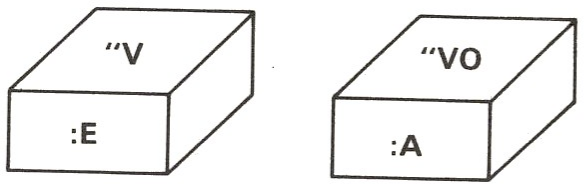
4. **LINHA 3 — NÍVEL 1 ⟶ VERIFP :E SP :A**

Reinicie o procedimento **VERIFP** , atribuindo:

a. À variável **:E** seu próprio valor.

À variável **:A** seu próprio valor sem o primeiro elemento.

Teremos:



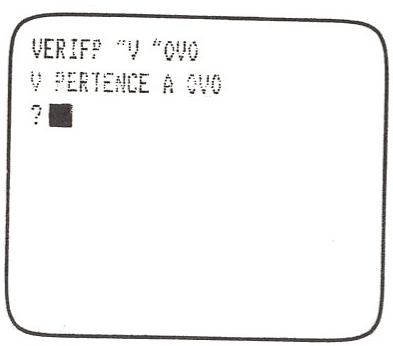
5. **LINHA 1 — NÍVEL 2 ⟶ SE :A = " ENTAO (MO :E [NAO PERTENCE A] :A) PARE**

O conteúdo de **:A ("VO)** não é uma palavra vazia (").

Portanto, passe à linha seguinte.

6. **LINHA 2 — NÍVEL 2 ⟶ SE :E = PRIMEIRO :A ENTAO (MO :E [PERTENCE A] :A) PARE**

O conteúdo de **:E ("V)** é igual ao primeiro de **:A ("V)** . Portanto, mostre no vídeo **V PERTENCE A OVO** e pare.



Para listas, substituiremos apenas a linha:

```
SE :A = " ENTAO ( MO :E [NAO PERTENCE A]
:A ) PARE
```

por

```
SE :A = [] ENTAO ( MO :E [NAO PERTENCE A]
:A ) PARE
```

então **VERIFL :E :A** ficará:

```
AP VERIFL :E :A
SE :A = [] ENTAO ( MO :E [NAO PERTENCE A]
:A ) PARE
SE :E = PRIMEIRO :A ( MO :E [PERTENCE A]
:A ) PARE
VERIFL :E SP :A
FIM
```

Digite:

```
VERIFL "PRETO [AZUL BRANCO
AMARELO]
```

Obteremos ⟶

```
PRETO NAO PERTENCE A AZUL
BRANCO AMARELO
```

ou seja, a palavra **"PRETO** , não pertence à lista **[AZUL BRANCO AMARELO]** .

IV. Com o comando **MOSTRAR** , vamos criar procedimentos que invertam palavras e listas. Exemplo:

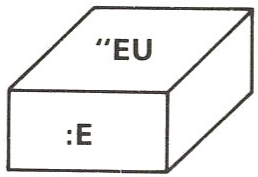
CASA — ASAC

BOM DIA — DIA BOM

```
AP INVP :E
SE :E = " REPITA 2 [MO []] PARE
MOSTRAR ULTIMO :E
INVP SU :E
FIM
```

Digite **INVP "EU** e acompanhe a resolução.

1. **VISUALIZANDO A VARIÁVEL E SEU CONTEÚDO**

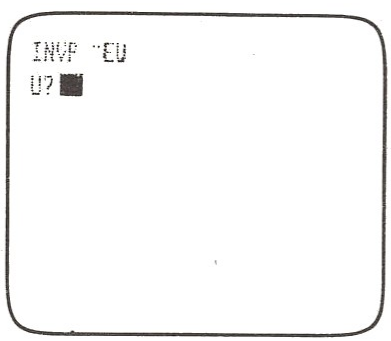


2. **LINHA 1 — NÍVEL 1 ⟶ SE :E = " REPITA 2 [MO[]] PARE**

   O conteúdo de **:E ("EU)** não é uma palavra vazia (").
   Portanto, passe à linha seguinte.

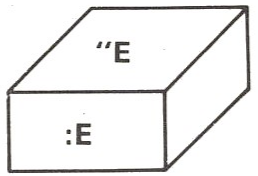
3. **LINHA 2 — NÍVEL 1 ⟶ MOSTRAR ULTIMO :E**

   Mostre no vídeo o último elemento de **:E ("U)** , mantendo o cursor logo após o caractere mostrado:



4. **LINHA 3 — NÍVEL 1 ⟶ INVP SU :E**

   Reinicie o procedimento **INVP** , atribuindo à variável **:E** seu próprio valor sem o último elemento. Teremos:



5. **LINHA 1 — NÍVEL 2 ⟶ SE :E = " REPITA 2 [MO [ ]] PARE**

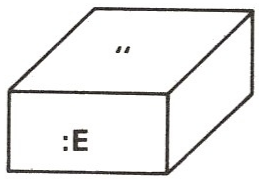
   O conteúdo de **:E ("E)** não é uma palavra vazia (").
   Portanto, passe à linha seguinte.

6. **LINHA 2 — NÍVEL 2 ⟶ MOSTRAR ULTIMO :E**

   Mostre no vídeo o último elemento de **:E** , mantendo o cursor logo após o caractere mostrado.

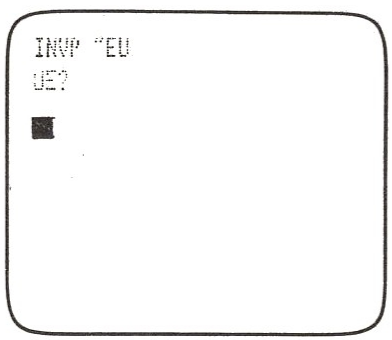
7. **LINHA 3 — NÍVEL 2 ⟶ INVP SU :E**

   Reinicie o procedimento **INVP** , atribuindo à variável **:E** seu próprio valor sem o último elemento. Teremos:



8. **LINHA 1 — NÍVEL 3 ⟶ SE :E = " REPITA 2 [MO [ ]] PARE**

   O conteúdo de **:E** (") é uma palavra vazia ("). Então pular duas linhas (mostrar duas listas vazias) e parar.



Lembre-se de que o comando **MOSTRAR** , junta elementos sem espaço de separação, então, para criarmos o procedimento **INVL :E** que inverta listas precisaremos fazer algumas modificações:

```
SE :E = " REPITA 2 [MO [ ]] PARE
```


será mudado para:

```
SE :E = [ ] REPITA 2 [MO [ ]] PARE

MOSTRAR ULTIMO :E
```

ficará:

```
MOSTRAR SN ULTIMO :E "
```

A palavra vazia (") ao final da ordem foi colocada para obtermos um espaço entre uma palavra e outra; caso contrário, se quiséssemos inverter **BOM DIA** obteríamos como resposta **DIABOM** .

Então; **INVL :E** ficará assim:

```
AP INVL :E
SE :E=[ ] REPITA 2 [ MO [ ]] PARE
MOSTRAR SN ULTIMO :E "
INVL SU :E
FIM
```

Digitando:

```
INVL [BOM TRABALHO]
```

Obteremos ⟶

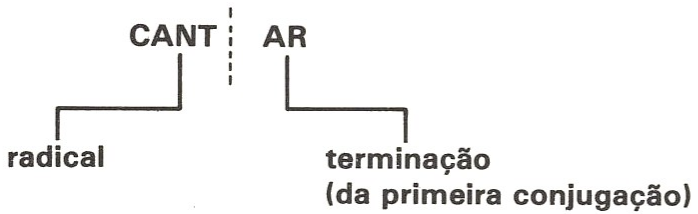
```
TRABALHO BOM
```

V. Você sabe conjugar verbos regulares de primeira conjugação no Presente?

   Vejamos:

   Tomemos como exemplo o verbo **CANTAR** .

O verbo **CANTAR** está no Infinitivo, assim, podemos dividi-lo em duas partes:



Conjugando-o no presente teremos:

PRONOME	RADICAL	TERMINAÇÃO DA PRIMEIRA CONJUGAÇÃO NO PRESENTE
EU	CANT	O
TU	CANT	AS
ELE	CANT	A
NOS	CANT	AMOS
VOS	CANT	AIS
ELES	CANT	AM

Os pronomes e as terminações variam de pessoa para pessoa, enquanto o radical permanece constante durante toda a conjugação.

Note que os pronomes e estas terminações são as mesmas para qualquer verbo regular terminado por **AR** . Para que obtenhamos a conjugação do verbo **BRINCAR** no presente, basta, no quadro acima, substituirmos **CANT** por **BRINC** .

Vamos ensinar o computador a conjugar verbos regulares da primeira conjugação?

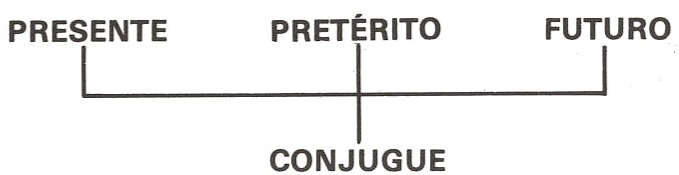
Faremos, inicialmente, procedimentos que apenas atribuirão valores às variáveis **:PRO :RAD :TER** . Veja que somente a variável **:TER** determinará o tempo em que o verbo será conjugado. As restantes permanecerão constantes em todos os procedimentos:

1.
```
AP PRESENTE :VERBO
FACA "PRO [EU TU ELE NOS VOS ELES]
FACA "RAD SU SU :VERBO
FACA "TER [O AS A AMOS AIS AM]
CONJUGUE :PRO :RAD :TER
FIM
```

2.
```
AP PRETERITO :VERBO
FACA "PRO [EU TU ELE NOS VOS ELES]
FACA "RAD SU SU :VERBO
FACA "TER [EI ASTE OU AMOS ASTES ARAM]
CONJUGUE :PRO :RAD :TER
FIM
```

3.
```
AP FUTURO :VERBO
FACA "PRO [EU TU ELE NOS VOS ELES]
FACA "RAD SU SU :VERBO
FACA "TER [AREI ARAS ARA AREMOS
AREIS ARAO]
CONJUGUE :PRO :RAD :TER
FIM
```

Ao final de todos os procedimentos, mandamos que o procedimento **CONJUGUE** seja executado, assim:



Precisamos, então, criar o procedimento **CONJUGUE** :

```
AP CONJUGUE :PRO :RAD :TER
SE :PRO = [] PARE
MO SN PRIMEIRO :PRO PALAVRA :RAD
PRIMEIRO :TER
CONJUGUE SP :PRO :RAD SP :TER
FIM
```

Por exemplo, se digitarmos:

a. `PRESENTE "BRINCAR`

Obteremos ⟶

```
EU BRINCO
TU BRINCAS
ELE BRINCA
NOS BRINCAMOS
VOS BRINCAIS
ELES BRINCAM
```

b. `FUTURO "ANDAR`

Obteremos ⟶

```
EU ANDAREI
TU ANDARAS
ELE ANDARA
NOS ANDAREMOS
VOS ANDAREIS
ELES ANDARAO
```

SUGESTÃO:

Tente, agora, criar procedimentos que conjuguem verbos regulares de outras conjugações.

CAPÍTULO 6

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
PEGUE	**PE**	Permite que se responda com um caractere durante a execução do programa.
ENTRE	**EN**	Permite que se responda com uma lista durante a execução do programa.
EXECUTE n		Executa as ordens contidas na lista **n** especificada.
CP?		Verifica se existe ou não algum caractere pendente.

DIALOGANDO

É possível fazermos procedimentos que aguardem uma entrada através do teclado. Desta forma, podemos estabelecer um diálogo com a máquina. Há algumas primitivas que permitem interagirmos com um procedimento em funcionamento.

São elas:

I. **PEGUE (PE):** esta operação instrui o computador para que aguarde um caractere que entrará através do teclado. Ela devolve uma palavra de um só caractere correspondente à tecla digitada.

EXPERIMENTE!!!

Vamos digitar: `FACA "P PEGUE`

RETURN

Observe que o cursor se posiciona na linha abaixo, sem a costumeira interrogação, à espera de uma entrada via teclado. Digite qualquer caractere e, automaticamente, a interrogação volta a aparecer, sem que para isso seja necessário teclar **RETURN** .

A seguir, solicite: `MO :P`

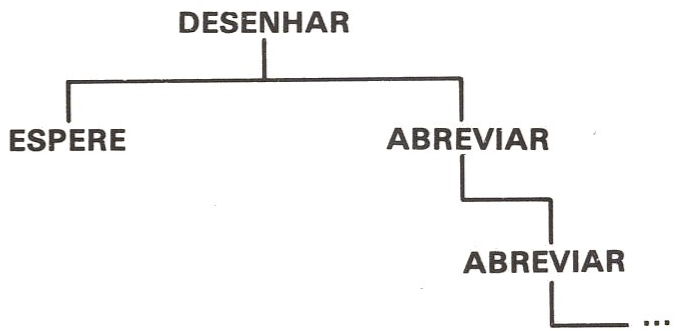
E obterá, como resposta, o caractere que você acabou de digitar.

Como usar o **PEGUE** em procedimentos?

1.
```
AP ESCOLHA
MO [QUE FIGURA VOCE QUER QUE EU DESENHE?]
MO []
MO [DIGITE 1 PARA QUADRADO.]
MO []
MO [DIGITE 2 PARA TRIANGULO.]
MO []
MO [DIGITE 3 PARA CIRCUNFERENCIA.]
MO []
FACA "R PEGUE
SE :R = 1 ENTAO QUA
SE :R = 2 ENTAO TRI
SE :R = 3 ENTAO CIR
FIM
```

Para que **ESCOLHA** seja executado, você deve ensinar os procedimentos **QUA** , **TRI** e **CIR** .

2. Podemos também criar uma forma própria de desenhar:



2.1.
```
AP DESENHAR
MO [VAMOS DESENHAR DE MODO MAIS SIMPLES]
MO []
MO [DIGITE D PARA DIREITA]
MO [DIGITE E PARA ESQUERDA]
MO [DIGITE F PARA FRENTE]
MO [DIGITE V PARA VOLTE]
MO []
MO [QUANDO SEU DESENHO ESTIVER PRONTO]
MO [DIGITE P PARA PARAR]
ESPERE 100
ABREVIAR
FIM
```

2.2.
```
AP ABREVIAR
FACA "OPCAO PEGUE
SE :OPCAO = "D ENTAO DI 15
SE :OPCAO = "E ENTAO ES 15
SE :OPCAO = "F ENTAO FR 10
SE :OPCAO = "V ENTAO VO 10
SE :OPCAO = "P ENTAO PARE
ABREVIAR
FIM
```

Ao chamarmos o procedimento **DESENHAR** , este mostrará as mensagens no vídeo e, a seguir, acionará o procedimento **ABREVIAR** , colocando a tartaruga à nossa disposição para receber as ordens de comando. Lembre-se ainda que **ESPERE :T** já foi definido em capítulos anteriores.

3. Vamos fazer um procedimento para adivinhar números?

```
AP SORTE
FACA "N 1 + SORTEIE 9
MO [ESCOLHA UM NUMERO ENTRE 1 E 9]
FACA "R PEGUE
TESTE :R = :N
SV MO [ACERTOU, PARABENS!]
SF ( MO [ERROU!! O NUMERO SORTEADO
FOI] :N )
MO [QUER TENTAR NOVAMENTE? (S/N)]
FACA "RESP PEGUE
TESTE :RESP = "S
SV SORTE
SF MO [ATE' LOGO]
FIM
```

4. Qual será o resultado do procedimento abaixo?

```
AP SURPRESA :N
SE :N <0 PARE
LIMPETEXTO
MO [DIGITE UM NUMERO DE 1 A 9]
FACA "X PEGUE
DI (:X*10) FR 20
SURPRESA :N-1
FIM
```

Não se esqueça de que o número dado como entrada deverá ser sempre de 1 a 9. A tartaruga girará à direita **(:X * 10)** graus e se locomoverá 20 passos, isto, durante 30 vezes. Observe o resultado do procedimento entrando com diferentes números.

SUGESTÃO

Elabore testes para serem respondidos por seus colegas, onde as respostas sejam **V** ou **F** , **S** ou **N** . Não se esqueça de colocar mensagens.

Por exemplo:

```
AP QUEST
MO [RESPONDA COM V OU F]
MO []
MO [O BRASIL ESTA' DIVIDIDO EM 4 REGIOES.]
TESTE PEGUE = "F
SV MO [VOCE ACERTOU!]
SF MO [O BRASIL POSSUI 5 REGIOES.]
MO []
MO [O AMAZONAS PERTENCE A REGIAO NORTE.]
TESTE PEGUE = "V
SV MO [ACERTOU!]
SF MO [A AFIRMACAO E' VERDADEIRA.]
FIM
```

II. **ENTRE (EN):** esta operação instrui o computador para que aguarde uma entrada qualquer através do teclado. A seguir pressione a tecla **RETURN** . Quando isto ocorre, a ordem **ENTRE** faz com que o que foi digitado seja armazenado em forma de lista.

Digite: FACA "A ENTRE **RETURN**

Observe que o cursor se posiciona na linha abaixo, aguardando uma entrada. Escreva uma palavra qualquer e novamente pressione **RETURN** . A seguir, digite:

```
MO :A
```

E obterá como resposta a lista digitada.

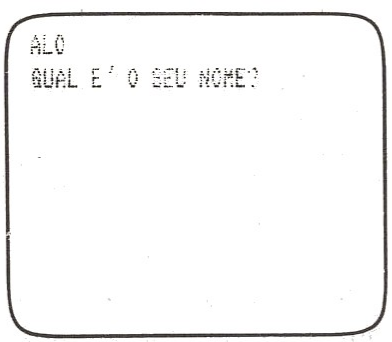
Aplicações em procedimentos:

```
1. AP ALO
   MO [QUAL E' O SEU NOME?]
   FACA "NOME ENTRE
   ( MO "OLA, :NOME )
   FIM
```

Digite **ALO** e acompanhe sua resolução.

1.1. **LINHA 1 — NÍVEL 1 ⟶ MO QUAL E O SEU NOME?]**

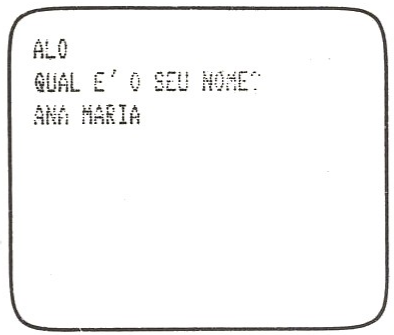
Mostre no vídeo a frase: **QUAL E O SEU NOME?**



1.2. **LINHA 2 — NÍVEL 1 ⟶ FACA "NOME ENTRE**

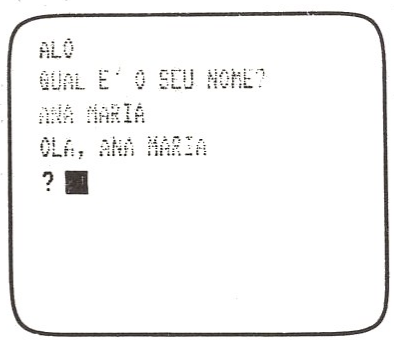
Crie a variável **:NOME** e atribua a ela, o que for digitado. Observe que o cursor estará piscando e

não aparecerá a interrogação (**?**). Isto significa que está esperando a entrada. (Vamos digitar **ANA MARIA** como entrada).



1.3. **LINHA 3 — NÍVEL 1 ⟶ (MO "OLA, :NOME)**

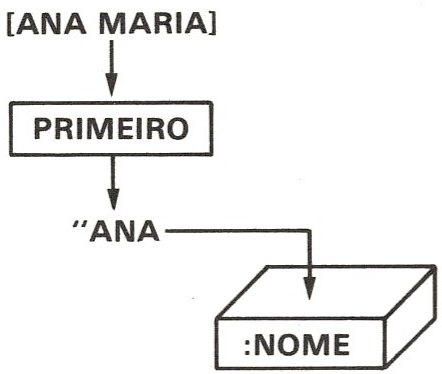
Mostre no vídeo, a sentença formada pela junção da palavra **"OLA** , com o conteúdo da variável **:NOME (ANA MARIA)** .



Às vezes pode ser interessante que a lista dada como entrada seja transformada em palavra. Assim:

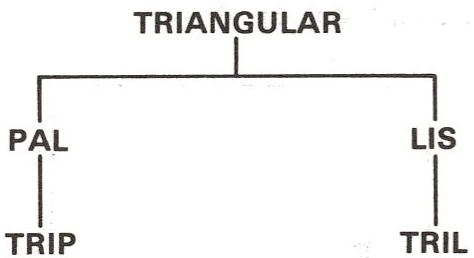
```
FACA "NOME PRIMEIRO ENTRE
```

Se digitarmos **ANA MARIA** , **ANA** é a palavra que passará a ser o conteúdo da variável **:NOME** . Observe:



2. A partir de agora, alguns procedimentos que você já realizou poderão ser melhorados.

Por exemplo, os procedimentos para triangular palavras ou listas poderão tornar-se subprocedimentos de:



2.1.
```
AP TRIANGULAR
MO [VOCE QUER TRIANGULAR UMA PALAVRA
OU UMA LISTA?]
FACA "R PRIMEIRO ENTRE
SE :R = "PALAVRA ENTAO PAL
SE :R = "LISTA ENTAO LIS
MO [QUER CONTINUAR? (S/N)]
FACA "R PEGUE
SE :R = "S TRIANGULAR SENAO MO [ATE'
BREVE] PARE
FIM
```


2.2.
```
AP PAL
MO [QUAL A PALAVRA QUE VOCE QUER
TRIANGULAR?]
FACA "P PRIMEIRO ENTRE
TRIP :P
FIM
```

2.3.
```
AP LIS
MO [QUAL A LISTA QUE VOCE QUER
TRIANGULAR?]
FACA "L ENTRE
TRIL :L
FIM
```

SUGESTÕES

a. Tente aperfeiçoar seus procedimentos para conjugar verbos, introduzindo as seguintes perguntas:

— Qual o verbo regular que deverá ser conjugado?

— Em que tempo você quer que o verbo escolhido seja conjugado?

— Quer continuar a conjugar verbos?

b. Há várias outras maneiras de se interagir com o computador. Utilizando as novas primitivas **PEGUE** e **ENTRE** , procure criar procedimentos em que o computador e o usuário mantenham um diálogo.

III. **EXECUTE:** como já foi visto, tanto as primitivas como os procedimentos ensinados ao computador são palavras, sendo assim, podem fazer parte de uma lista.

Por exemplo:

```
FACA "CONTEUDO [FR 20 DI 30 VO 15]
```

:CONTEUDO é uma lista de seis elementos.

Poderíamos também incluir na lista **:CONTEUDO** o título de qualquer procedimento já definido. Veja:

```
FACA "CONTEUDO [FR 20 DI 30 VO 15 QUA]
```

A lista de instruções que atribuímos à variável **:CONTEUDO** pode ser executada pelo LOGO através do comando **EXECUTE** .

Ordenando:

```
EXECUTE :CONTEUDO
```

O LOGO executará as ordens contidas na lista, ou seja, a tartaruga andará 20 passos para a frente, girará trinta graus à direita, voltará 15 passos e desenhará um quadrado que já deverá estar na memória do computador.

O comando **EXECUTE** pode ser utilizado em procedimentos, permitindo-nos alternar telas de texto e gráficas, com ordens dadas via teclado, sem sairmos do procedimento inicial. Assim:

1.
```
AP DESCOBERTAS
MO [QUE ORDENS VOCE DARIA PARA
DESENHAR]
MO [UM QUADRADO CUJOS LADOS MEDEM
50 PASSOS]
MO [GIRANDO PARA A DIREITA]
FACA "R ENTRE
TESTE :R = [REPITA 4 [FR 50 DI 90]]
SV EXECUTE :R
```



```
SF MO [TENTE NOVAMENTE] DESCOBERTAS
MO [AGORA VOU GIRAR A TARTARUGA]
ESPERE 100
DI 30
MO [VAMOS CONSTRUIR UM TELHADO? (S/N)]
TESTE PEGUE = "S
SV TELHADO
SF MO [QUE PENA]
FIM
```

2.
```
AP TELHADO
MO [DE AS ORDENS PARA A CONSTRUCAO DO]
MO [TRIANGULO QUE FORMARA' O TELHADO]
FACA "R ENTRE
TESTE :R = [REPITA 3 [FR 50 DI 120]]
SV EXECUTE :R
SF MO [PRESTE MAIS ATENCAO!] TELHADO
LIMPETEXTO
MO [PARABENS!]
MO [AGORA COMPLETE O DESENHO COMO
QUISER]
FIM
```

Nos computadores, existe uma espécie de memória especial para teclas, que permite que alguns caracteres sejam digitados durante a execução de programas. Esta espécie de memória recebe o nome de **BUFFER** . O buffer armazena os caracteres na ordem em que são digitados.

Quando trabalhamos com o LOGO, não podemos digitar nada enquanto um procedimento seja executado, a não ser que o próprio procedimento o exija, pois estes caracteres são perdidos porque não temos como consultar o buffer para ver se possui algum caractere pendente.

Porém, existe uma operação lógica primitiva que nos possibilita fazer essa verificação. É ela:

IV. **CP?:** esta operação nos responde **VERD** caso exista algum caractere pendente no buffer e **FALSO** se não houver. É normalmente usada associada ao **PEGUE** .

Vamos a um exemplo:

TECLAR
|
TECLAS

1.
```
AP TECLAR
MO [VAMOS TREINAR DATILOGRAFIA. NAO
DEIXE]
MO [QUE SUA PALAVRA SEJA DIVIDIDA.
PARA ]
MO [INTERROMPER, PRESSIONE CTRL + G]
ESPERE 500
TECLAS
FIM
```

2.
```
AP TECLAS
TESTE CP?
SV MOSTRAR PEGUE
SF MO "
ESPERE 30
TECLAS
FIM
```


CAPÍTULO 7

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
SAIDA	**SA**	Faz com que o programa "saia" com determinado valor, permitindo-nos criar operações.

CRIANDO OPERAÇÕES

O vocabulário LOGO é composto por primitivas que, como vimos na introdução deste livro, dividem-se em dois tipos: Comandos e Operações.

Criamos, até então, apenas comandos que podem ou não precisar de dados de entrada, de acordo com nossas necessidades particulares.

Em nossos procedimentos (até agora comandos) por vezes utilizamos operações pertencentes ao vocabulário LOGO (+, /, *, **SN** , **PALAVRA** ,...), porém não criamos nada desta natureza. Para isso, necessitaremos de um novo comando: **SAIDA** .

Veja a diferença entre os procedimentos abaixo:

```
AP SOMA :A :B
FACA "S :A + :B
MO :S
FIM
```

```
AP SOMAR :A :B
FACA "S :A + :B
SAIDA :S
FIM
```

Digite:

```
SOMA 5 8
```

Obteremos →

```
13
```

Digite:

```
SOMAR 5 8
```

Obteremos →

```
IGUAL:13
```

A mensagem obtida após a resolução de **SOMAR (IGUAL:)** significa que a "operação" **SOMAR** foi realizada, porém não especificamos o que deverá ser feito com o resultado.

Digite então:

```
MO SOMAR 5 8
```

Obteremos →

```
13
```



```
MO SOMAR 5 8
```

Operação

Comando (ação)

Ao passo que digitando:

```
MO SOMA 5 8
```

receberemos a seguinte mensagem:

```
SOMA NAO TEM SAIDA
```

Ou seja, **SOMA** é um comando e não uma operação; portanto, não determina um "resultado" (saída).

Podemos, então, transformar alguns comandos que criamos anteriormente em operações muito interessantes:

OBSERVE!

1.


```
AP CONTAGEM :E
TESTE LISTA? :E
SV FACA "N CONTEL :E 0
SF FACA "N CONTEP :E 0
( MO :N "POSSUI :N "ELEMENTOS )
FIM
```

Percebemos que o procedimento **CONTAGEM** utiliza-se de: **CONTEL** e **CONTEP** , que vimos em capítulos anteriores. Vamos recordá-los:

```
AP CONTEL :E :N
SE :E = [] MO :N PARE
CONTEL SP :E :N+1
FIM

AP CONTEP :E :N
SE :E = " MO :N PARE
CONTEP SP :E :E+1
FIM
```

Ao digitarmos **CONTAGEM "CASA** , obteremos como mensagem:

```
CONTEP NAO TEM SAIDA NA LINHA
SF FACA "N CONTEP :E 0 NO
NIVEL 1 DE CONTAGEM
```

Isto é, **CONTEP** não é operação e sim comando. Portanto, vamos transformá-lo, fazendo o mesmo com **CONTEL** .

Comando

```
AP CONTEP :E :N
SE :E = " MO :N PARE
CONTEP SP :E :N+1
FIM

AP CONTEL :E :N
SE :E = [] MO :N PARE
CONTEL SP :E :N+1
FIM
```


Operação

```
AP CONTEP :E :N
SE :E = " SAIDA :N PARE
SAIDA CONTEP SP :E :N+1
FIM

AP CONTEL :E :N
SE :E = [] SAIDA :N PARE
SAIDA CONTEL SP :E :N+1
FIM
```

Após estas modificações, ao digitarmos:

```
CONTAGEM "CASA
```

Obteremos ⟶

```
CASA POSSUI 4 ELEMENTOS
```

2. Vamos criar uma operação que retire um determinado elemento de uma lista, pedido pela sua posição:

```
AP RETIRE :N :L
SE :N = 1 SAIDA PRIMEIRO :L PARE
SAIDA RETIRE :N-1 SP :L
FIM
```

Digitando:

```
RETIRE 3 [GATO PATO RATO
MACACO]
```

Obteremos ⟶

```
IGUAL :RATO
```

```
MO RETIRE 2 [GATO PATO]
```

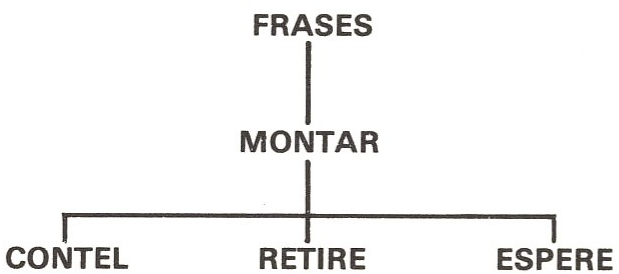
Obteremos ⟶

```
PATO
```

```
MO PALAVRA "CARRA RETIRE 3
[GATO SAPO PATO]
```
Obteremos → CARRAPATO

```
RETIRE 2 "SOLAR
```
Obteremos → IGUAL:0

Utilizando as operações **CONTEL** e **RETIRE** e o comando **ESPERE** (todos ensinados por nós), construiremos um procedimento que crie frases juntando os elementos aleatoriamente.



2.1.
```
AP FRASES
MO [DIGITE ALGUNS NOMES DE ANIMAIS
(NO MASCULINO)]
FACA "ANIMAL ENTRE
MO [DIGITE ALGUMAS CORES (NO
MASCULINO)]
FACA "COR ENTRE
MO [DIGITE ALGUNS VERBOS (NO
INFINITIVO)]
FACA "VERBO ENTRE
MO [DIGITE ALGUNS SUBSTANTIVOS
MASCULINOS]
MO [QUE INDIQUEM LUGAR]
FACA "SUBST ENTRE
MONTAR
FIM
```


2.2.
```
AP MONTAR
FACA "NA 1 + SORTEIE CONTEL :ANIMAL 0
FACA "NC 1 + SORTEIE CONTEL :COR 0
FACA "NV 1 + SORTEIE CONTEL :VERBO 0
FACA "NS 1 + SORTEIE CONTEL :SUBST 0
( MO "O RETIRE :NA :ANIMAL RETIRE :NC
:COR [GOSTA DE] RETIRE :NV :VERBO "NO
RETIRE :NS :SUBST )
ESPERE 30
MONTAR
FIM
```

3.
```
AP INCLUIR :L
( MO [QUE ELEMENTO VOCE QUER INCLUIR
NA LISTA] :L "? )
FACA "X ENTRE
TESTE :X = []
SV SAIDA :L
SF FACA "L SN :L PRIMEIRO :X
SAIDA INCLUIR :L
FIM
```

Digite:

```
INCLUIR [AZUL AMARELO]
```
RETURN

Digite →
```
QUE ELEMENTO VOCE QUER INCLUIR
NA LISTA AZUL AMARELO?
VERMELHO
QUE ELEMENTO VOCE QUER INCLUIR
NA LISTA AZUL AMARELO VERMELHO?
```

Tecle → RETURN

Obteremos → AZUL AMARELO VERMELHO

4.
```
AP POTENCIA :N :X
SE :N = 0 ENTAO SAIDA 1 PARE
FACA "EXP :N
SAIDA POT :N :X
FIM

AP POT :N :X
SE :X = 1 ENTAO SAIDA :EXP PARE
FACA "EXP :EXP * :N
SAIDA POT :N :X-1
FIM
```

Digite:

```
MO POTENCIA 2 3
```
Obteremos → 8 $(= 2^3)$

```
MO POTENCIA 0 100
```
Obteremos → 1 $(= 0^{100})$

```
MO POTENCIA -4 3
```
Obteremos → -64 $(= (-4)^3)$

```
MO  POTENCIA 2 3
+  POTENCIA 3 2
```
Obteremos → 17 $(= 2^3 + 3^2)$

CAPÍTULO 8

RESUMO DE PRIMITIVAS		
PRIMITIVA	ABREV.	FUNÇÃO
ASC a		Responde com o valor numérico (Tabela **ASCII**) correspondente ao caractere **a**.
CARAC n		Responde com o caractere correspondente ao número **n** (Tabela **ASCII**).
CURSOR x y		Posiciona o cursor na coluna **x** e na linha **y** especificada.

NOVAS PRIMITIVAS

Quando digitamos qualquer informação através do teclado, utilizamos letras, números ou outros caracteres existentes. No entanto, para que essa informação seja processada, é necessário convertê-la a um código numérico que associa cada caractere a um número. Este código é conhecido como **American Standard Code for Information Interchange** (**ASCII**).

I. **ASC:** é uma primitiva do LOGO; retorna com o valor numérico da tabela **ASCII** , correspondente ao caractere digitado como entrada. Assim, se solicitarmos:

```
      ASC "A
Obteremos ──► 65

      ASC "C
Obteremos ──► 67
```

No apêndice ao final deste livro, você encontrará a tabela **ASCII** por nós utilizada.

II. **CARAC:** esta primitiva, ao contrário da anterior, necessita de um número como argumento e seu efeito é transformar o número digitado no caractere equivalente pela tabela **ASCII** .
Digitando:

```
      CARAC 65
Obteremos ──► A

      CARAC 68
Obteremos ──► D
```


Vamos ensinar, agora, um procedimento que crie um código e outro que decodifique:

1.
```
AP COD :P
SE :P = "" ENTAO MO [] PARE
MOSTRAR PALAVRA ASC PRIMEIRO :P CARAC 32
COD SP :P
FIM
```

Digite:

```
      COD "BOLA
Obteremos ──► 66 79 76 65
```

2.
```
AP DECOD :L
SE :L = [] PARE
MOSTRAR CARAC PRIMEIRO :L
DECOD SP :L
FIM
```

Digite:

```
      DECOD [66 79 76 65]
Obteremos ──► BOLA
```

No primeiro procedimento, a palavra dada como entrada tem cada caractere convertido ao seu valor numérico correspondente, através da tabela **ASCII** , assim:

B = 66
O = 79
L = 76
A = 65

Já no segundo procedimento, ao darmos uma lista de números como entrada, obteremos, através da primitiva **CARAC** , o caractere correspondente a cada um dos valores da lista.

EXPERIMENTE!!!

```
REPITA 5 [DECOD [66 32 79 32 66
32 79 32 7 13]
```

Observação:

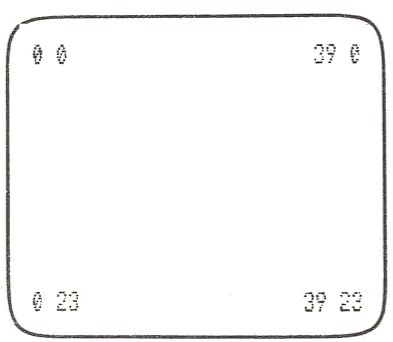
- O **CARAC 32** corresponde ao espaço em branco.
- O **CARAC 7** corresponde ao som emitido ao final de cada linha.
- O **CARAC 13** equivale a pressionar a tecla **RETURN**

Já vimos que o LOGO trabalha em dois tipos de tela simultaneamente: a tela gráfica e a de textos.

Se desejarmos trabalhar somente na tela gráfica, pressionamos as teclas **CTRL F** ; por outro lado, se quisermos somente a tela de textos pressionamos **CTRL T** .

Porém, se pressionarmos **CTRL S** , obteremos a combinação das duas, encontrando as últimas quatro linhas reservadas para texto.

A tartaruga é utilizada na tela gráfica e o cursor na tela de textos. Esta última possui 24 linhas por 40 colunas a partir do canto superior esquerdo. Observe:



III. **CURSOR:** esta primitiva nos permite posicionar o cursor em qualquer posição que determinarmos na tela de texto. O cursor necessita de dois dados numéricos de entrada, sendo que o primeiro corresponde à coluna e o segundo à linha.

Vamos utilizar esta primitiva juntamente com **ASC** e **CARAC** em alguns procedimentos:

1.
```
AP SURPRESA :X
LIMPETEXTO
CURSOR SORTEIE 30 SORTEIE 20
MO :X
MO CARAC 7
ESPERE 20
SURPRESA :X
FIM
```

2.
```
AP PISCAR :P
CURSOR 12 10
MO :P
ESPERE 5
CURSOR 12 10
MO []
ESPERE 5
PISCAR :P
FIM
```

3.
```
AP ABC :N
TESTE SETODOS :N > 65 :N < 90
SV MO CARAC :N
SF MO [DIGITE OUTRO NUMERO] FACA
"N PRIMEIRO ENTRE
ABC :N
FIM
```

4.
```
AP ESCADA :COL :LIN
TESTE SEUM :COL > 30 :LIN > 20
SV PARE
SF CURSOR :COL :LIN ( MO "DEGRAU
CARAC 7 )
ESCADA :COL+2 :LIN+1
FIM
```


CAPÍTULO 9

VOLTANDO A DESENHAR

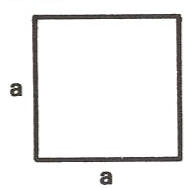
I. QUADRILÁTEROS

O termo quadrilátero refere-se a todas as figuras fechadas simples que possuam quatro lados. Estes podem ser regulares (lados de igual tamanho e ângulos congruentes, no caso, quadrado) ou irregulares (retângulo, paralelogramo e trapézio).

Estudaremos cada um deles separadamente. Comecemos pelo caso mais simples.

1. QUADRADO

Quadrado (como já vimos no livro ***LOGO: O Primeiro Passo***) é uma figura que possui lados iguais e ângulos congruentes (iguais a 90 graus).

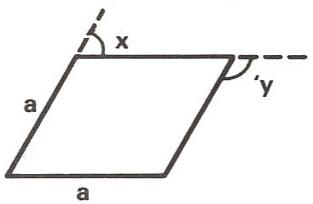


Chamaremos de "a" o tamanho do lado do nosso quadrado. Dessa forma, podemos criar um procedimento **QUA** , assim:

```
AP QUA :A
REPITA 4 [ FR :A DI 90 ]
FIM
```

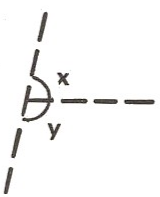
2. LOSANGO

Se inclinarmos nosso quadrado, obteremos um losango. Observe:



Portanto, as únicas coisas que diferem um quadrado de um losango são os ângulos. Os ângulos de um quadrado medem sempre 90 graus, porém no losango não (ver figura acima). Chamamos de **x** e **y** os ângulos assim determinados.

Porém, **x** e **y** são suplementares, ou seja, **x + y = 180°**



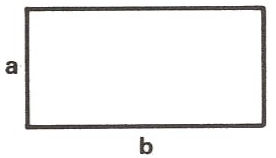
Daí, concluímos que y = 180° − x.

Como podemos obter **y** em função de **x**, o procedimento **LOS** que criaremos necessita de apenas duas variáveis, o tamanho dos lados (**a**) e o ângulo (**x**):

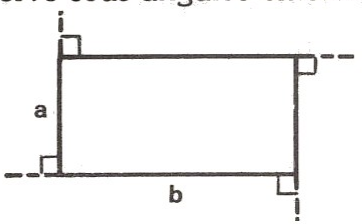
```
AP LOS :A :X
REPITA 2 [FR :A DI :X FR :A DE 180-:X]
FIM
```


3. RETÂNGULO

Para criarmos um procedimento que desenhe retângulos, devemos observar como desenhamos tal figura. Veja:



Através do desenho, chamaremos de **a** o lado à altura do retângulo e **b** o lado referente à base, sendo ambos iguais dois a dois. Agora, observe seus ângulos externos:



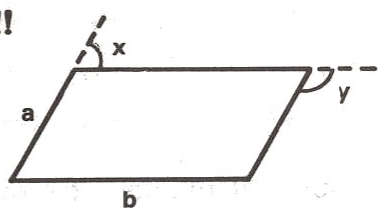
Isso mesmo, todos são iguais a 90°. Portanto, as únicas variáveis são a altura e a base. Assim, podemos criar o procedimento **RET** da seguinte forma:

```
AP RET :A :B
REPITA 2 [ FR :A DI 90 FR :B DI 90 ]
FIM
```

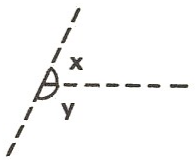
4. PARALELOGRAMO

Se tivermos um retângulo de altura **a** e base **b** e o inclinarmos, sem tirá-lo do lugar, obteremos um paralelogramo.

OBSERVE!!!



A única coisa que difere um paralelogramo de um retângulo são os ângulos. Na figura, chamamos o primeiro de **x** e o segundo de **y**. Porém, se eliminarmos os deslocamentos e nos fixarmos só nos ângulos teremos:



então, **x + y = 180°**
y = 180° - x

Como podemos obter **y** em função de **x**, nosso problema se reduz a três variáveis: a altura **a**, a base **b** e o ângulo **x**. Vamos então criar o procedimento **PAR** da seguinte forma:

```
AP PAR :A :B :X
REPITA 2 [FR :A DI :X FR :B DI 180-:X]
FIM
```

Porém, é curioso notarmos que todos estes quadriláteros são, em parte, comuns, e como o paralelogramo é o caso mais específico, podemos obter todos os outros através do procedimento **PAR** , atribuindo os dados de maneira adequada.

EXPERIMENTE!!!

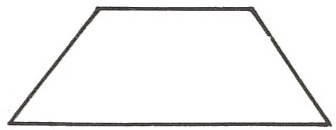
Descubra que tipo de quadrilátero resultará das ordens abaixo:

1. `PAR 50 50 90`
2. `PAR 50 50 45`
3. `PAR 60 80 90`
4. `PAR 60 80 120`
5. `PAR 60 80 135`
6. `PAR 80 60 30`

Lembre-se de que o último dado é o que se refere ao ângulo.

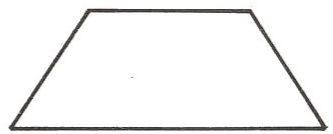
5. TRAPÉZIO

Veja o desenho de um trapézio:



Observe que suas bases são paralelas e que é impossível utilizarmos o comando **REPITA** para sua execução.

Estudaremos um caso de trapézio em particular: o trapézio isósceles, que está representado na figura abaixo:

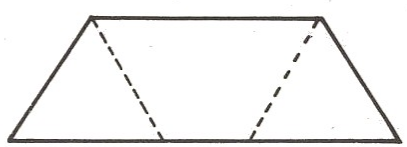


OBSERVE!

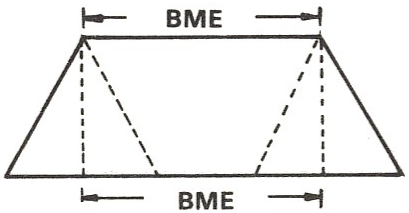
A soma dos ângulos externos deverá ser sempre 360°.

No caso específico deste trapézio, cujos ângulos externos medem respectivamente 60° e 120°, a base maior corresponde à soma da base menor com o lado. Vejamos por quê:

Vamos desenhar novamente o trapézio, completando a partir de seus lados dois triângulos equiláteros:

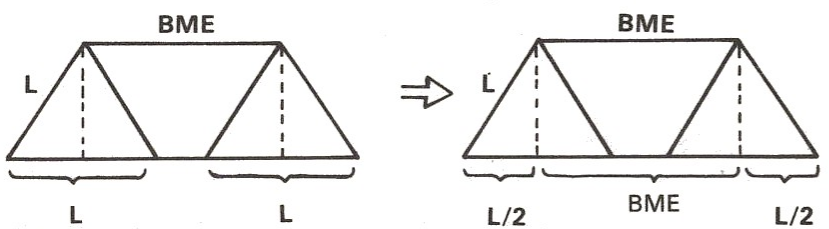


A seguir, tracemos a altura de cada um dos dois triângulos observando o retângulo assim formado:



Resta-nos saber quanto temos que somar à base menor (**BME**) para chegarmos à base maior (**BMA**).

Como a altura divide a base do triângulo equilátero em duas partes iguais e essa mesma base possui tamanho **L**, cada uma das partes mede **L/2**.



Portanto, **BMA = BME + L**

Então, para construirmos um procedimento variável para a execução de trapézios isósceles (de qualquer tamanho), cujos ângulos externos sejam 60° e 120°, necessitaremos de duas variáveis:

BME ⟶ correspondente à base menor
L ⟶ correspondente ao lado.

Assim, nosso procedimento **TRAP** ficará:

```
AP TRAP :BME :L
FR :BME DI 60
FR :L DI 120
FR :BME + :L DI 120
FR :L DI 60
FIM
```

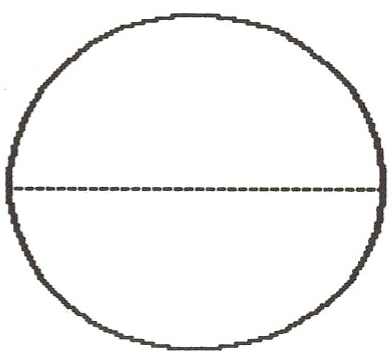
Faremos, agora, um aprofundamento da parte gráfica, para que você possa ter uma visão um pouco mais abrangente do que o LOGO é capaz de fazer.

II. CIRCUNFERÊNCIAS E ARCOS

1. CIRCUNFERÊNCIAS

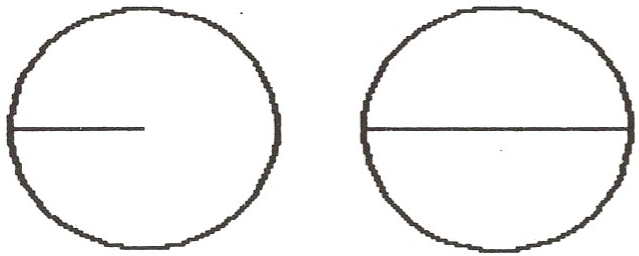
Temos utilizado o polígono de 36 lados para representar uma circunferência. Nesse mesmo polígono, podemos variar o tamanho de cada uma das frentes aumentando ou diminuindo seu perímetro (soma dos lados).

Porém, é muito difícil determinarmos o quanto a tartaruga deve andar para "atravessar" a circunferência, passando exatamente pelo centro:



OBSERVE!

O caminho a ser traçado é o **diâmetro** da circunferência, ou melhor, duas vezes o tamanho do **raio**:



Podemos construir um procedimento **CIRC** que admita como argumento o valor do **raio** da circunferência. Para tanto, devemos rever alguns conceitos.

Ao utilizarmos o polígono de 36 lados no lugar da circunferência, estamos supondo que seus perímetros sejam iguais ou muito aproximados.

Supondo o perímetro do polígono de 36 lados igual ao da circunferência teremos:

```
REPITA 36 [ FRENTE :T DI 10 ]
```

36 * **:T** = Perímetro do polígono de 36 lados (P_{36}).

e **2 * π * :R** = Perímetro da circunferência (P_C).

Raio

3.14

Portanto,

$P_{36} = 36 * :T$

$P_C = 2 * 3.14 * :R$ ou $P_C = 6.28 * :R$

Pela suposição acima temos: $P_{36} = P_C$

então, podemos dizer que: $\underbrace{36 * :T}_{P_{36}} = \underbrace{6.28 * :R}_{P_C}$

assim, **:T = 6.28 * :R / 36**

Basta agora substituir **:T** por **6.28 * :R / 36** na instrução:

```
REPITA 36 [ FR :T DI 10 ]
```

teremos:

```
REPITA 36 [ FR 6.28 * :R /36 DI 10 ]
```

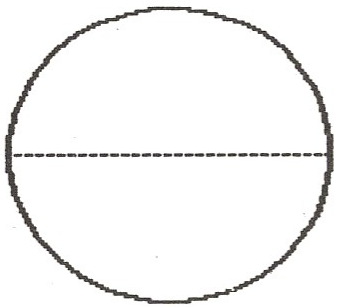
EXPERIMENTE!!!

Vamos, então, ensinar o procedimento **CIRC** :

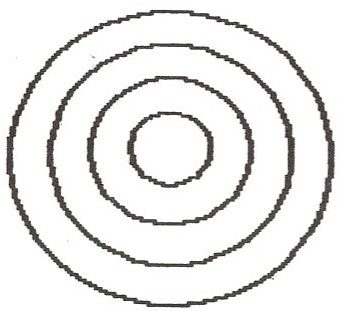
```
AP CIRC :R
REPITA 36 [ FR 6.28 * :R / 36 DI 10 ]
FIM
```

Digite:

Obteremos → `CIRC 2?`



Tente fazer o seguinte desenho usando o procedimento **CIRC**:



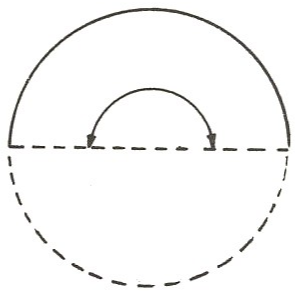
2. ARCOS

Arcos de circunferência podem ser muito úteis na elaboração de projetos. Observe:

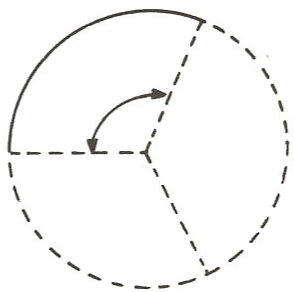
→ 4 arcos de 1/2 de circunferência

→ 2 arcos de 1/4 de circunferência

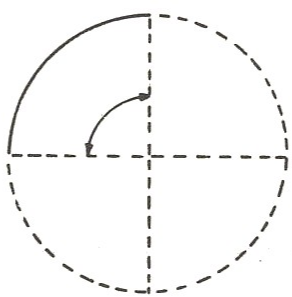
Vamos recordar:



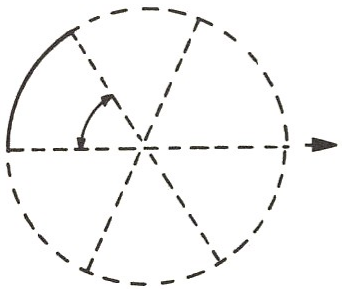
Arco de 1/2 de circunferência. O giro total da tartaruga para sua execução é 180 graus, ou seja, $\frac{1}{2}$ 360.



Arco de 1/3 de circunferência. O giro total da tartaruga para sua execução é 120 graus, ou seja, $\frac{1}{3}$ 360.



Arco de 1/4 de circunferência. O giro total da tartaruga para sua execução é 90 graus, ou seja, $\frac{1}{4}$ 360.



Arco de 1/6 de circunferência. O giro total da tartaruga para sua execução é 60 graus, ou seja, $\frac{1}{6}$ 360.

OBSERVE!

Em todos os arcos, o que variou foi somente a "quantidade" de circunferência que foi desenhada (1/2, 1/3, 1/4,...).

Podemos, então, criar um procedimento **ARCO** que admita como variável a "quantidade" de circunferência a ser desenhada e o raio da mesma:

```
AP ARCO :Q :R
REPITA :Q * 36 [ FR 6.28 * :R /36 DI 10 ]
FIM
```

Digite:

1. `ARCO 1/4 20`

Obteremos →

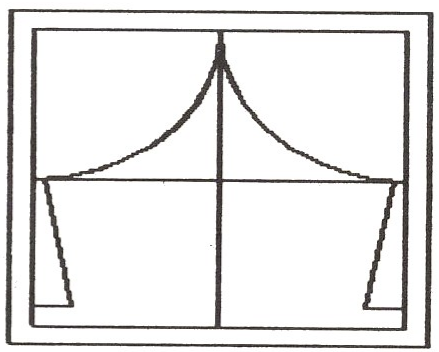
ou seja, desenhamos um arco de 1/4 de uma circunferência de raio 20.

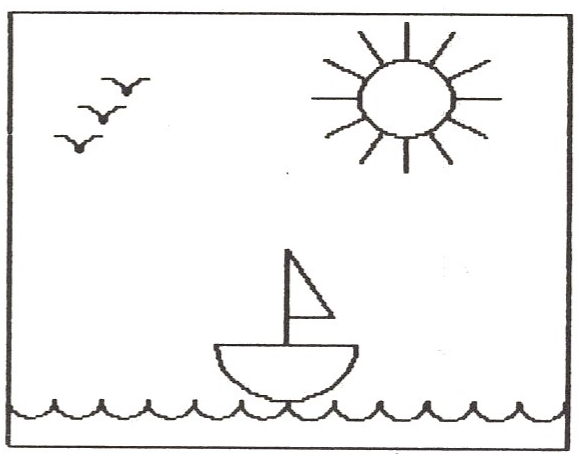
2. `ARCO 1/2 30`

Obteremos →

SUGESTÕES

Utilizando o procedimento **ARCO** , tente fazer estes desenhos:

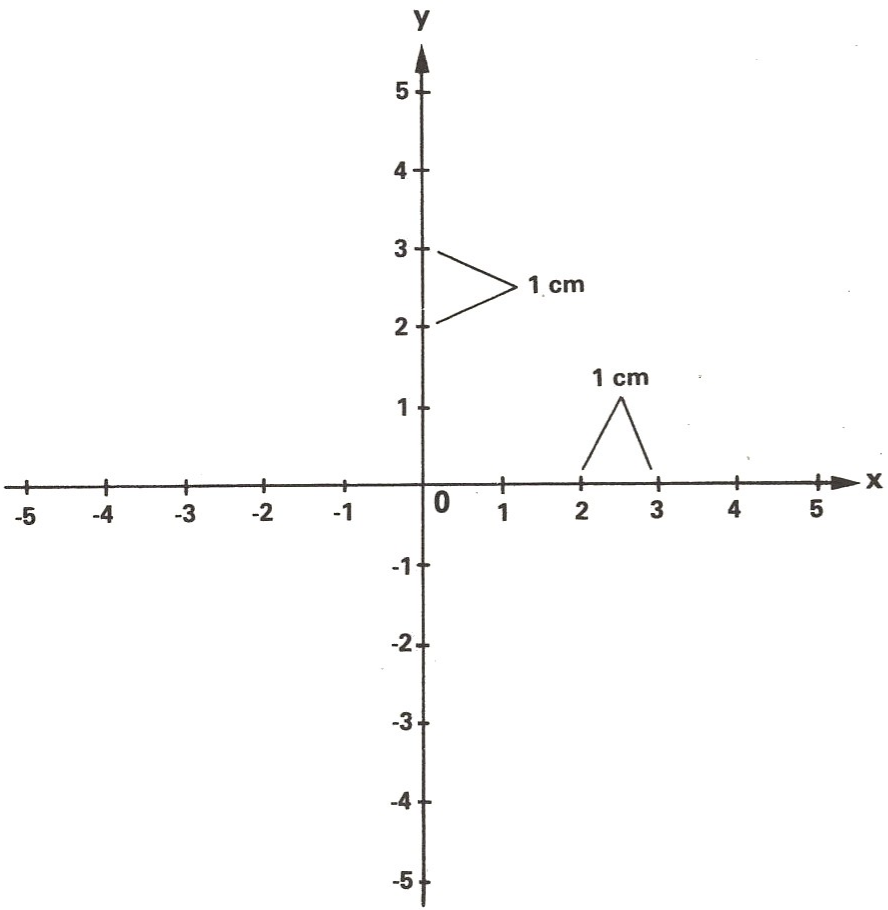




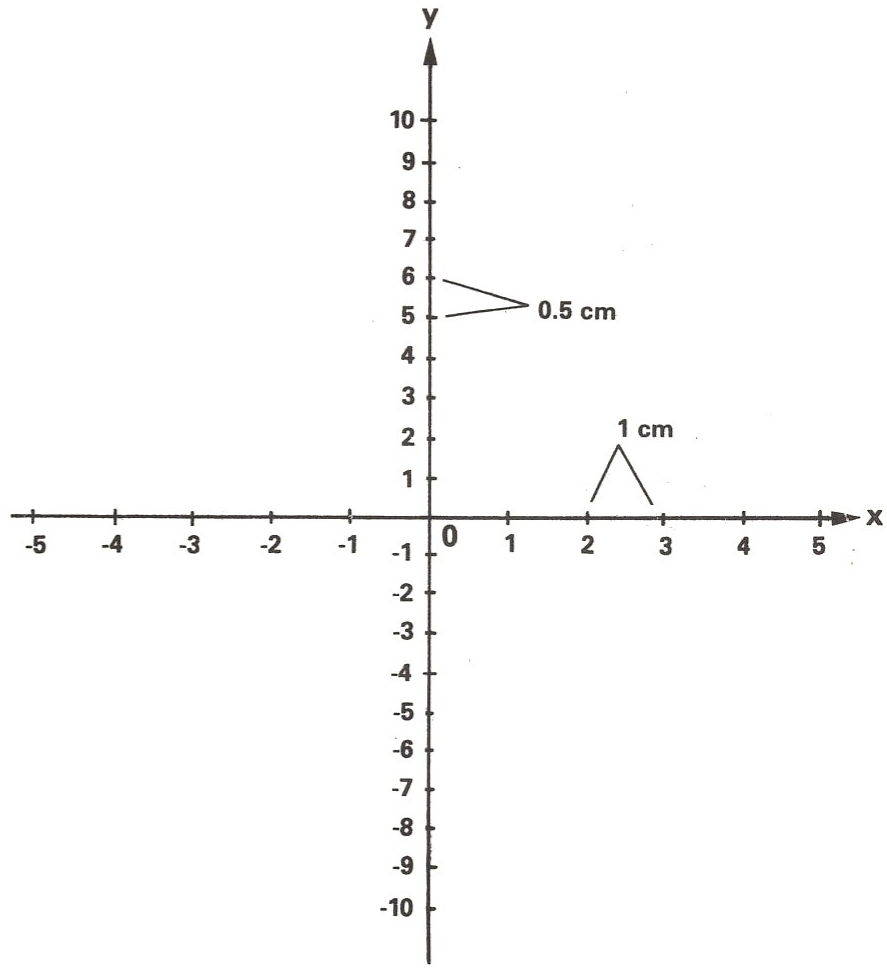
III. MUDANÇA DE ESCALA

Para a construção de gráficos, normalmente determinamos qual a escala em que serão desenhados, isto é, a distância entre cada um dos pontos marcados nos eixos, podendo ser diferente em **X** e **Y**. Observe:

a. Escala horizontal (**X**) = escala vertical (**Y**) = 1 cm:



b. Escala horizontal (**X**) = 1 cm
Escala vertical (**Y**) = 0,5 cm



Perceba que quanto menor a escala, mais pontos cabem no eixo, tendo este um tamanho fixo, isto é:

Num eixo de 20 cm cabem 20 pontos com distâncias de 1 cm, 40 pontos com distâncias de 0,5 cm, 10 pontos com distâncias de 2 cm e assim por diante.

No LOGO, é possível mudar somente a escala vertical, ou seja, a distância entre os pontos do eixo **Y**, através de uma de suas primitivas:

.VERTICAL: Muda a escala vertical utilizada na tela gráfica. Normalmente, seu valor é 0,8. Alterando este valor, também será alterada a quantidade de pontos que caberão no eixo Y, sendo maior se **.VERTICAL** for menor do que 0,8 e menor caso contrário, sem alterar a coordenada **X**.

Observe!

VALOR DA ESCALA VERTICAL	QUANTIDADE DE PONTOS
0,4	480
0,8	240
1,6	120

e assim por diante.

Caso você queira descobrir quantos pontos cabem no eixo Y ao alterar a escala vertical, basta lembrar que:

0,8 * 240 = valor determinado à vertical * quantidade de pontos. Exemplo:

quantos pontos cabem na vertical se alterarmos a escala para 1?

$$0{,}8 * 240 = 1 * Q$$

portanto:

$$Q = \frac{0{,}8 * 240}{1} = 192$$

Assim, com **.VERTICAL** valendo 1, posso marcar 192 pontos no meu eixo vertical.

Digite:

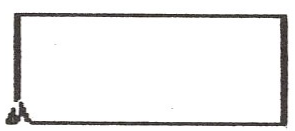
1.
```
AP QUA :T
REPITA 4 [ FR :T DI 90 ]
FIM
```

2.
```
AP CIRC :R
REPITA 36 [ FR 6.28 * :R / 36 DI 10 ]
FIM
```

Após retornar ao modo gráfico, digite:

a. `.VERTICAL 0.4 QUA 90`

Obteremos →

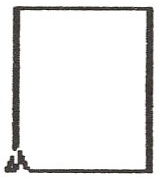


```
CIRC 40
```

Obteremos →

b. .VERTICAL 1.2 QUA 50

Obteremos →



CIRC 30

Obteremos →

Para continuar desenhando na escala normal, digite **.VERTICAL 0.8** .

A primitiva **.VERTICAL** pode ter uma variável como argumento desde que esta seja numérica, bem como pertencer a procedimentos:

```
AP GLOBO :V
SE :V < 0.1 ENTAO .VERTICAL 0.8 PARE
.VERTICAL :V
CIRC 30
GLOBO :V-0.1
FIM
```

Digite **GLOBO 1.6** e observe o resultado no vídeo.

Crie um procedimento que escreva no vídeo a palavra **BRASIL** , desenhando letra por letra.

OBSERVE!

Alterando **.VERTICAL** antes de sua execução, obteremos resultados bastante interessantes:

1. .VERTICAL 0.5 BRASIL

2. .VERTICAL 0.8 BRASIL

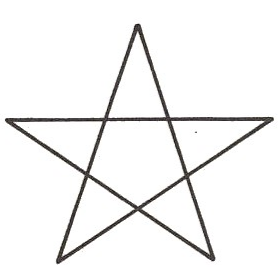
3. .VERTICAL 1.2 BRASIL

SUGESTÃO

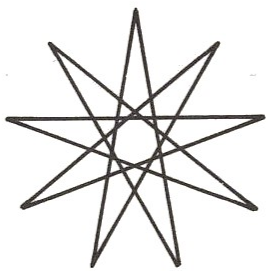
Tente com outros valores e observe o resultado.

IV. DESENHANDO ESTRELAS

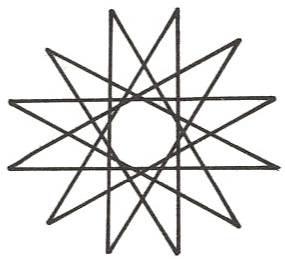
Um problema muito curioso, é como desenhar estrelas de diversos números de pontas, por exemplo, 5, 9, 12, ...



ESTRELA DE 5 PONTAS



ESTRELA DE 9 PONTAS



ESTRELA DE 12 PONTAS

Para desenhá-las, deveríamos ordenar:

```
REPITA :N [ FR :L DI :A ]
```

onde **:N** é o número de pontas da estrela
:L o tamanho que ela deverá ter
:A o ângulo determinado para sua construção.

Porém, o problema é: "Qual o valor do ângulo?"

Podemos perceber que **:N * :A** é um múltiplo de 360, pois para que a tartaruga retorne à posição de origem (onde iniciou o desenho), ela deve girar um número inteiro de vezes 360 graus.

Vamos tomar um exemplo real:

```
REPITA 9 [ FR 100 DI 160 ]
```

Pela instrução acima, podemos estabelecer a seguinte igualdade:

160 * 9 = 360 * X

onde **X** é o menor número natural que multiplicado por 360 representa o giro total da tartaruga para completar a figura.

Resolvendo a expressão acima temos:

1440 = 360 * X

portanto **X = 1440 / 360**, ou seja, **X = 4**

Isto significa que a tartaruga gira 4 vezes 360 graus para completar a figura, isto é, o giro total da tartaruga é de 1440 graus.

Porém, o que precisamos saber é:

1. Quantas pontas possui a estrela cujo ângulo é 144 graus? ou
2. Qual o ângulo necessário para desenhar uma estrela de 12 pontas?

Responderemos a estas perguntas uma de cada vez.

Primeiramente nos fixaremos em responder à pergunta nº 1.

Observe que determinamos o ângulo para tentarmos descobrir o número de pontas.

Retornemos ao exemplo, que desenha a estrela de nove pontas e notemos que 1440 (9*360) é o menor número múltiplo simultaneamente de 160 e 360, isto é, 1440 é o MMC (mínimo múltiplo comum) entre 160 e 360.

Por sua vez: **N * 160 = 1440**
portanto **N = 1440 / 160**
isto é **N = 9 (número de pontas da estrela)**

Vamos então responder à primeira pergunta — quantas pontas possui a estrela cujo ângulo é 144 —, para depois ensinarmos ao computador como desenhá-la.

1. Calcular o MMC entre 144 e 360

144	**360**	**2**
72	**180**	**2**
36	**90**	**2**
18	**45**	**2**
9	**45**	**3**
3	**15**	**3**
1	**5**	**5**
1	**1**	

$2^4 * 3^2 * 5 = 720$ ⟶ MMC entre **144** e **360**

2. O número de pontas é, então, o MMC entre o ângulo (144) e 360 dividido pelo ângulo (144), isto é:

   N = (MMC A 360)/A N = 720 / 144 N = 5

A estrela cujo ângulo é 144 possui 5 pontas.

Portanto, para desenhá-la, basta ordenar:

```
REPITA 5 [ FR 100 DI 144 ]
```


Passemos a criar os procedimentos necessários para a construção de estrelas a partir de um ângulo determinado:

ESTRELA
|
MMC
|
CALCULAR

```
AP ESTRELA1 :T :A
FACA "N (MMC 360 :A) / :A
REPITA :N [ FR :T DI :A ]
( MO [ESTA ESTRELA TEM] :N [PONTAS E
SEUS ANGULOS MEDEM ] :A "GRAUS )
FIM
```

ESTRELA1 necessita da operação **MMC** que criaremos a seguir.

Para entendê-la melhor, note que dado o ângulo, se o multiplicarmos por números naturais a partir do 1, um de cada vez, e dividirmos por 360, quando o resto for igual a zero, o resultado da multiplicação será o valor do **MMC** entre o ângulo e 360, ou seja, o giro total da tartaruga para a construção da respectiva estrela. Veja:

135 * 1 = 135 O resto de 135/360 = 135
135 * 2 = 270 O resto de 270/360 = 270
135 * 3 = 405 O resto de 405/360 = 45
135 * 4 = 540 O resto de 540/360 = 180
135 * 5 = 775 O resto de 775/360 = 55
135 * 6 = 810 O resto de 810/360 = 90
135 * 7 = 945 O resto de 945/360 = 225
135 * 8 = 1080 O resto de 1080/360 = 0 ⇒ 1080 é o MMC entre 135 e 360. Comprove:

135	360	2
135	180	2
135	90	2
135	45	3
45	15	3
15	5	3
5	5	5
1	1	

$2^3 * 3^3 * 5 = 1080$

Este método para nós é muito trabalhoso, o que não ocorre com o computador, pois ele faz contas com muita facilidade.

Vamos ensinar a operação MMC por este caminho:

1.
```
AP MMC :A :B
FACA "N 1
SAIDA CALCULAR :N :A :B
FIM
```

2.
```
AP CALCULAR :N :A :B
FACA "PRODUTO :N * :A
TESTE RESTO :PRODUTO :B = 0
SV SAIDA :PRODUTO
SF SAIDA CALCULAR :N+1 :A :B
FIM
```

Experimente digitar:

1. ESTRELA1 100 144
2. ESTRELA1 100 150
3. ESTRELA1 100 170
4. ESTRELA1 100 234

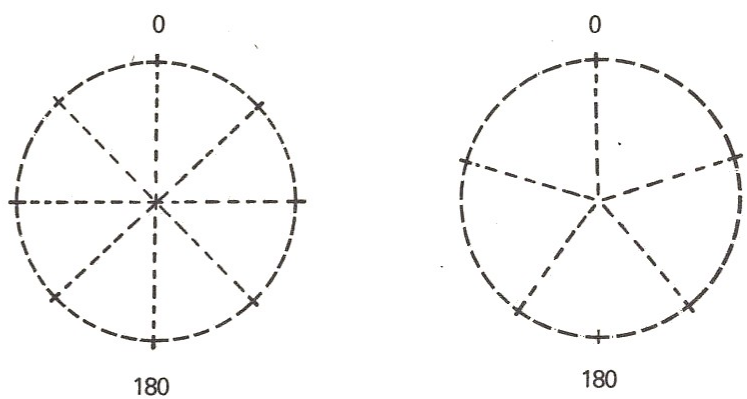
Mas este método de desenhar estrelas ainda não é muito útil, pois temos que "chutar" um ângulo qualquer para ver que estrela sairá, e com certeza você não "chutará" o ângulo de 163.636 graus, que é o ângulo necessário para obtermos uma estrela de 11 pontas. Vejamos se conseguimos melhorar e simplificar a construção de estrelas, tentando responder à pergunta nº 2 — qual é o ângulo necessário para desenharmos uma estrela de 12 pontas?

EXPERIMENTE!!!

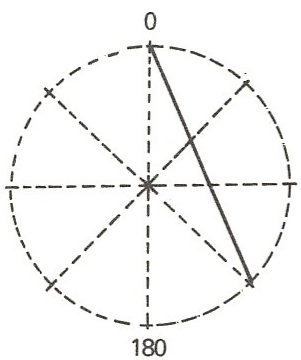
Para compreender a construção do novo procedimento, o qual chamaremos de **ESTRELA2** , vamos estudar como desenhá-las.

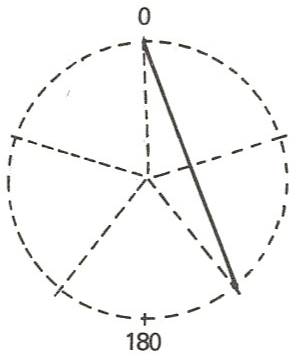
Como exemplo, vamos desenhar simultaneamente uma estrela de oito e uma de cinco pontas. Para tanto, basta seguir os seguintes passos:

1. Desenhe uma circunferência e divida-a em partes iguais, tantas vezes quantas forem as pontas, marcando os ângulos 0 e 180 graus:

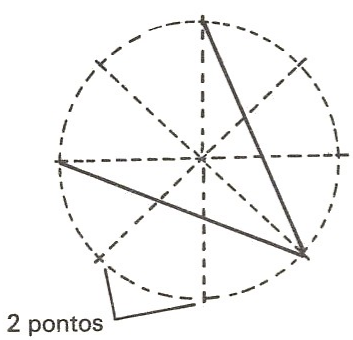


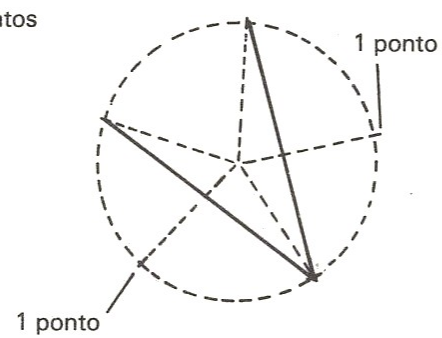
2. Una o ponto da origem (ângulo 0) ao ponto que possui o maior ângulo menor do que 180 graus:



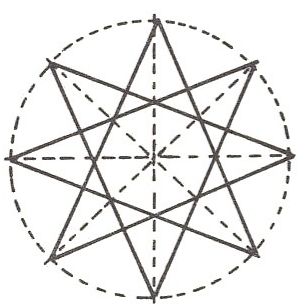


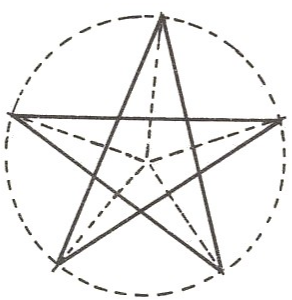
3. Observe quantos pontos "pulamos" para unir estes dois e este último ao próximo, mantendo a mesma distância:





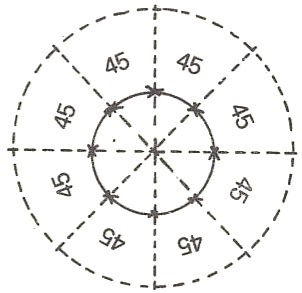
4. Faça o mesmo até voltar ao ponto 0.



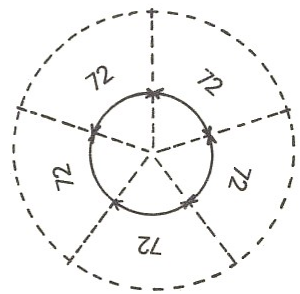


Quando dividimos a circunferência em partes iguais (tantas quantas forem as pontas), os ângulos determinados por cada parte são iguais, observe:

360	8
40	45
0	

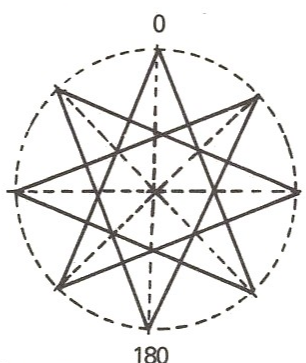


360	5
10	72
0	

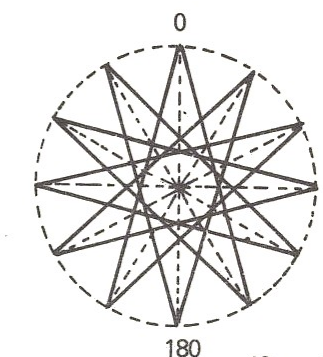


Porém, para unir o ponto da origem com o que possui maior ângulo menor do que 180, quantos graus a tartaruga terá que girar?

Nas estrelas de número par de pontas, um dos pontos sempre determina um ângulo de 180 graus a partir da origem, observe:



8 pontas



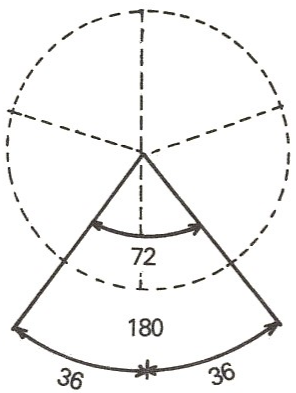
12 pontas

Portanto, o ponto que possui maior ângulo menor do que 180 possui exatamente **180 – 360/N** graus, onde **N** é o número de pontas da estrela.

Assim, podemos determinar o ângulo que a tartaruga deverá girar a cada vez:

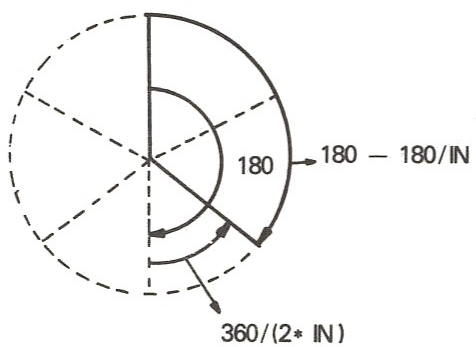
A = 180 – 360/N

Porém, quando a estrela possui um número ímpar de pontas, o ângulo de 180 graus a partir do 0 divide o ângulo determinado por dois pontos em 2 partes iguais, note:



A tartaruga deverá girar 180 — a metade de **360/N**, isto é, **180 – (360/N) / 2**, ou melhor, **180 – 360/2N**

Portanto: **A = 180 – 360 / (2 * N)**



Resumindo:

A = 180 – 360 / N se **N** for par

e

A = 180 – 360 / (2 * N) se **N** for ímpar.

EXPERIMENTE!!!

Vamos, agora, criar o procedimento **ESTRELA2** :

```
AP ESTRELA2 :N :T
TESTE RESTO :N 2 = 0
SV FACA "A 180-360/:N
SF FACA "A 180-360/(2*:N)
REPITA :N [ FR :T DI :A ]
( MO [ESTA ESTRELA TEM] :N [PONTAS E
SEUS ANGULOS MEDEM] :A "GRAUS. )
FIM
```

verifica se o número de pontas da estrela é par

Digite:

1. `ESTRELA2 16 100`
2. `ESTRELA2 24 100`
3. `ESTRELA2 9  100`
4. `ESTRELA2 11 100`

Porém, se digitarmos `ESTRELA2 10 100` ,

obteremos:

```
ESTA ESTRELA POSSUI 10 PONTAS E
SEUS ANGULOS MEDEM 144 GRAUS.
```

Note que apesar da mensagem dizer que a estrela possui 10 pontas, o desenho só possui 5.

Isto ocorre porque o ângulo referente à estrela de 10 pontas é igual ao da de 5, veja:

A = 180 – 360 / 10 **porque 10 é par.**
A = 180 – 36
A = 144

e

A = 180 – 360 / 2 * 5 **porque 5 é ímpar.**
A = 180 – 360 / 10
A = 180 – 36
A = 144

Este fato ocorrerá sempre que a metade do número de pontas determinado for ímpar. Exemplo:

6 pontas (6/2 = 3 que é ímpar)
14 pontas (14/2 = 7 que é ímpar)
22 pontas (22/2 = 11 que é ímpar)

Então, vamos incluir em **ESTRELA2** a ordem:

```
SE RESTO (:N/2 ) 2 = 1 MO [ NAO SEI
DESENHA-LA ] PARE
```

que verifica se a metade do número de pontas é ímpar. **ESTRELA2** ficará:


```
AP ESTRELA2 :N :T
SE RESTO (:N/2) 2 =1 MO [NAO SEI
DESENHA'-LA] PARE
TESTE RESTO :N 2 = 0
SV FACA "A 180-360/:N
SF FACA "A 180-360/(2*:N)
( MO [ESTA ESTRELA TEM] :N [PONTAS E
SEUS ANGULOS MEDEM] :A "GRAUS. )
FIM
```

Experimente desenhá-las.

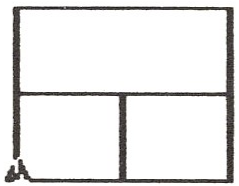
V. PROCEDIMENTOS COM MAIS DE UMA RECURSÃO

Podemos criar procedimentos com mais de uma chamada recursiva, pois, como vimos na recursão simples, quando estabelecemos uma condição para interrupção da execução, e quando esta condição é satisfeita, o controle é devolvido ao nível anterior, e assim sucessivamente, até que o nível 1 finalize por completo a execução.

Para exemplificar, tomemos o procedimento **QUADROS** :

```
AP QUADROS :L
SE :L < 3 ENTAO PARE
REPITA 4 [ FR :L DI 90 ]
QUADROS :L/2
DI 90
FR :L/2
ES 90
QUADROS :L/2
ES 90
FR :L/2
DI 90
FIM
```

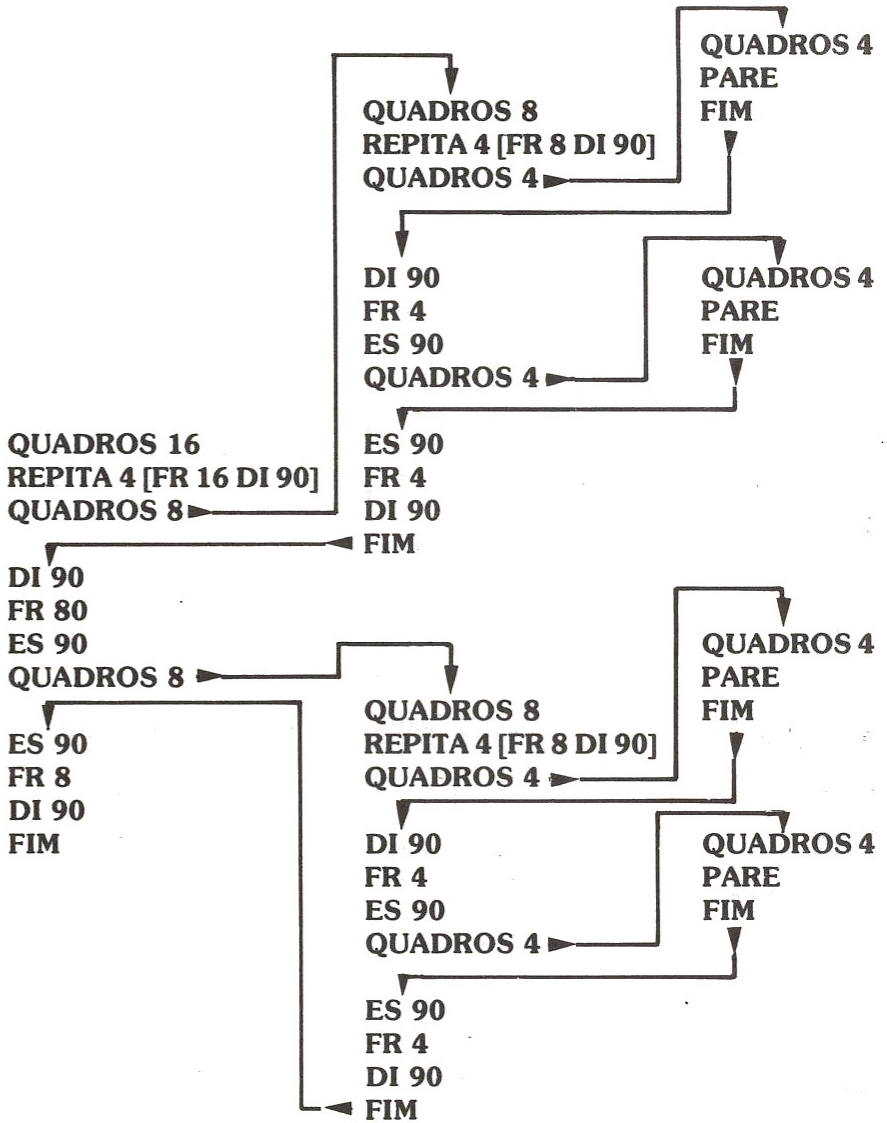
Ao digitarmos **QUADROS 16** obteremos:



Explicaremos sua execução através de esquemas, porém, antes, vamos mostrar como ficaram seus valores nos diferentes níveis:

NÍVEL 1	NÍVEL 2	NÍVEL 3
QUADROS 16	QUADROS 8	QUADROS 4
REPITA 4 [FR 16 DI 90]	REPITA 4 [FR 8 DI 90]	PARE
QUADROS 8	QUADROS 4	FIM
DI 90	DI 90	
FR 8	FR 4	
ES 90	ES 90	
QUADROS 8	QUADROS 4	
ES 90	ES 90	
FR 8	FR 4	
DI 90	DI 90	
FIM	FIM	

Passemos ao esquema de execução de **QUADROS 16**



Digite então: **QUADROS 32** , **QUADROS 20** , **QUADROS 80** e outros valores quaisquer.

Vejamos, como exemplo, a construção de uma **árvore binária**, onde de cada ramo partem outros dois:

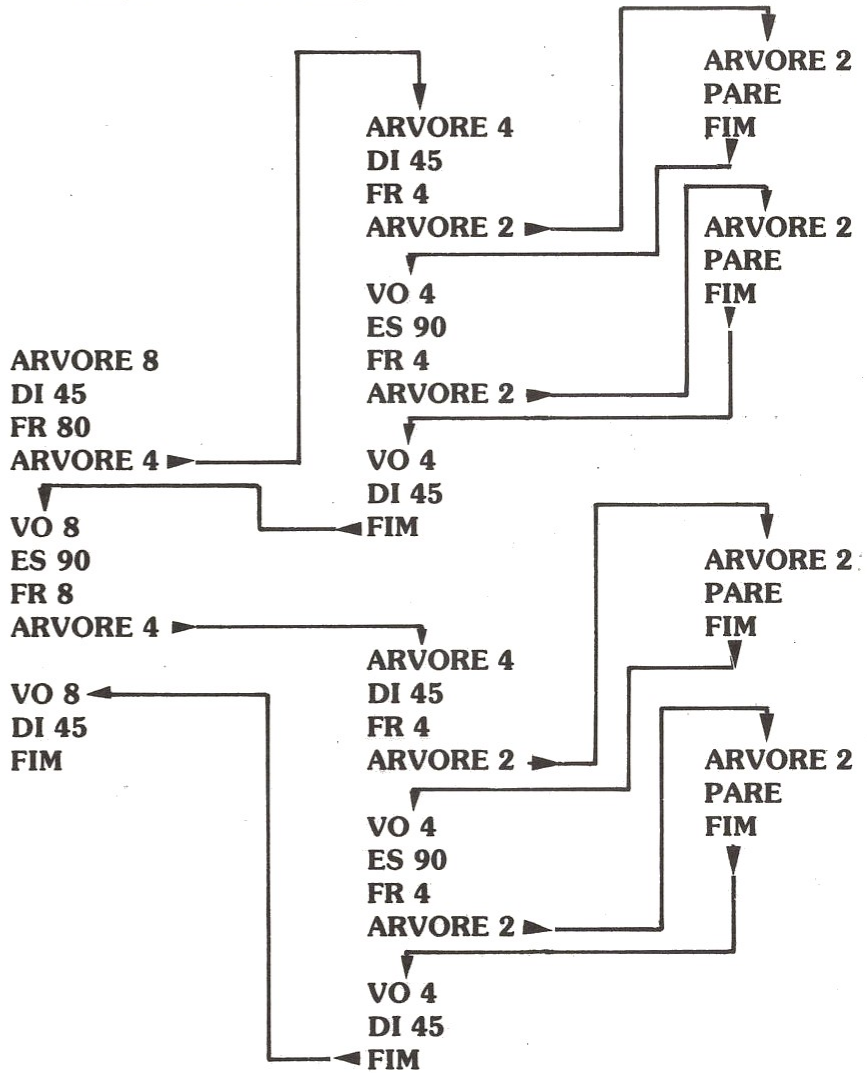
```
AP ARVORE :T
SE :T < 3 PARE
DI 45
FR :T
ARVORE :T/2
VO :T
ES 90
FR :T
ARVORE :T/2
VO :T
DI 45
FIM
```

Digitando **ARVORE 8** obteremos:

Valores das variáveis nos diferentes níveis:

NÍVEL 1	NÍVEL 2	NÍVEL 3
ARVORE 8	ARVORE 4	ARVORE 2
DI 45	DI 45	PARE
FR 8	FR 4	FIM
ARVORE 4	ARVORE 2	
VO 8	VO 4	
ES 90	ES 90	
FR 8	FR 4	
ARVORE 4	ARVORE 4	
VO 8	VO 4	
DI 45	DI 45	
FIM	FIM	

Esquema de resolução:



APÊNDICE

COMANDOS PARA OPERAÇÃO DE MEMÓRIA E DISCO

São comandos que nos possibilitam levar procedimentos da memória do computador para o disquete e vice-versa. São úteis ainda para eliminarmos procedimentos indesejados.

COMANDO	FUNÇÃO
ADEUS	Apaga tudo o que estiver na memória do computador, exceto os comandos LOGO.
VERNOMES	Mostra na tela a relação de nomes dos procedimentos que estão na memória do computador.
GRAVE "n	Ordena que os procedimentos existentes na memória do computador sejam gravados no disco.
GRAVED "n	Permite a gravação em disco do conteúdo da tela gráfica (desenho).
VERDISCO	Mostra os nomes de todos os arquivos contidos no disquete.

LEIA "n	Lê do disco os procedimentos que foram gravados da memória do computador, mostrando-os no vídeo.
LEIAD "n	Lê o arquivo gráfico (desenho) previamente armazenado em disco, mostrando seu conteúdo no vídeo.
SEM n	Apaga um procedimento da memória do computador. **n** é o nome do procedimento que queremos apagar.
SEMARQUIVO "n	Remove do disco o arquivo especificado pelo nome do procedimento (**n**), precedido de aspas.
SEMDESENHO "n	Apaga do disco o desenho especificado pelo nome (**n**), sendo este precedido de aspas (**"**).

TABELA ASCII

Na tabela abaixo, você poderá encontrar as teclas com seus respectivos valores no sistema decimal:

VALOR	TECLAS
1	**CTRL A**
2	**CTRL B**
3	**CTRL C**
4	**CTRL D**
5	**CTRL E**
6	**CTRL F**
7	**CTRL G**

VALOR	TECLAS
46	**.**
47	**/**
48	**0**
49	**1**
50	**2**
51	**3**
52	**4**

8	CTRL H ou ⟶
9	CTRL I
10	CTRL J
11	CTRL K
12	CTRL L
13	CTRL M ou RETURN
14	CTRL N
15	CTRL O
16	CTRL P
17	CTRL Q
18	CTRL R
19	CTRL S
20	CTRL T
21	CTRL U ou ⟶
22	CTRL V
23	CTRL W
24	CTRL X
25	CTRL Y
26	CTRL Z
27	ESC
28	n.d.
29	CTRL SHIFT M
30	CTRL
31	n.d.
32	ESPACO
33	!
34	"
35	§
36	$
37	%
38	&
39	'
40	(
41	)
42	*
43	+
44	,
45	—

53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?
64	
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z

MENSAGENS DE ERRO DO *MLOGO*

Descreveremos, aqui, algumas mensagens de erro que poderão surgir ao se trabalhar com LOGO, além das mencionadas no livro ***LOGO — O Primeiro Passo***.

1. 'comando' **NAO ACEITA** 'dado' **COMO DADO. ACEITA VERD OU FALSO**

Ocorre quando atribuímos a comandos como SE, SEUM, SETODOS ou TESTE, dados que estes não aceitam. Lembre-se de que estes comandos só aceitam como dado resultados de operações lógicas.

2. 'comando' **SOMENTE PODE SER USADO EM ROTINAS**

Ocorre quando utilizamos comandos do tipo SAIDA, PARE ou FIM fora de procedimentos.

3. **EI! DIVISAO POR ZERO**

Ocorre quando tentamos dividir qualquer número por zero.

4. **'ENTAO' ESTA' ERRADO ou 'SENAO' ESTA' ERRADO**

Ocorre quando utilizamos SE...ENTAO ou SE...ENTAO ...SENAO de forma imprópria. Lembre-se de que estas ordens têm que estar todas numa única linha do procedimento.

5. **MUITOS DADOS NOS PARENTESES**

Ocorre quando utilizamos os parênteses de forma inadequada, não permitindo ao LOGO compreender a expressão que lá se encontra.

CONVERSÃO DOS COMANDOS MLOGO UTILIZADOS NESTE LIVRO PARA O LOGO ITAUTEC

MLOGO		ITAUTEC	
COMANDO	ABREV.	COMANDO	ABREV.
ASC		ASCII	
CARAC		CARACTER	CAR
CP?		TEMCAR	
CURSOR		MUDECURSOR	
ENTRE	EN	LIN.ENTRADA	LINE
ESCREVA		TELATEXTO	TT
EXECUTE		FAÇA	
FACA		ATRIBUA	ATR
INT		INT	
LIMPETEXTO		APAGUETEXTO	ATT
LISTA?		E'LISTA	
MOSTRE	MO	ESCREVA	ESC
MOSTRAR		—	
NUMERO?		E'NUMERO	
PALAVRA		PALAVRA	PAL
PALAVRA?		E'PALAVRA	
PARE		PARE	
PEGUE		CAR.ENTRADA	CARE
PRIMEIRO		PRIMEIRO	PRI
QUOC		DIV	
RESTO		RESTO	
RQD		RAIZQ	RQ
SAIDA	SA	ENVIE	
SE		SE	
SEFALSO	SF	SEFALSO	
SEMPRIMEIRO	SP	SEMPRIMEIRO	SP
SEMULTIMO	SU	SEMÚLTIMO	SU
SENTENCA	SN	SENTENÇA	SN
SETODOS		TODOS	
SEUM		ALGUM	
SEVERD	SV	SEVERD	
SORTEIE		SORTEIEATÉ	
TESTE		TESTE	
ULTIMO		ÚLTIMO	ULT