MANUAL DO PROGRAMADOR EM

BASIC

Aprenda a linguagem de programação mais usada em todo o mundo

DICAS DE PROGRAMAÇÃO para melhorar o seu BASIC

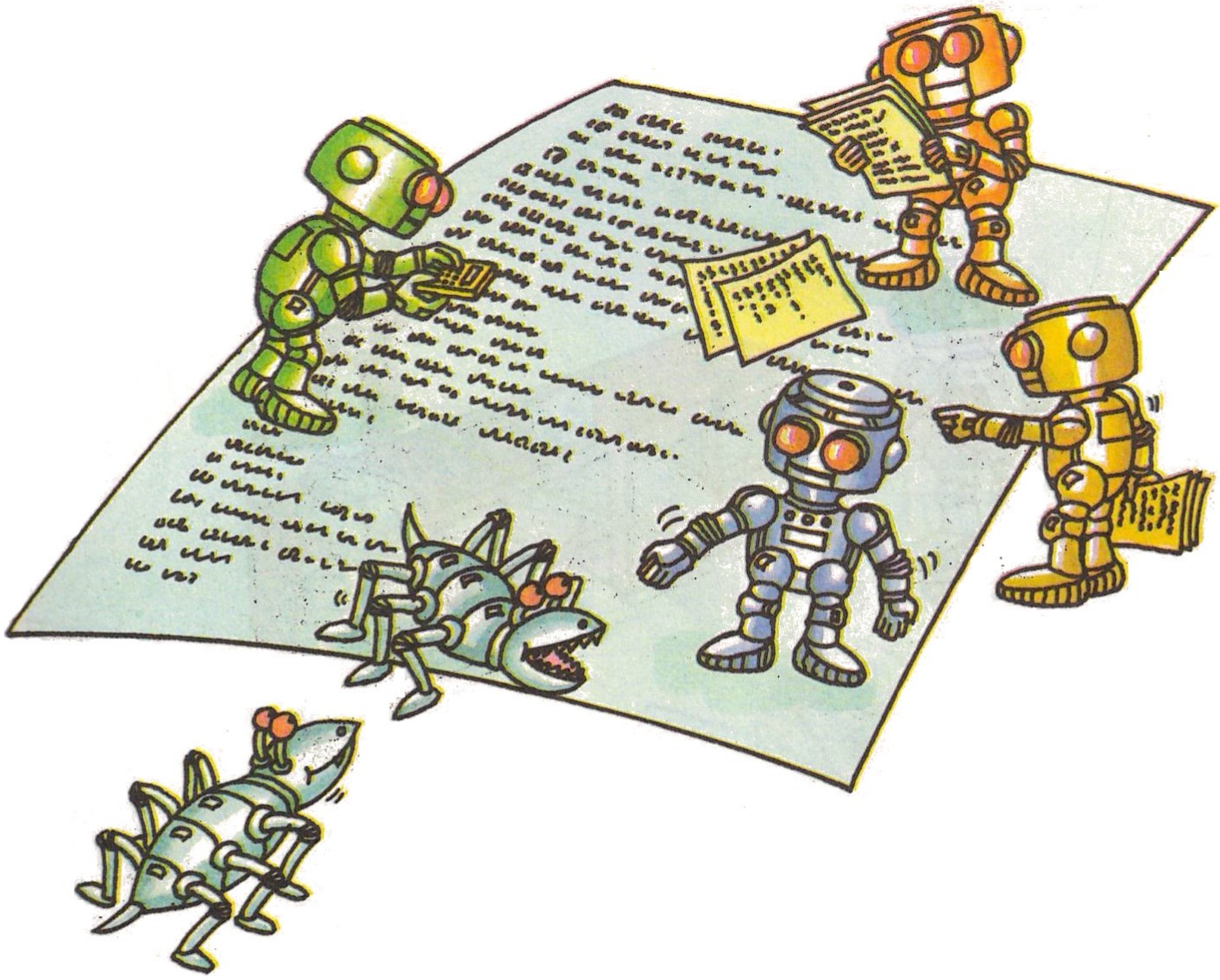
EDITORA LUTÉCIA

MANUAL DO PROGRAMADOR EM BASIC

Brian Reffin Smith
e Lisa Watts

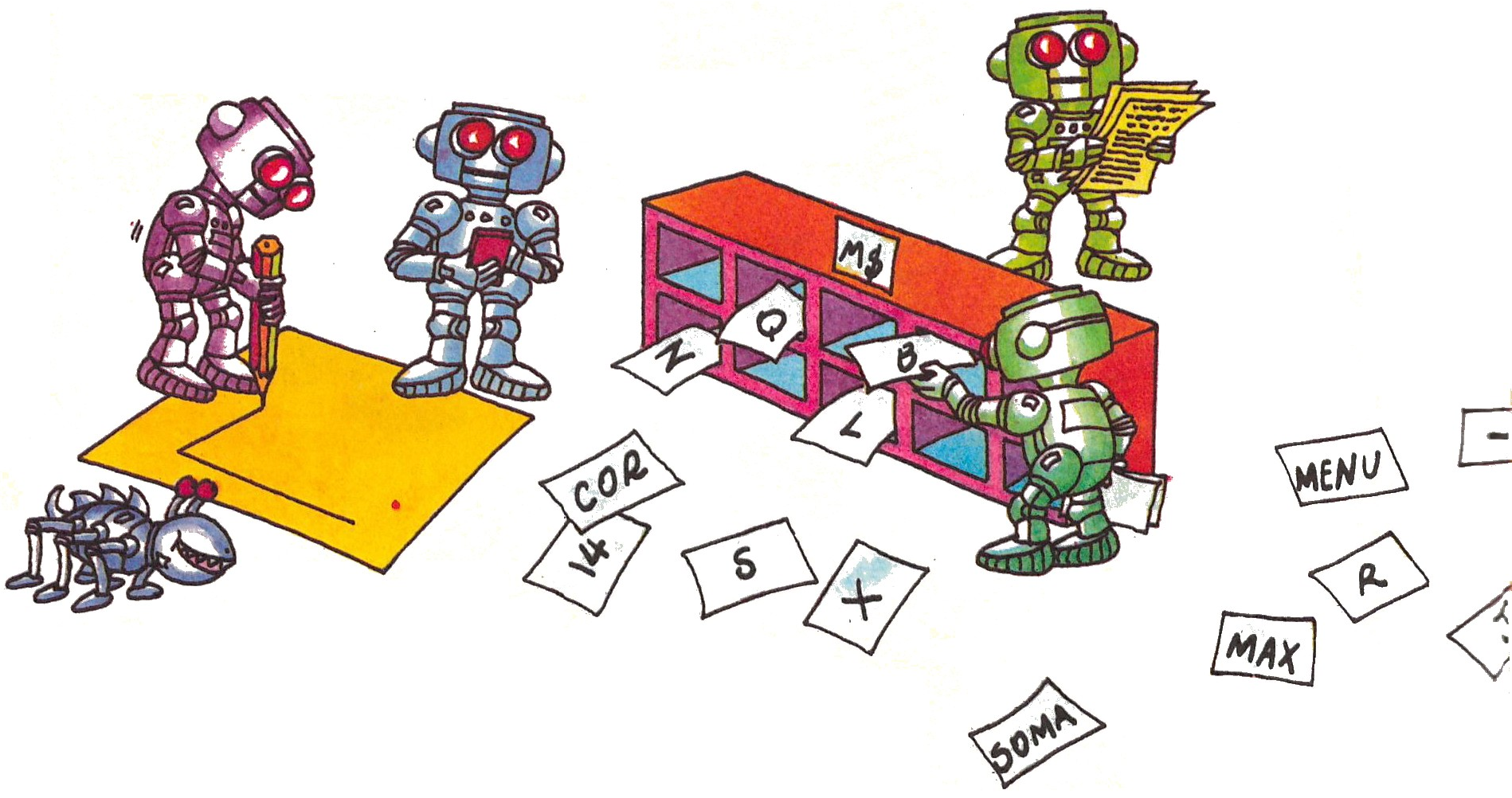
Tradução e Adaptação de Ronaldo Sergio de Biasi

Ilustrações e Diagramação: Graham Round



LUTÉCIA

Sumário



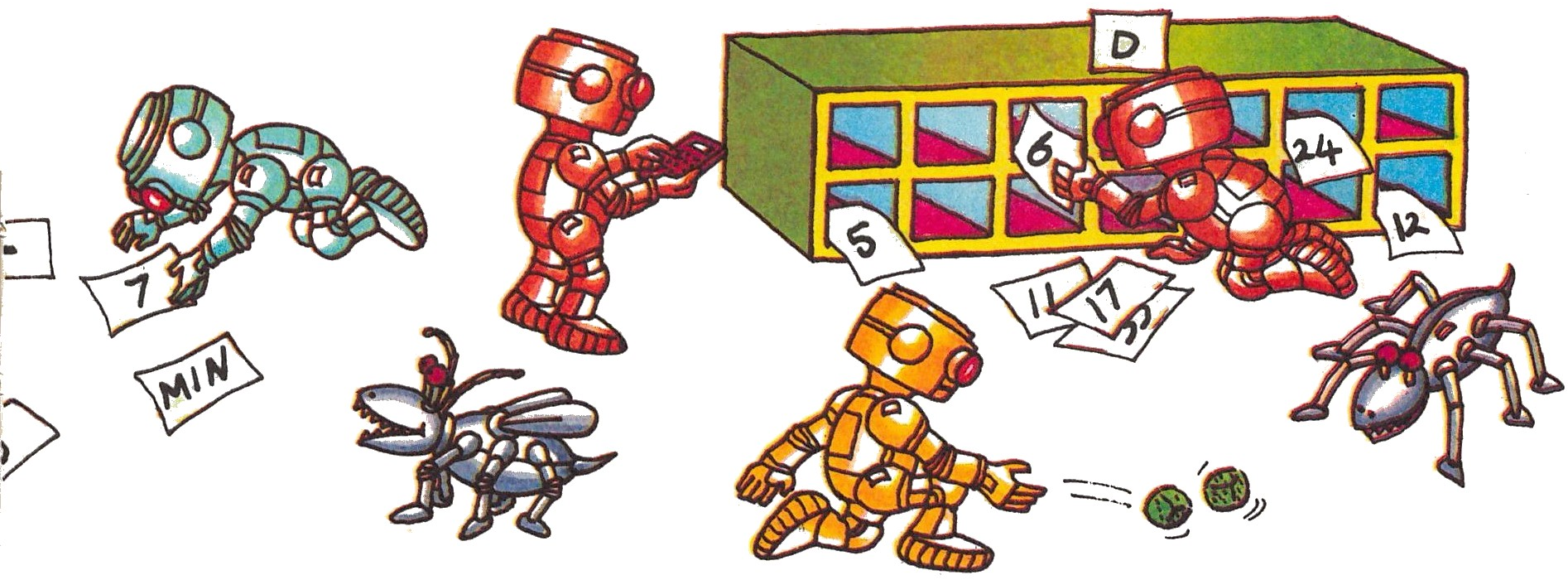
Apresentação

Este livro tem por objetivo ajudá-lo a compreender programas escritos em BASIC e melhorar suas técnicas de programação nessa linguagem. Nem todos se interessam em escrever seus próprios programas, mas mesmo para adaptar ou corrigir programas escritos por outras pessoas, é preciso conhecer os fundamentos da linguagem BASIC.

No início do livro são explicados os principais comandos de BASIC e seu uso é ilustrado através de numerosos exemplos. Em seguida, o livro mostra como os comandos podem ser usados em programas para fazer coisas bastante complicadas, como criar uma base de dados, desenhar na tela e ordenar dados.

Os programas deste livro foram escritos em BASIC "padrão", isto é, em uma versão de BASIC que, com pequenas alterações, pode ser usada na maioria dos microcomputadores. A adaptação dos programas a modelos particulares de micros é discutida nas páginas 10 e 11; a conversão para os micros da família Sinclair, que usam um dialeto de BASIC um pouco diferente, é tratada no final do livro.

Todos os programas são acompanhados por uma descrição detalhada de seu funcionamento e de técnicas e rotinas que você pode aplicar em seus próprios programas. Existem também muitas idéias para modificar e expandir os programas, adaptando-os às suas necessidades.



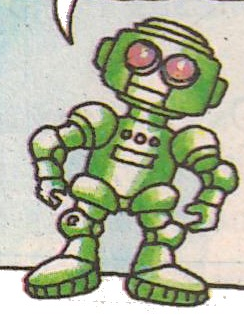
Introdução ao BASIC

A linguagem de programação BASIC utiliza cerca de cem palavras. Cada palavra é uma instrução que diz ao computador para fazer alguma coisa. Para que o computador execute uma tarefa, é preciso fornecer-lhe uma lista de instruções e dados que é chamada de programa. Você só pode usar palavras de BASIC como instruções, e além disso deve respeitar as regras da linguagem.

Em BASIC, cada linha de instruções tem um número. As linhas são geralmente numeradas de 10 em 10, para que mais tarde seja possível acrescentar novas linhas sem ter que numerar de novo todo o programa.

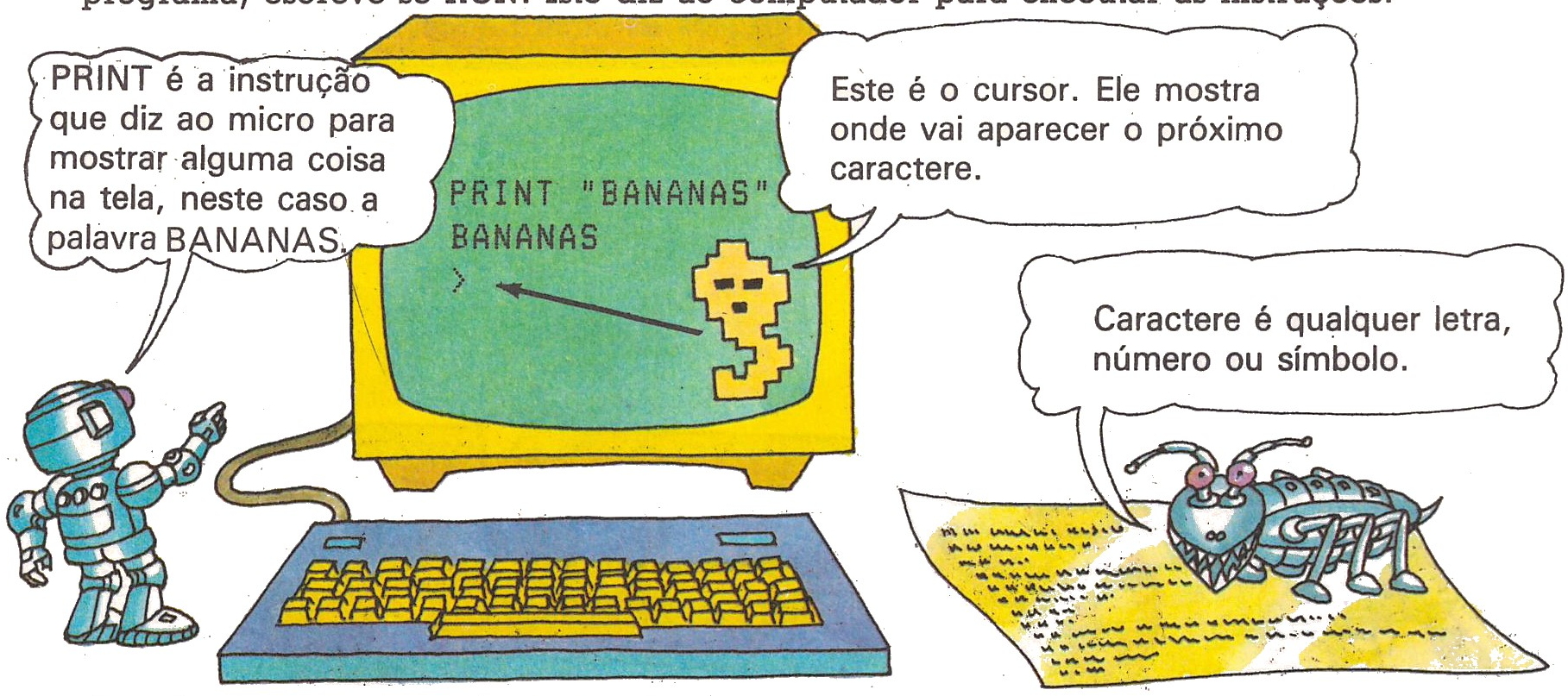
```
 10 CLS
 20 PRINT "DECOLAGEM"
 30 LET G=INT(RND(1)*20+1)
 40 LET W=INT(RND(1)*40+1)
 50 LET R=G*W
 60 PRINT "GRAVIDADE ";G
 70 PRINT "DIGA A FORCA"
 80 FOR C=1 TO 10
 90 INPUT F
100 IF F>R THEN PRINT "MUITO GRANDE";
110 IF F<R THEN PRINT "MUITO PEQUENA";
120 IF F=R THEN GOTO 190
130 IF C<>10 THEN PRINT
140 NEXT C
150 PRINT
160 PRINT "VOCE FALHOU"
170 PRINT "E FOI CAPTURA
180 STOP
190 PRINT "BOA VIAGEM"
200 STOP
```

Este é um programa escrito em BASIC.



Como carregar os programas

Quando você está carregando um programa no computador, deve apertar a tecla RETURN (ou ENTER, NEWLINE ou CR, dependendo do micro) no final de cada linha. Isto faz o computador guardar na memória a linha que você acabou de bater e ficar à espera da linha seguinte. Depois de carregar todas as linhas do programa, escreve-se RUN. Isto diz ao computador para executar as instruções.



Quando você escreve uma instrução sem número de linha, o computador a executa imediatamente, logo que você aperta a tecla RETURN (ou ENTER, NEWLINE ou CR). Isto é chamado de comando direto. Assim, por exemplo, para dizer ao computador para mostrar as linhas de um programa que acaba de ser carregado, basta escrever LIST como um comando direto. Para apagar o que está escrito na tela, na maioria dos micros, basta escrever CLS.

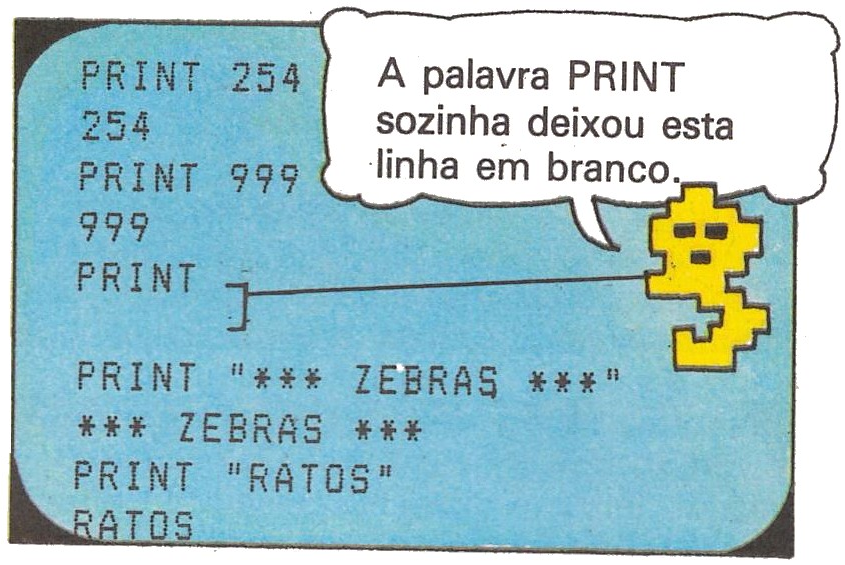
Você precisa tomar cuidado para carregar os programas exatamente como foram escritos. Se escrever uma das palavras de BASIC com a grafia errada ou esquecer algum sinal de pontuação, o computador não poderá seguir as instruções. A maioria dos erros nos programas são erros de datilografia, mas às vezes existem erros na lógica de um programa que podem levar a resultados imprevistos.

Instruções de BASIC

Nas próximas páginas, vamos examinar os principais comandos de BASIC e a forma de utilizá-los. Se você tem um micro, deve verificar os comandos no seu manual, já que algumas das palavras e regras variam ligeiramente de computador para computador.

PRINT ►

Esta instrução diz ao computador para mostrar alguma coisa na tela. Como vemos nos exemplos à direita, as letras e símbolos devem ser colocados entre aspas, mas não há necessidade de fazer isso com os números. Nesses exemplos não há número de linha, de modo que o computador executa a instrução assim que você aperta a tecla RETURN. O computador mostra exatamente o que você escreveu entre as aspas, inclusive os espaços. A palavra PRINT sozinha diz ao computador para deixar uma linha em branco.



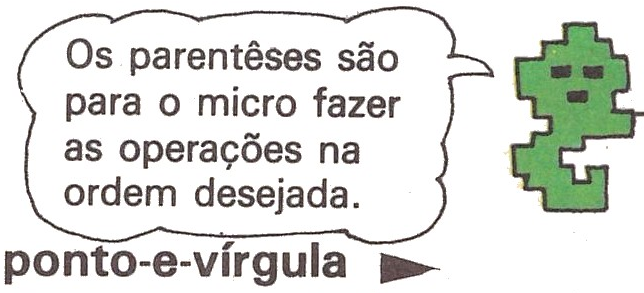
Fazendo cálculos com PRINT ►

Você também pode usar PRINT para fazer contas. O computador usa os sinais comuns de somar e subtrair, mas para multiplicar usa o sinal * e para dividir o sinal / . SQR(N) é a instrução para calcular a raiz quadrada de um número, N. O sinal ↑ ou ^ ou ** significa elevado a. Assim, por exemplo, 3*2 significa 3 elevado a 2, ou 3 ao quadrado.

No caso de expressões complicadas, o computador sempre executa as multiplicações e divisões antes das somas e subtrações. Para evitar que isso aconteça, é preciso usar parênteses. Nas expressões com vários parênteses, o computador sempre começa pelos parênteses mais internos.

```
PRINT 12095+277
12372
PRINT 239-51
188
PRINT 17*5
85
PRINT 221/13
17
```

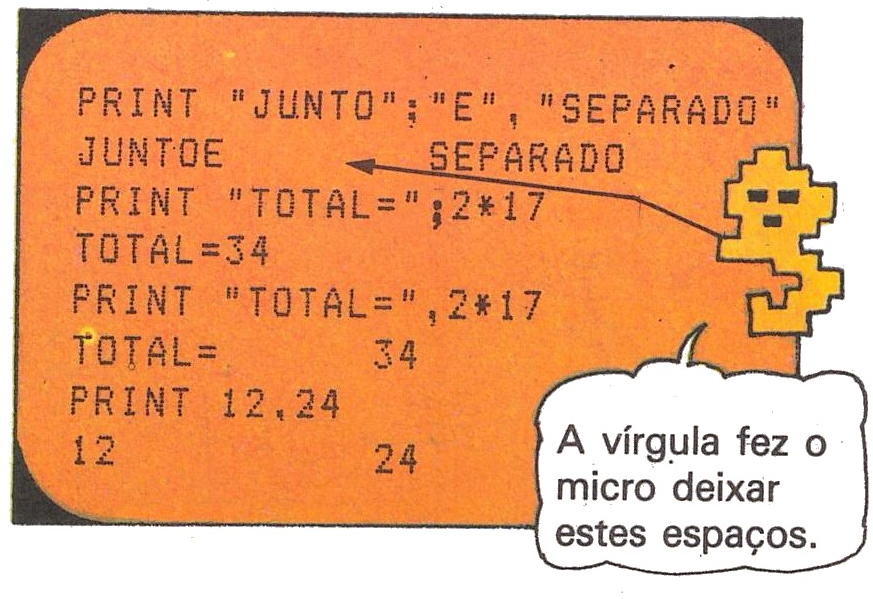
```
PRINT SQR(9)
3
PRINT SQR(9)+3/\2
12
PRINT 2*17-5
29
PRINT 2*(17-5)
24
```



Vírgula e ponto-e-vírgula ►

Estes sinais dizem ao computador em que ponto da tela deve mostrar o sinal seguinte. O ponto-e-vírgula diz ao micro para mostrar o sinal na mesma linha, sem deixar nenhum espaço; a vírgula, para deixar um certo número de espaços (que varia de micro para micro).

Alguns computadores precisam de uma vírgula ou ponto-e-vírgula para separar instruções PRINT de dados ou variáveis (letras que representam dados armazenados na memória do computador). Experimente os exemplos ao lado para ver como funcionam no seu micro.



Variáveis ►

As informações fornecidas ao computador são chamadas de dados. Quando você fornece um dado para o computador guardar na memória, precisa fornecer também um nome para o dado. Este nome é chamado de variável; sempre que você quer que o computador faça alguma coisa com o dado, você se refere a ele pelo nome da variável. O nome é chamado de variável porque o dado a que se refere pode mudar durante o programa.

Você pode usar letras do alfabeto, ou letras e números, como A6, por exemplo, para designar dados numéricos. Quando o dado também possui letras e símbolos, é chamado de string. A variável usada para designar uma string deve terminar com um cifrão, como P6$, por exemplo. O número máximo de letras e números permitido para uma variável muda de acordo com o modelo de micro.

LET ►

Esta é uma forma de fornecer dados ao computador. LET A = 5 diz ao computador para guardar o número 5 na memória e rotulá-lo como A; LET C$ = "COELHOS" guarda a palavra *coelhos* em um espaço de memória chamado de C$. As strings devem sempre estar entre aspas mas os números não precisam de aspas.

```
10 LET A=5
20 LET C$="COELHOS"
30 PRINT A
40 PRINT C$
RUN
5
COELHOS
```

Para mostrar os dados na tela, basta usar PRINT seguido pelo nome da variável.

INPUT ►

Esta é uma forma de fornecer dados ao computador durante a execução do programa. A palavra INPUT é seguida pelo nome de uma variável, e quando o computador chega à instrução INPUT, mostra na tela um ponto de interrogação (ou outro símbolo) e espera o operador entrar com o dado. Quando a variável que vem depois de INPUT é uma variável numérica, o dado tem que ser um número; quando é uma variável string, esta limitação não existe. Na maioria dos micros, é possível colocar palavras entre aspas em uma instrução INPUT para torná-la mais clara, como no segundo exemplo à direita. Na maioria dos micros, é preciso usar um ponto-e-vírgula entre a mensagem e o nome da variável.

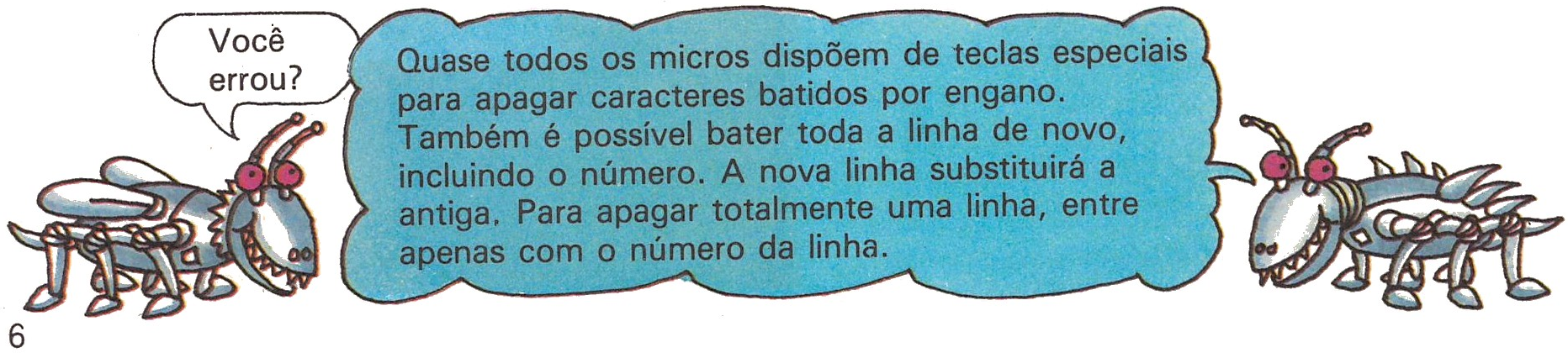
```
10 PRINT "QUAL E O SEU NOME?"
20 INPUT N$
RUN
QUAL E O SEU NOME?
?ROBERTO
```

Interrogação do micro.

A resposta da pessoa é guardada em N$.

```
10 INPUT"QUAL E O SEU NOME";N$
20 INPUT"QUANTOS ANOS TEM";A
RUN
QUAL E O SEU NOME? ROBERTO
QUANTOS ANOS TEM? 21
```

Interrogações do computador



READ/DATA►

Esta é outra forma de fornecer dados ao computador. A palavra READ é seguida pelo nome de uma ou mais variáveis, e os dados para as variáveis são colocados em uma linha que começa com a palavra DATA. A linha de dados pode ser colocada em qualquer lugar do programa. Quando o computador chega à instrução READ, ele procura a palavra DATA e associa os dados às respectivas variáveis, respeitando a ordem em que aparecem. Os dados devem ser separados por vírgulas; em alguns micros, todos os dados não-numéricos (strings) devem estar entre aspas. Em outros, só é preciso usar aspas se as strings contêm espaços ou sinais de pontuação.

```
10 READ A$,B,C$,N
.
.
.
.
100 DATA PLATINADOS, 2000
110 DATA "LONAS DE FREIO", 500
```

Este dado está entre aspas porque contém espaços.

IF/THEN►

Esta é uma forma de testar dados e dizer ao computador para fazer certas coisas, dependendo do resultado. Usando os símbolos que aparecem à direita, é possível verificar se dois dados são iguais, diferentes e se um é maior ou menor que o outro. Quase todas as instruções podem ser usadas depois da palavra THEN, mas quando a condição não é satisfeita, o computador ignora o que vem depois do THEN e passa para a linha seguinte.

```
IF A=B THEN PRINT "SAO IGUAIS"
IF X<>Y THEN PRINT "SAO DIFERENTES"
IF X>Y THEN PRINT "X E MAIOR"
IF X<Y THEN PRINT "X E MENOR"

IF A$="NAO" THEN STOP
IF X+Y=5 THEN LET X=X+1
```

Esta instrução é para interromper a execução do programa.

GOTO►

Esta instrução diz ao computador para pular para outra linha do programa. Geralmente é usada com IF/THEN, para que o pulo só ocorra em certas condições. Cuidado ao usar GOTO isoladamente, pois há sempre o perigo de você criar um loop sem saída, como no exemplo à direita. A única maneira de fazer parar este programa seria apertando a tecla BREAK ou ESCAPE (o nome da tecla varia de micro para micro).

```
100 IF A=5 THEN GOTO 175
.
.
.
.
.
175 PRINT "MEUS PARABENS"
180 PRINT "*VOCE GANHOU*"
185 GOTO 180
```

Evite usar o GOTO para criar um loop sem saída, como na linha 185.

GOSUB/RETURN►

GOSUB faz o computador pular para uma sub-rotina, uma parte do programa destinada a executar uma tarefa específica. A palavra RETURN, no final da sub-rotina, manda o computador de volta para a instrução seguinte a GOSUB. Se você esquecer a instrução RETURN, o programa não funcionará.

```
100 INPUT "QUER JOGAR? ";T$
110 IF T$="SIM" THEN GOSUB 500
120 REM INICIO DO JOGO
.
.
.

500 REM SUB-ROTINA BOM DIA
510 PRINT "QUAL E O SEU NOME?"
520 INPUT N$
530 PRINT "BOM DIA, ";N$
540 RETURN
```

O computador ignora esta linha.

REM►

O computador ignora as linhas que começam com a palavra REM. Ela é usada para incluir comentários no programa.

A linha 540 manda o computador de volta para a instrução que vem depois de GOSUB.

Loops com FOR/NEXT ►

As palavras FOR, TO e NEXT fazem o computador repetir parte de um programa um certo número de vezes. No exemplo à direita, as linhas 10 e 30 são repetidas três vezes e de cada vez o computador imprime a mensagem da linha 20. J é uma variável usada para contar o número de repetições. A linha 30 manda o computador de volta para verificar qual é o valor seguinte de J; cada vez que o loop é repetido, o valor de J é aumentado de 1. Quando J = 3, o computador passa direto pela instrução NEXT.

```
10 FOR J=1 TO 3
20 PRINT "J VALE ";J
30 NEXT J
40 PRINT
RUN
J VALE 1
J VALE 2
J VALE 3
```

A variável J no final da linha faz o micro mostrar o valor de J cada vez que o loop é repetido.

STEP ►

Esta palavra é usada para mudar o incremento da variável de controle do loop. Assim, por exemplo, FOR J = 1 TO 10 STEP 2 faz J aumentar de 2 cada vez que o loop é repetido e STEP X faria J aumentar do valor armazenado em X. No exemplo à direita, STEP —1 faz o computador contar para trás.

```
10 FOR J=10 TO 1 STEP-1
20 PRINT J;" DIAS PARA O NATAL"
30 NEXT J
40 PRINT "FELIZ NATAL"
RUN
10 DIAS PARA O NATAL
 9 DIAS PARA O NATAL
 8 DIAS PARA O NATAL
 7 DIAS PARA O NATAL
```

Loops internos ►

Você pode escrever programas bastante complexos usando loops dentro de loops. Assim, por exemplo, no programa à direita, cada vez que o loop das linhas 10 a 50 é repetido, o loop das linhas 20 a 40 é executado 12 vezes. Cada vez que o loop interno é repetido, o valor de JxI é impresso na linha 30.

Para que não haja erro, tanto o início como o final do loop interno devem estar dentro do loop externo.

```
10 FOR I=2 TO 12
20 FOR J=1 TO 12
30 PRINT J;" VEZES ";I;"=";J*I
40 NEXT J
50 NEXT I
```

Loop externo

Comandos gráficos ►

O computador faz desenhos acendendo na tela pequenos quadrados chamados pixels. As instruções para acender os pixels variam de computador para computador. Os programas deste livro usam a instrução PLOT X,Y onde X e Y são as coordenadas de um pixel. Para traçar uma reta, os programas usam DRAW X,Y. Quase todos os micros têm instruções semelhantes, mas alguns podem precisar de uma instrução especial que especifique o modo gráfico desejado.*

RND ►

Esta instrução faz o computador gerar um número aleatório, mas a forma exata varia de micro para micro. Em alguns modelos, RND(9) gera um número entre 1 e 9. Outros precisam de uma instrução mais complicada, como: INT(RND(1)*9 + 1). O computador calcula primeiro o que está entre parênteses. RND(1) faz o micro gerar um número entre 0 e 1. Esse número é multiplicado por 9, o maior número desejado. É preciso somar 1 porque a palavra INT faz com que o número seja arredondado para baixo.

```
PRINT RND(9)
7
PRINT INT(RND(1)*9+1)
7
```

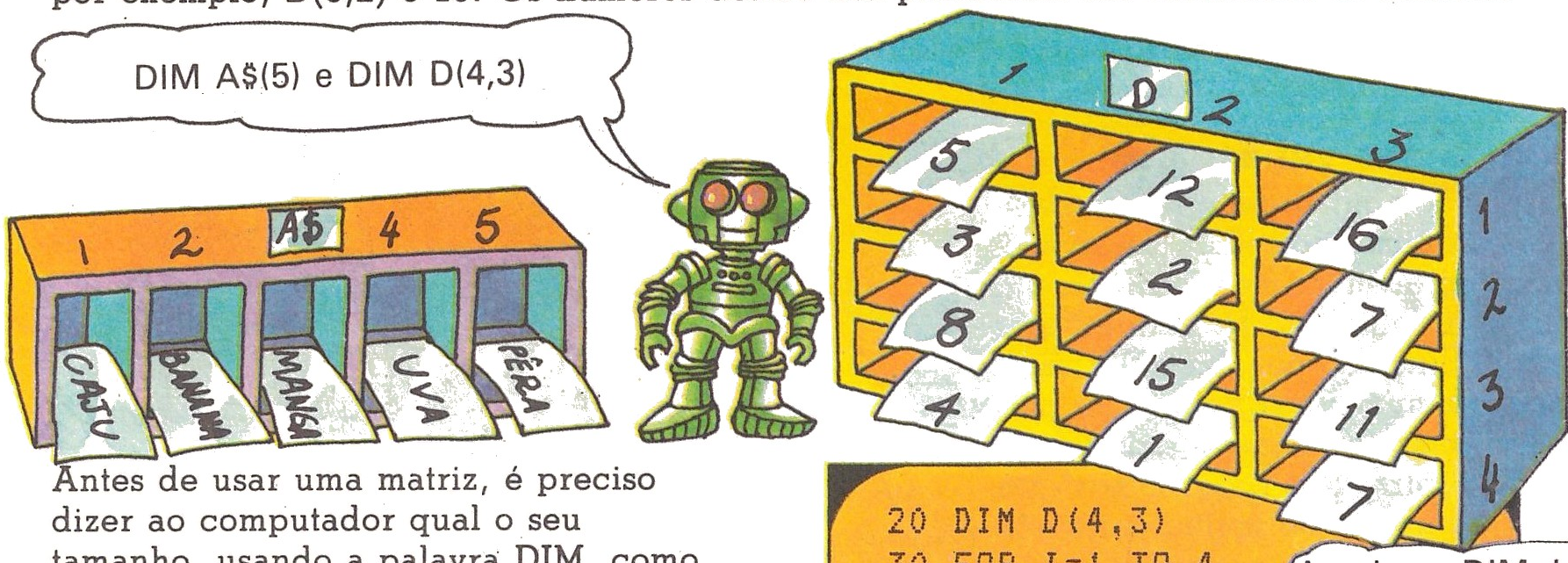
Alguns computadores usam RND(0) em lugar de RND(1).

0.0109425 * 9 + 1 =

*Muitos micros podem funcionar em vários "modos" diferentes; a cada modo corresponde um número diferente de pixels e de cores.

Matrizes

Matriz é um conjunto de dados associados ao nome de uma única variável. Esta variável pode ser imaginada como um espaço na memória do computador com vários compartimentos. As matrizes podem ser unidimensionais, isto é, com uma única fila de escaninhos, ou bidimensionais, com várias filas de escaninhos. Um dado de uma matriz unidimensional é identificado pelo número do escaninho em que se encontra. Assim, por exemplo, na figura abaixo A$(4) é UVA. No caso de matrizes bidimensionais, é preciso indicar o número da fila e da coluna. Assim, por exemplo, D(3,2) é 15. Os números dentro dos parênteses são chamados de índices.



Antes de usar uma matriz, é preciso dizer ao computador qual o seu tamanho, usando a palavra DIM, como no exemplo à direita. Para guardar dados em uma matriz, pode-se usar READ/DATA dentro de um loop. No caso de matrizes bidimensionais, são usados dois loops, um dentro do outro, como no exemplo à direita.

Neste exemplo, I é o número da linha e J o número da coluna. Cada vez que o loop interno J é repetido, um dado é armazenado em uma das colunas da linha. Quando o loop I é repetido, o micro começa uma nova linha.

```
20 DIM D(4,3)
30 FOR I=1 TO 4
40 FOR J=1 TO 3
50 READ D(I,J)
60 NEXT J
70 NEXT I
80 DATA 5,12,16
90 DATA 3,2,7
100 DATA 8,15,11
110 DATA 4,1,7
```

A palavra DIM deve ficar no começo do programa e só pode ser usada uma vez para cada variável.

LEFT$ e RIGHT$

Essas instruções são usadas para manipular os caracteres que estão guardados em uma variável string. Assim, por exemplo, LEFT$(A$,4) diz ao computador para tomar quatro caracteres da esquerda de A$ e RIGHT$(A$,5) para tomar cinco caracteres da direita. Os micros da família Sinclair (TK-83/85, CP-200) não usam estes comandos (vide pág. 11).

```
10 LET A$="PERIQUITO"
20 PRINT LEFT$(A$,4)
25 PRINT LEFT$(A$,2)
30 PRINT
40 PRINT RIGHT$(A$,5)
RUN
PERI
PE

QUITO
```

MID$ e LEN

MID$ diz ao computador para tomar alguns caracteres do meio de uma string e LEN é usado para verificar quantos caracteres tem uma string, incluindo os espaços e sinais de pontuação. Assim, por exemplo, MID$(K$,2,4) significa tomar quatro caracteres do meio de K$, começando pelo segundo. As instruções para os micros da família Sinclair estão na pág. 11.

```
10 LET K$="TRAVESSURA"
20 PRINT MID$(K$,2,4)
25 PRINT MID$(K$,4,4)
30 PRINT
35 PRINT LEN(K$)
RUN
RAVE
VESS

10
```

Este é o número de caracteres em K$.

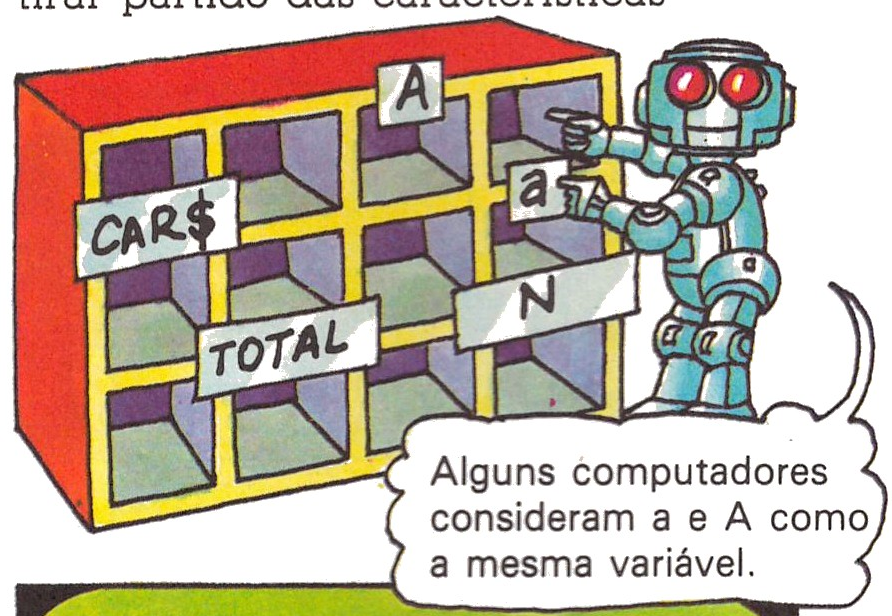
O BASIC usado neste livro

Os programas deste livro estão escritos em BASIC "padrão". Alguns computadores, porém, utilizam outros dialetos de BASIC, de modo que talvez você tenha que fazer algumas pequenas mudanças para que os programas funcionem no seu micro. Nestas duas páginas estão alguns pontos que você deve observar com atenção especial.

Os programas foram escritos para funcionar em micros de muitas marcas diferentes, de modo que não levam em conta as características particulares de nenhum micro em especial. Sabendo como os programas funcionam, porém, você poderá adaptá-los de modo a tirar partido das características especiais do seu micro.

Nomes das variáveis ►

Alguns computadores podem usar palavras como nomes de variáveis; outros só aceitam uma letra, ou uma letra e um número. Nos micros da família Sinclair, por exemplo, as variáveis numéricas podem ser palavras curtas, mas as variáveis string só podem ter uma letra. Nos programas deste livro, os nomes de muitas variáveis são palavras. Se o seu computador não aceita palavras, use apenas a primeira letra do nome da variável.



LET e THEN ►

Muitos computadores não precisam da palavra LET em uma instrução como LET FRUTAS$="UVA". Alguns micros também não precisam de THEN nas instruções IF...THEN. Todos os programas neste livro usam LET e THEN, mas você pode omitir essas palavras se o seu micro não precisa delas.

```
100 FRUTA$="UVA"
110 TOTAL=TOTAL+1
120 IF TOTAL=10 PRINT"TERMINEI"
```

Inicialização das variáveis ►

Em alguns micros, é preciso dar um valor concreto às variáveis antes de usá-las pela primeira vez. Em outros, o valor inicial é tomado como 0 para as variáveis numéricas e como uma string vazia para as variáveis string. Os programas deste livro incluem instruções para inicializar todas as variáveis, mas você pode omiti-las se não forem necessárias no seu micro.

```
10 LET A=0
20 LET FRASE$=""
.
.
.
.
100 LET FRASE$=FRASE$+"K"
110 LET A=A+1
```

Uma string sem nada é chamada de string vazia.

INPUT ►

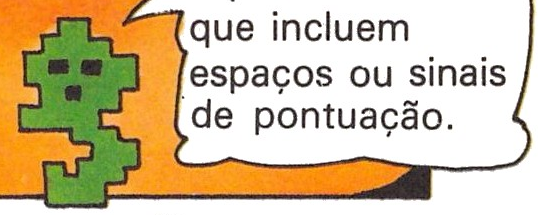
A maioria dos micros aceita palavras entre aspas em uma instrução INPUT. Alguns, porém, precisam de um ponto-e-vírgula antes do nome da variável e outros deixam automaticamente um espaço entre a mensagem e os dados a serem introduzidos. Para descobrir a forma apropriada para o seu micro, consulte o manual.

```
10 INPUT"QUAL E O SEU NOME";N$
20 PRINT"BOM DIA ";N$
```

Em alguns micros, é preciso deixar um espaço no final da mensagem.

DATA

```
100 DATA CAMUNDONGO,RATO,RATAZANA
110 DATA CORITIBA,"PATO BRANCO"
120 DATA "GATO,PRETO","RATO,BRANCO",GIRAFA
```



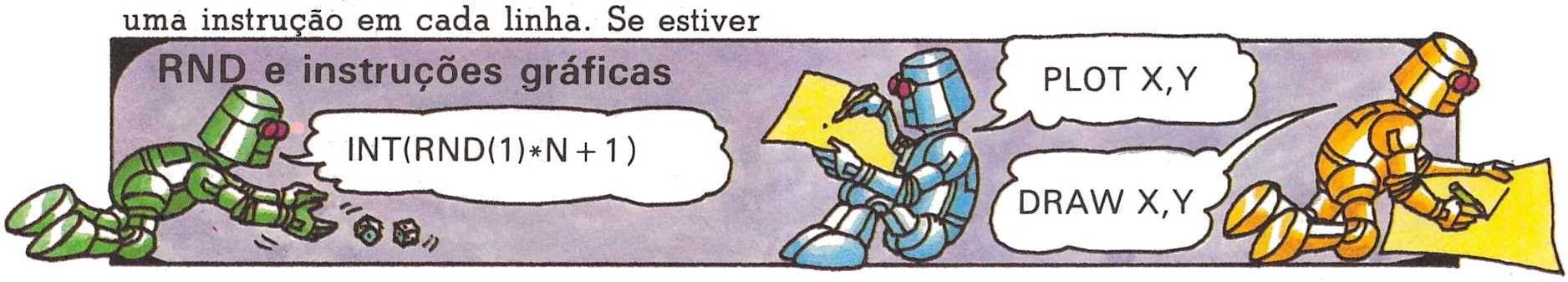
Seja especialmente cuidadoso com as linhas de dados. Os dados devem ser separados por vírgulas. Em alguns micros, todos os dados não-numéricos devem estar entre aspas. Em outros, as aspas só são necessárias se o dado inclui espaços ou sinais de pontuação.

Linhas com várias instruções

```
500 PLOT 40,1: DRAW 1,1
180 IF A=10 THEN PRINT "CERTO": GOTO 100
```

Muitos micros aceitam várias instruções na mesma linha, separadas por dois pontos. Este método gasta menos memória e pode tornar o programa mais legível. Se o seu micro não aceita linhas com várias instruções, coloque uma instrução em cada linha. Se estiver usando mais de uma instrução na mesma linha, cuidado ao colocar novas instruções na mesma linha que uma instrução IF...THEN, pois elas só serão executadas se a condição do IF for satisfeita.

RND e instruções gráficas



Estes comandos variam de micro para micro. Nos programas deste livro, a instrução para gerar um número aleatório entre 1 e N (onde N é um número qualquer) é INT(RND(1)*N+1).

Os comandos gráficos são PLOT X,Y para um ponto e DRAW X,Y para uma reta. Você deve substituir esses comandos pelos que se aplicam ao seu micro.

As strings nos micros da família Sinclair

Os micros da família Sinclair manipulam as strings de forma diferente. Não usam LEFT$, RIGHT$ ou MID$; é preciso indicar exatamente quais os caracteres escolhidos. Assim, por exemplo, em um micro da família Sinclair PRINT A$(1 TO 4) é a mesma coisa que PRINT LEFT$(A$,4) e PRINT A$(4 TO 8) é o mesmo que MID$(A$,4,5).

Nas matrizes string, todas as strings devem ter o mesmo número de caracteres (você pode completar as strings mais curtas com espaços) e cada caractere de uma string é armazenado em um compartimento separado da matriz. Alguns programas deste livro devem ser modificados para funcionarem em micros da família Sinclair; as modificações necessárias estão nas páginas 46 e 47.

Alterações nos programas

Depois que um programa estiver funcionando no seu micro, e você tiver uma boa idéia de como funciona, não tenha receio de modificá-lo e adaptá-lo, introduzindo dados diferentes ou complementando-o com instruções de cores e sons.

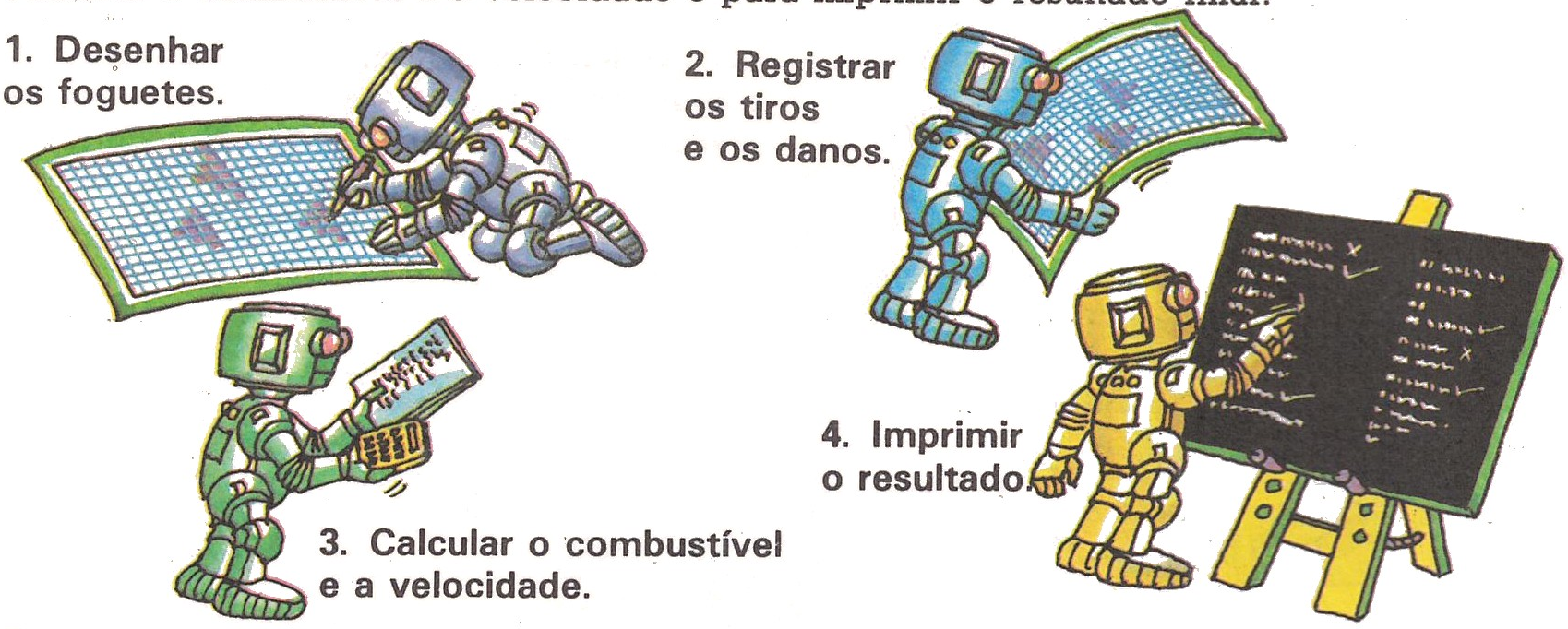
Quando estiver modificando um programa, verifique com atenção todas as linhas novas. Procure não usar um número excessivo de loops e não se esqueça de mudar as instruções DIM quando aumentar o número de elementos de uma matriz.

Como aprender BASIC analisando programas

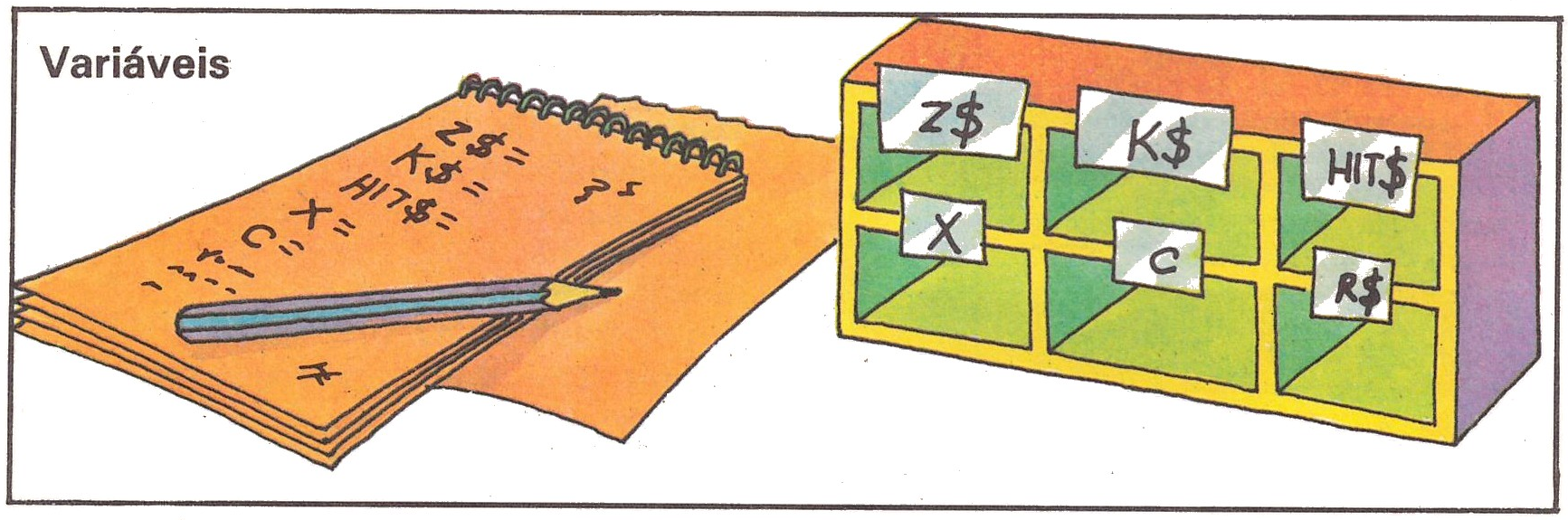
Uma boa maneira de aprender BASIC é analisar os programas escritos por outras pessoas para ver como funcionam. Estudando os programas deste livro, você aprenderá a usar loops e strings, a escrever programas gráficos e a ordenar e armazenar dados. À primeira vista, alguns dos programas parecem muito complicados. Entretanto, um programa complicado nada mais é que uma longa série de instruções de BASIC, arrumados de forma lógica. Nestas duas páginas você vai encontrar algumas dicas que o ajudarão a estudar e analisar programas.

Como estudar um programa

Quase todos os programas são feitos de várias partes diferentes (às vezes chamadas de rotinas ou módulos) que executam diferentes tarefas. Assim, por exemplo, em um jogo de batalha de foguetes, uma parte do programa é para desenhar os foguetes na tela e outras partes são para registrar os tiros e os danos, para verificar o combustível e a velocidade e para imprimir o resultado final.



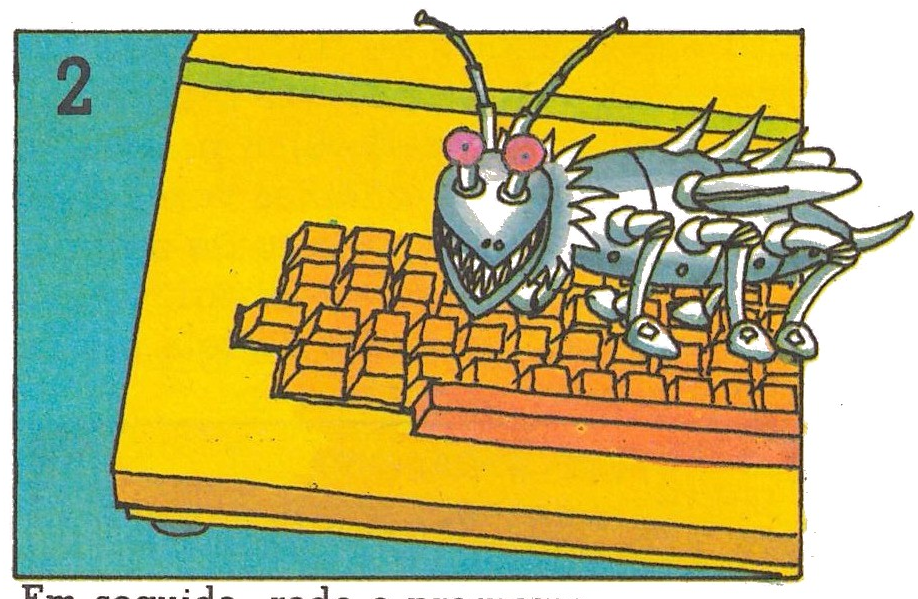
O primeiro passo para analisar um programa é reconhecer as diferentes partes e verificar para que servem. Assim você pode ter uma idéia geral de como o programa funciona. Preste atenção nas sub-rotinas e em grandes saltos na numeração das linhas; números de linha redondos muitas vezes indicam o início de um novo módulo do programa. Às vezes as diferentes partes do programa são indicadas por linhas de REM.



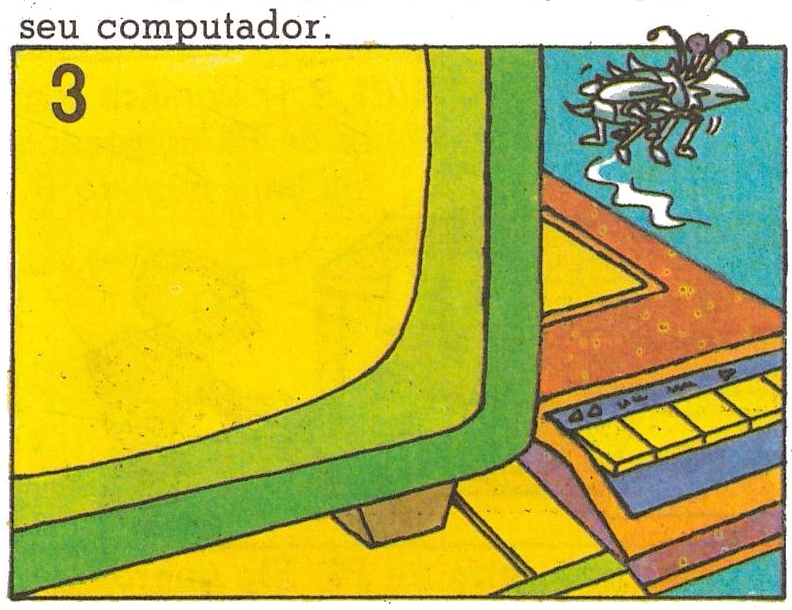
A coisa mais difícil de entender em um programa provavelmente são as variáveis. Antes de carregar um programa em um micro, você deve tentar compreender a função de cada variável. Certas variáveis são usadas quase sempre em determinadas funções. Assim, por exemplo, as letras I,J,K e L em geral são usadas em loops e Z e Z$ são usadas para guardar dados temporários.

Execução e correção de erros nos programas

Depois de verificar a função das variáveis, carregue o programa em um micro. Como os programas são escritos em BASIC padrão, talvez você tenha que adaptar algumas instruções para o seu computador.

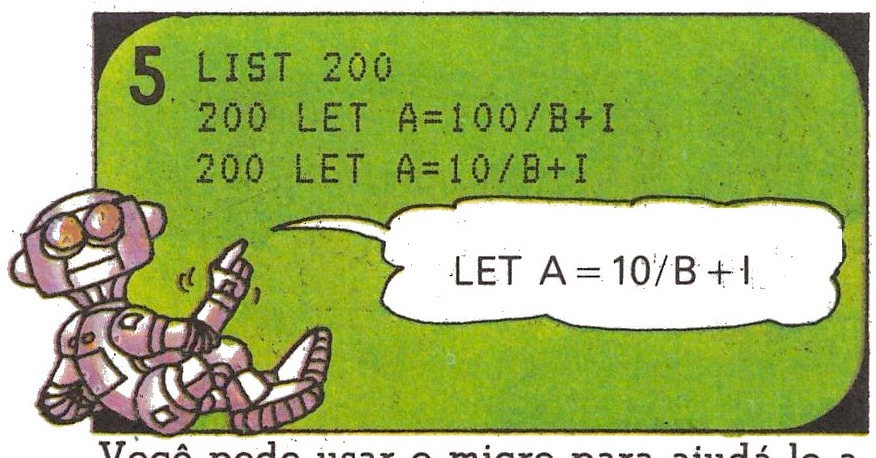


Em seguida, rode o programa. Provavelmente haverá alguns erros; para localizá-los, mande listar o programa na tela e procure erros de datilografia ou instruções que sejam incompatíveis com o seu micro.



Depois de corrigir todos os erros, rode o programa algumas vezes para ver como funciona. Se possível, armazene-o em cassete ou disquete para não ter que digitá-lo de novo.

Consulte então a listagem. Estude cada linha, tentando compreender sua função no programa. Preste atenção em rotinas curtas, que mais tarde talvez você possa incluir em seus próprios programas.



Você pode usar o micro para ajudá-lo a compreender como o programa funciona. Experimente mudar o valor de uma das variáveis e observe o efeito sobre o programa. Faça apenas uma mudança pequena de cada vez. Não se esqueça de colocar de volta a linha original depois que terminar suas observações.

Você também pode inserir instruções para imprimir os valores das variáveis, para ver como eles mudam durante o programa. Às vezes também é útil inserir instruções de STOP, para estudar o programa por partes. Em alguns micros, para continuar depois de um STOP basta escrever a instrução CONTINUE (CONT).

Uso de strings

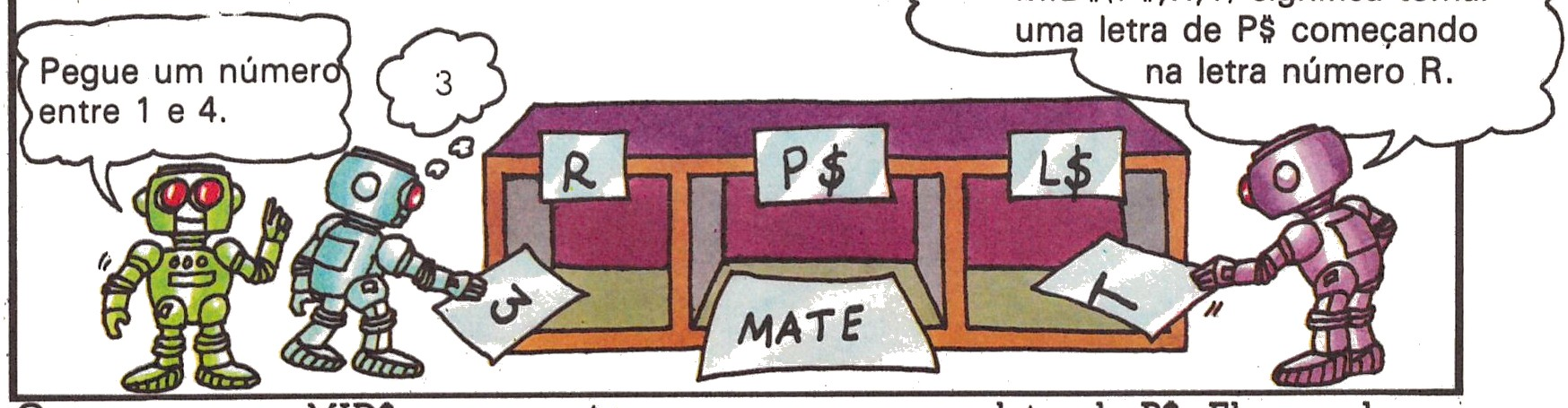
O programa da página seguinte mostra como é possível combinar instruções simples de BASIC para fazer com que o computador execute operações complicadas. O programa é um jogo de procurar palavras no qual o micro lhe pede uma palavra, coloca as letras aleatoriamente na tela e pergunta a você quantas vezes a palavra aparece. O programa usa as instruções de manipulação de strings MID$, RIGHT$ e LEN.*

O programa pode ser dividido em duas partes, a primeira para colocar as letras aleatoriamente na tela e a segunda para contar o número de vezes que a palavra apareceu.

```
PROCURE A PALAVRA
DIGA UMA PALAVRA PEQUENA
?MATE
VEJA SE ENCONTRA A PALAVRA
NO MEIO DAS LETRAS QUE VAO
APARECER NA TELA.
APERTE RETURN PARA COMECAR
```

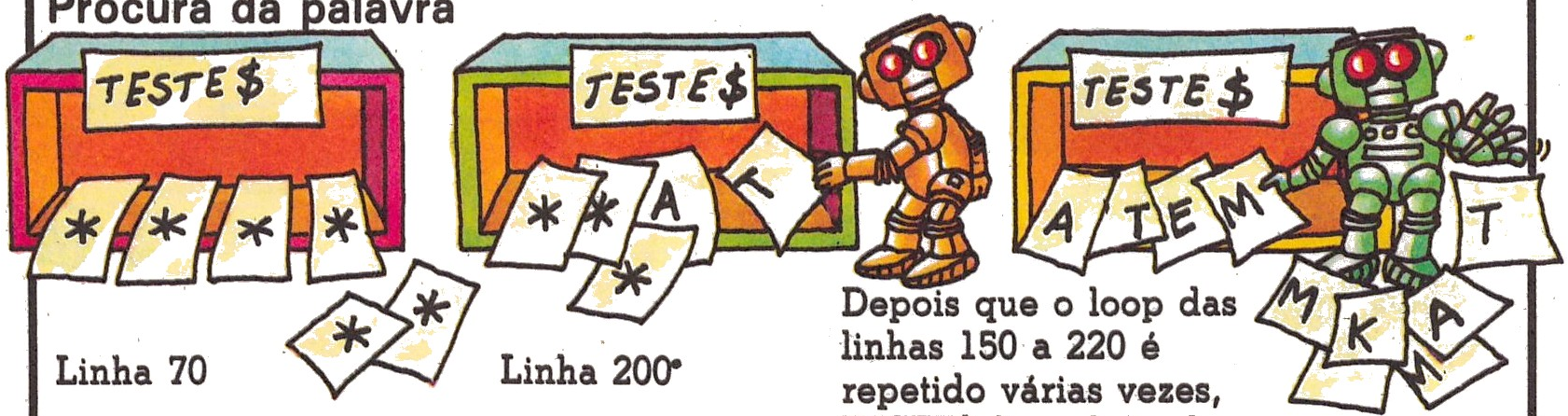
```
M T E T A E M A A T M T A E M
A T E M E A T M E A T M A T E
A M A E E M T T M M A E M A T
DIGA QUANTAS VEZES
A PALAVRA APARECEU NA TELA ?1

ERRADO!
A PALAVRA APARECEU 2 VEZES
```

Escolha das letras



O programa usa MID$ com um número aleatório para escolher a letra a ser mostrada. A sua palavra é guardada em P$. Na linha 160, o micro escolhe um número aleatório entre 1 e o tamanho da sua palavra e o guarda em R. Na linha 170, usa o número que está em R para separar uma letra de P$. Ele guarda a letra em L$ e a mostra na tela na linha 180. Cada vez que o loop de 150 a 220 é repetido, um novo número é guardado em R e uma nova letra de P$ é escolhida.

Procura da palavra



No início do programa, o computador reserva um espaço na memória chamado TESTE$ e o preenche com um número de asteriscos igual ao número de letras da palavra que você escolheu. Cada vez que escolhe uma nova letra, joga fora a primeira letra de TESTE$ e acrescenta a letra escolhida no final da string (linha 200). Em seguida, na linha 210, compara TESTE$ com P$ e se as strings são iguais soma 1 a N.

*Para adaptar o programa para micros da família Sinclair, vide página 46.

Jogo de procurar palavras

```
10 CLS
20 PRINT "PROCURE A PALAVRA":PRINT
30 LET TESTE$=""
40 LET N=0
45 PRINT "DIGA UMA PALAVRA PEQUENA"
50 INPUT P$
60 FOR I=1 TO LEN(P$)
70 LET TESTE$=TESTE$+"*"
80 NEXT I
90 PRINT
100 PRINT "VEJA SE ENCONTRA A PALAVRA"
110 PRINT "NO MEIO DAS LETRAS QUE VAO"
115 PRINT "APARECER NA TELA."
120 INPUT "APERTE RETURN PARA COMECAR";Z$
```

Use aqui a instrução do seu micro para limpar a tela.

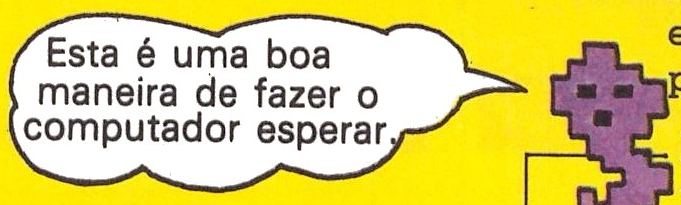
Nesta linha, os dois pontos são usados para separar duas instruções.

Define variáveis que serão usadas mais tarde.

Pede uma palavra e guarda em P$.

Este loop coloca em TESTE$ um número de asteriscos igual ao número de letras da palavra escolhida.

A linha 120 faz o computador esperar até você estar preparado para começar.

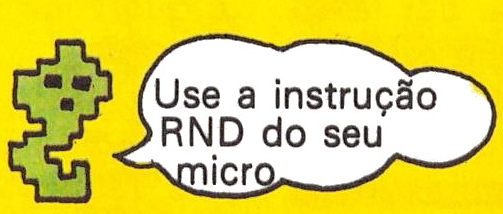



```
130 CLS
140 REM ESCOLHENDO LETRAS AO ACASO
150 FOR I=1 TO 50*LEN(P$)
160 LET R=INT(RND(1)*LEN(P$)+1)
170 LET L$=MID$(P$,R,1)
```

Inicia um loop que é repetido 50 vezes o número de letras da palavra escolhida.

Escolhe um número aleatório entre 1 e o número de letras da palavra e guarda em R.

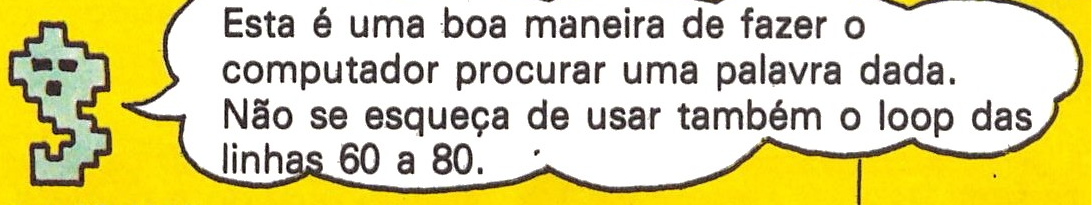
Usa o número que está em R para escolher uma letra de P$ e guarda essa letra em L$.




```
180 PRINT L$+" ";
190 REM PROCURANDO A PALAVRA
200 LET TESTE$=RIGHT$(TESTE$,LEN(P$)-1)+L$
210 IF TESTE$=P$ THEN LET N=N+1
220 NEXT I
```

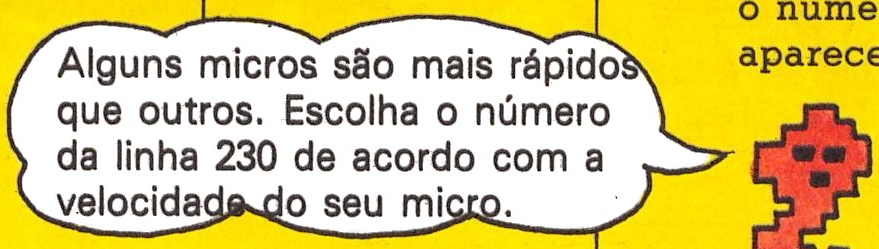
Imprime a letra que está em L$ seguida por um espaço. O ponto-e-vírgula faz com que o computador não mude de linha.

Toma LEN(P$) —1 letras da direita de TESTE$, acrescenta a letra que está em L$ e guarda o resultado de volta em TESTE$.




```
230 FOR I=1 TO 1000
240 REM NAO ESTA FAZENDO NADA
250 NEXT I
```

A variável N é usada para contar o número de vezes que a palavra apareceu.




```
260 CLS
265 PRINT "DIGA QUANTAS VEZES"
270 PRINT "A PALAVRA APARECEU NA TELA"
275 INPUT G
280 PRINT
290 IF G=N THEN PRINT "CERTO!"
300 IF G<>N THEN PRINT "ERRADO!"
305 IF N=1 THEN GOTO 320
310 PRINT "A PALAVRA APARECEU ";N;" VEZES"
315 STOP
320 PRINT "A PALAVRA APARECEU 1 VEZ"
```

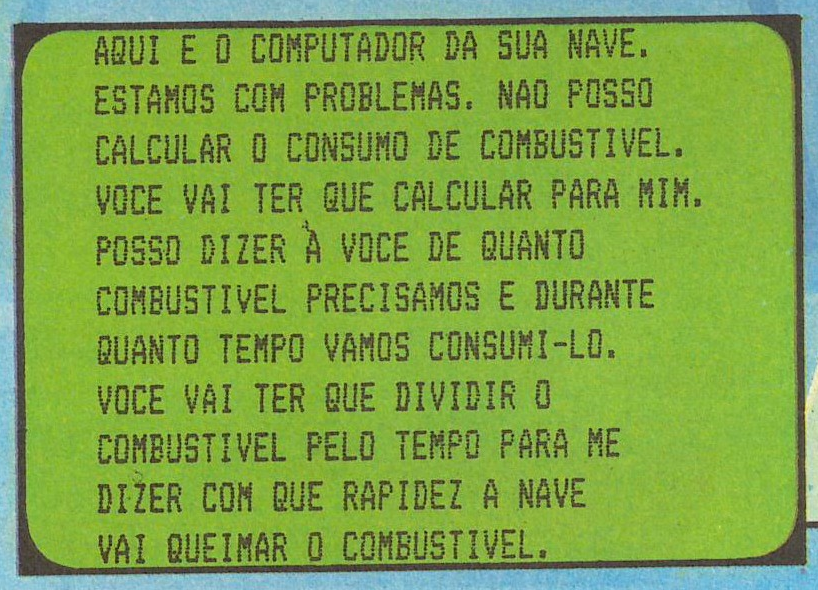
Este é um "loop de espera". O computador fica por alguns segundos sem fazer nada, a não ser aumentar o valor de I.

Guarda seu palpite em G.

Compara G com N, para verificar se sua resposta está correta.

Loops e números aleatórios

Este programa é um jogo espacial que também testa a sua capacidade de fazer cálculos de cabeça. Ele demonstra algumas das maneiras como é possível usar loops e números aleatórios e também alguns efeitos especiais na tela que você poderá incluir em seus programas.



Jogo da emergência espacial

```
100 CLS
110 FOR I=1 TO 20
115 PRINT "   *** CUIDADO ***"
118 PRINT "** ALERTA VERMELHO **"
120 FOR J=1 TO 10
125 REM NAO FAZ NADA
130 NEXT J
135 NEXT I
140 CLS
150 FOR I=1 TO 20
155 PRINT "*** CIRCUITOS DANIFICADOS ***"
160 FOR J=1 TO 10
165 REM NAO FAZ NADA
170 NEXT J
180 NEXT I
190 CLS
200 PRINT "AQUI E O COMPUTADOR DA SUA NAVE."
210 PRINT
220 PRINT "ESTAMOS COM PROBLEMAS. NAO POSSO"
225 PRINT "CALCULAR O CONSUMO DE COMBUSTIVEL."
230 PRINT
240 PRINT "VOCE VAI TER QUE CALCULAR PARA MIM."
250 PRINT
260 PRINT "POSSO DIZER A VOCE DE QUANTO"
265 PRINT "COMBUSTIVEL PRECISAMOS E DURANTE"
268 PRINT "QUANTO TEMPO VAMOS CONSUMI-LO.
270 PRINT
275 PRINT "APERTE QUALQUER TECLA"
276 A$=INKEY$:IF A$="" THEN GOTO 276
277 PRINT:PRINT:PRINT
280 PRINT "VOCE VAI TER QUE DIVIDIR O"
285 PRINT "COMBUSTIVEL PELO TEMPO PARA ME"
287 PRINT "DIZER COM QUE RAPIDEZ A NAVE"
289 PRINT "VAI QUEIMAR O COMBUSTIVEL."
290 PRINT
300 PRINT "EIS UM EXEMPLO"
310 PRINT "--------------"
320 PRINT "COMBUSTIVEL=24"
330 PRINT "TEMPO=6"
345 PRINT
350 PRINT "DIVIDA O COMBUSTIVEL PELO TEMPO"
355 PRINT "E ME DE A RESPOSTA DEPRESSA"
357 INPUT R
360 PRINT
370 IF R<>4 THEN PRINT "NAO. TENTE DE NOVO.":
GOTO 350
380 CLS
```

Mude os números nos loops de espera de acordo com o seu micro.

Use aqui a instrução do seu micro para limpar a tela.

Cada vez que o loop I é executado, mostrando na tela a mensagem das linhas 115 e 118, o loop J é executado dez vezes. O loop J é um loop de espera para dar tempo de que você leia o que está escrito na tela.

As linhas 150 a 180 funcionam como as linhas 110 a 135.

Da linha 200 à linha 345, o computador descreve como é o jogo.

A instrução PRINT sozinha deixa uma linha em branco.

Use o método da linha 276 quando o número de linhas de sua mensagem for maior que o que a tela do seu micro comporta.

Sublinha as palavras na linha acima.

Sua resposta é guardada em R.

A instrução GOTO só é executada se R não é 4. (Se o seu micro não permite mais de uma instrução na mesma linha, repita IF...THEN em outra linha com GOTO.)


```
390 PRINT "OK. AGORA VOCE NAO PODE ERRAR"
395 PRINT "OU A NAVE SERA DANIFICADA"
400 PRINT "MAIS DE DOIS ERROS"
405 PRINT "E VAMOS EXPLODIR"
410 PRINT
420 PRINT "APERTE RETURN QUANDO ESTIVER
    PRONTO"
425 INPUT Z$
430 CLS
440 PRINT "ATENCAO...LIGAR MOTORES!!!"
450 LET DANOS=0
460 FOR COMB=720 TO 120 STEP-120
470 LET T=INT(RND(1)*5+2)
480 PRINT
490 PRINT "COMBUSTIVEL=";COMB
500 PRINT "TEMPO=";T
510 PRINT
520 LET V=COMB/T
530 PRINT "DIGA A VELOCIDADE DE QUEIMA"
535 INPUT R
540 IF R=V THEN GOTO 600
550 LET DANOS=DANOS+1
560 PRINT
570 PRINT "*** DANOS ***"
580 PRINT
590 IF DANOS>2 THEN GOTO 640
600 NEXT COMB
610 CLS
620 PRINT "PARABENS...VOCE ME SUBSTITUIU"
625 PRINT "MUITO BEM. CHEGAMOS A TERRA"
627 PRINT "SAOS E SALVOS!!!"
630 GOTO 720
640 CLS
650 FOR I=1 TO 20
660 PRINT "*";
670 FOR J=1 TO INT(RND(1)*I+50)
680 PRINT "    ";
690 NEXT J
700 NEXT I
710 PRINT: PRINT "A NAVE EXPLODIU"
720 END
```

Espera você apertar RETURN.

Define uma variável chamada DANOS.

COMB é uma variável de contagem. Também é usada nos cálculos. COMB começa com o valor de 720 e diminui de 120 cada vez que o loop das linhas 460 a 600 é repetido. Esses números foram escolhidos para que o resultado da linha 520 seja sempre um número inteiro.

Escolhe um número entre 2 e 6, que divide exatamente o número que está em COMB.

Calcula o valor de COMB/T e guarda o resultado em V. Se a sua resposta está certa, vai para a linha 600.

A variável DANOS é usada para contar os seus erros.

Se você cometeu mais de dois erros, o computador sai do loop e pula para a linha 640.

Só imprime estas linhas se você cometeu menos de dois erros.

Estes loops imprimem estrelas ao acaso na tela. Cada vez que o loop I é repetido, o computador imprime uma estrela e em seguida o loop J imprime um número aleatório de espaços.

Base de dados para a Copa do Mundo

Base de dados é uma grande quantidade de informações armazenadas em um computador. Essas informações são organizadas de tal forma que o computador é capaz de combinar e comparar fatos e números de várias formas diferentes, o que permite ao usuário da base de dados obter informações úteis com facilidade e rapidez.

Nas próximas páginas você encontrará o programa de uma base de dados para a Copa do Mundo. Trata-se de um exemplo de uma pequena base de dados que pode ser usada para verificar que país ganhou a Copa do Mundo em qualquer ano a partir de 1930, ou em que ano um país em particular ganhou a Copa. No final do programa são apresentadas algumas idéias para adaptar o programa de base de dados para guardar outros tipos de informações, como o índice de uma coleção de revistas ou uma lista de pássaros.

Todo programa de base de dados pode ser dividido em três partes: uma para armazenar os dados, outra para recuperá-los e outra associada a um "menu". Menu é uma lista das várias coisas que um programa é capaz de fazer, com instruções para que você possa escolher uma das opções disponíveis.

Formato de saída do programa

```
DIGA O NOME DO TIME OU
ESCREVA MENU PARA VER
DE NOVO A LISTA ALEMANHA OCIDENTAL

ALEMANHA OCIDENTAL
GANHOU A COPA DO MUNDO EM 1954
FOI SEGUNDO LUGAR EM 1966
GANHOU A COPA DO MUNDO EM 1974
FOI SEGUNDO LUGAR EM 1982

APERTE RETURN PARA VER MENU
```

```
DIGA O ANO DA COPA OU
ESCREVA MENU PARA VER
DE NOVO A LISTA 1938

EM 1938
ITALIA GANHOU A COPA

APERTE RETURN PARA VER MENU
```

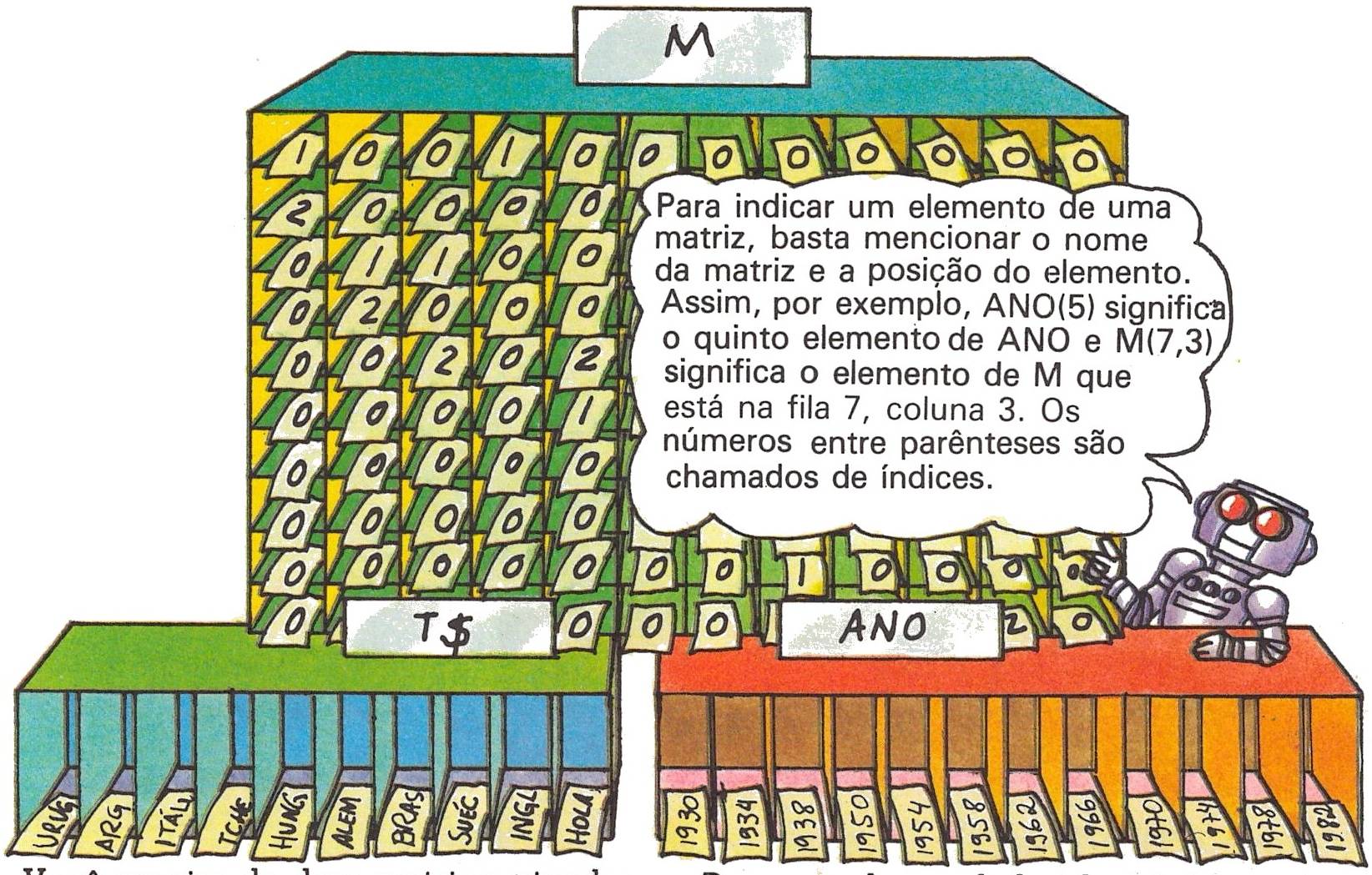
Armazenamento das informações

	1930	1934	1938	1950	1954	1958	1962	1966	1970	1974	1978	1982
URUGUAI	1	0	0	1	0	0	0	0	0	0	0	0
ARGENTINA	2	0	0	0	0	0	0	0	0	0	1	0
ITÁLIA	0	1	1	0	0	0	0	0	2	0	0	1
TCHECOSLOVÁQUIA	0	2	0	0	0	0	2	0	0	0	0	0
HUNGRIA	0	0	2	0	2	0	0	0	0	0	0	0
ALEMANHA OC	0	0	0	0	1	0	0	2	0	1	0	
BRASIL	0	0	0	0	0	1	1	0	1	0	0	0
SUÉCIA	0	0	0	0	0	2	0	0	0	0	0	0
INGLATERRA	0	0	0	0	0	0	0	1	0	0	0	0
HOLANDA	0	0	0	0	0	0	0	0	0	2	2	0

O programa usa uma "matriz" de informações e procura um time ou um ano da mesma maneira como você usaria a tabela acima. Na tabela, o 1 indica que uma seleção ganhou a Copa. e o 2 que tirou segundo lugar. Procurando a interseção entre uma linha e uma coluna, é possível verificar que país ganhou em que ano ou vice-versa. O programa realiza a busca automaticamente e com extrema rapidez.

Construção da matriz

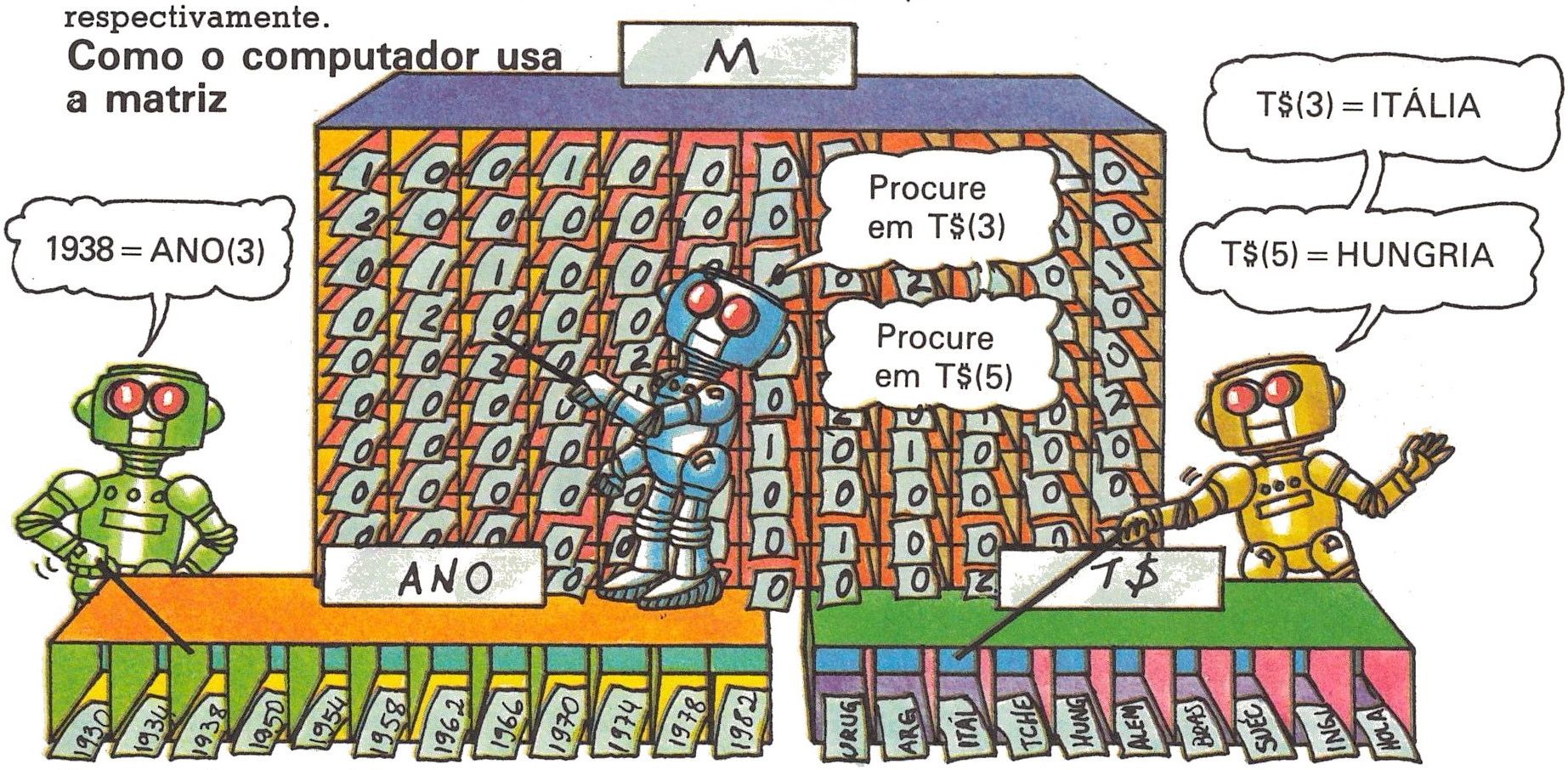
É muito fácil construir uma versão computadorizada da tabela da página anterior usando matrizes. Matriz é uma variável capaz de guardar vários dados diferentes.



Você precisa de duas matrizes simples, uma com 12 compartimentos para guardar a lista de anos e uma com 10 para guardar os nomes dos países. Elas são chamadas de ANO e T$, respectivamente.

Para guardar os dados da tabela, você precisa de uma matriz bidimensional com 10 linhas e 12 colunas. No programa, ela é chamada de M (inicial de matriz).

Como o computador usa a matriz

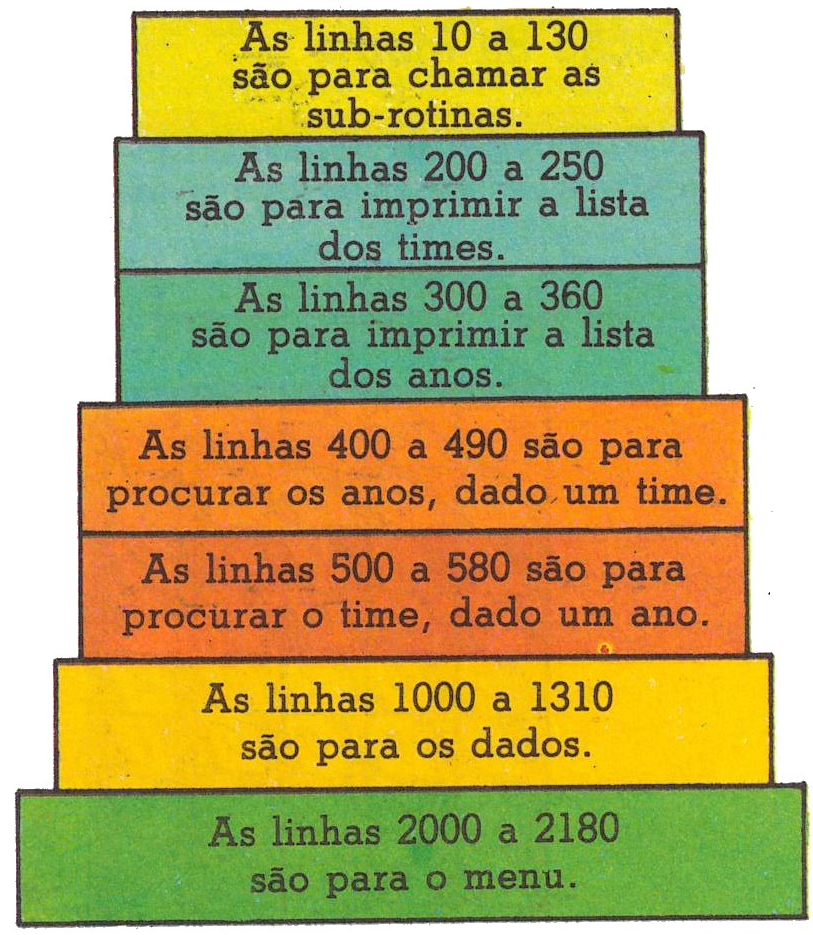


Para verificar qual foi o país que ganhou a Copa em 1938, por exemplo, o computador procura 1938 na matriz ANO e descobre que está no compartimento 3. Procura então na coluna 3 da matriz e quando encontra 1 ou 2 anota o número da linha e usa este número para procurar o nome do país em T$.

O programa base de dados

O programa tem sete partes principais, cada uma das quais correspondente a uma sub-rotina. As primeiras linhas dizem ao computador que sub-rotina deve executar; depois de terminar uma sub-rotina o computador sempre volta a essas linhas.

As sub-rotinas que começam nas linhas 200 e 300 são para imprimir as listas de times e de anos. As linhas 400 a 490 são para descobrir em que ano um dado time ganhou a Copa (ou chegou em segundo lugar), enquanto que as linhas 500 a 580 são para descobrir que time ganhou a Copa em um dado ano. Os dados estão a partir da linha 1000 e o menu a partir da linha 2000. Geralmente é melhor colocar os dados no final do programa.



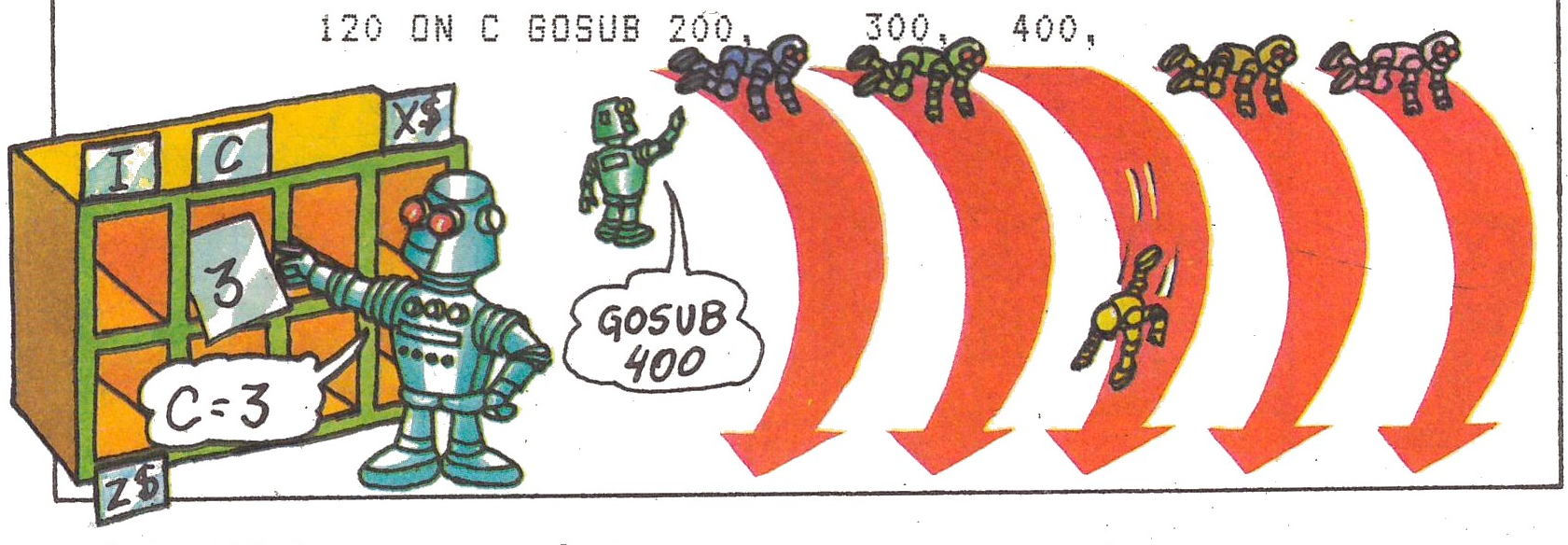
O menu

```
                MENU
                ****

PARA VER LISTA DE TIMES APERTE 1
PARA VER LISTA DE ANOS APERTE 2
PARA DIZER NOME DE TIME APERTE 3
PARA DIZER ANO APERTE 4
PARA TERMINAR APERTE 5
ESCREVA O NUMERO QUE ESCOLHEU
E DEPOIS APERTE RETURN
```

Menu é a parte do programa que diz o que ele é capaz de fazer e como escolher uma das opções disponíveis. Neste programa, você escolhe o que quer entrando com um número. O número é guardado na variável C e o computador usa este número para chamar a sub-rotina apropriada.

Chamada das sub-rotinas

```
120 ON C GOSUB 200,   300,   400,
```



A linha 120 do programa seleciona a sub-rotina a ser usada. A letra C é a variável que contém o número que você escolheu depois de ver o menu. O computador usa o número que está em C para escolher uma das sub-rotinas. Se C = 1, vai para a primeira sub-rotina da linha 120, isto é, a que começa na linha 200. Se C = 2, vai para a que começa em 300, e assim por diante. Se o seu micro não dispõe da instrução ON, utilize várias instruções IF...THEN, como IF C = 1 THEN GOSUB 200, etc.

Para que servem as variáveis

ANO	T$	M
Matriz unidimensional para guardar os anos.	Matriz unidimensional para guardar os nomes dos times.	Matriz bidimensional para guardar os dados.
C	Z$	X$
Variável para guardar a sua opção.	Nome do time ou ano que você escolheu.	Dados temporários.

O programa*

```
10 DIM T$(10): DIM ANO(12): DIM M(10,12)
```
Reserva espaço na memória para as matrizes.

```
100 GOSUB 1000: REM LER DADOS
```
Quando você roda o programa, a primeira coisa que o computador faz é ir para a sub-rotina que começa em 1000 para ler os dados.

```
110 GOSUB 2000: REM IMPRIMIR MENU
```
Em seguida, vai para 2000 para mostrar o menu na tela.


```
120 ON C GOSUB 200,300,400,500,600
```
Esta instrução manda o computador para a sub-rotina que executa a tarefa que você escolheu no menu. Depois, o computador volta para a linha 130 e é mandado de volta para 110 para mostrar de novo o menu.

```
130 GOTO 110

200 REM SUB-ROTINA PARA IMPRIMIR LISTA DOS TIMES

210 CLS
220 PRINT "LISTA DOS TIMES": PRINT "
```
Sublinha as palavras LISTA DOS TIMES.

```
230 FOR I=1 TO 10: PRINT T$(I):NEXT I
```
Loop para imprimir os nomes dos times. Cada vez que o loop é repetido, I aumenta de 1 e o computador imprime o nome seguinte de T$.

```
235 PRINT
240 INPUT "APERTE RETURN PARA VER MENU";X$
```
Esta instrução faz o computador esperar até que você aperte RETURN para passar para a linha seguinte.

```
250 RETURN
```


```
300 REM SUB-ROTINA PARA IMPRIMIR LISTA DOS ANOS
310 CLS
320 PRINT "LISTA DOS ANOS": PRINT "
330 FOR I=1 TO 12: PRINT ANO(I): NEXT I
```
Loop para mostrar os anos.

```
340 PRINT "NAO HOUVE COMPETICAO EM 1942 E 1946"
350 INPUT "APERTE RETURN PARA VER MENU";X$
360 RETURN
```

*Para converter este programa para os micros da família Sinclair, vide página 46.

```
400 REM SUB-ROTINA PARA DIZER O TIME
405 CLS


410 INPUT "DIGA O NOME DO TIME OU ESCREVA MENU PARA
    VER DE NOVO A LISTA ";Z$


415 IF Z$="MENU" THEN RETURN


420 FOR I=1 TO 10
425 IF Z$=T$(I) THEN GOTO 440
430 NEXT I


435 PRINT: PRINT "TIME DESCONHECIDO - TENTE DE NOVO,
    POR FAVOR": PRINT: GOTO 410


440 PRINT
445 PRINT Z$: PRINT
```

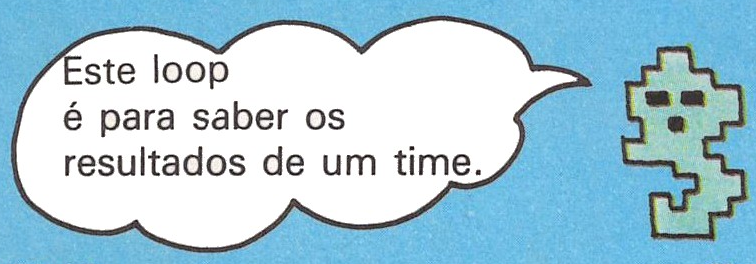
O computador guarda em Z$ o nome de um time ou a palavra "menu".

Se Z$ = MENU, o computador volta à linha 130 e de 130 vai a 110 para mostrar o menu.

Loop para comparar Z$ com todos os nomes que estão em T$. Quando os dois nomes são iguais, o computador pula para 440.

Esta linha é uma precaução para o caso em que você escrever errado o nome do país ou ele não constar da base de dados.

Imprime o nome do país.




```
450 FOR J=1 TO 12
455 IF M(I,J)=1 THEN PRINT "GANHOU A COPA DO MUNDO
EM ";ANO(J): PRINT
460 IF M(I,J)=2 THEN PRINT "FOI SEGUNDO LUGAR EM ";
ANO(J): PRINT
465 NEXT J


470 PRINT
480 INPUT "APERTE RETURN PARA VER MENU";X$


490 RETURN
```

Loop para o computador procurar na matriz o retrospecto do time que você escolheu. O valor de I é estabelecido pelo loop das linhas 420 a 430 e é o índice (número que mostra a posição na matriz) do seu time em T$. J é o número da coluna de M; cada vez que o loop é repetido, o computador procura em uma coluna diferente ao longo da linha I.

Igual à linha 240.

Volta para a linha 130.

```
500 REM SUB-ROTINA PARA DIZER O ANO
505 CLS


510 INPUT "DIGA O ANO DA COPA OU ESCREVA MENU PARA
VER DE NOVO A LISTA ":Z$


515 IF Z$="MENU" THEN RETURN


520 FOR I=1 TO 12
525 IF VAL(Z$)=ANO(I) THEN GOTO 540
530 NEXT I


535 PRINT: PRINT "ANO DESCONHECIDO - TENTE DE NOVO,
POR FAVOR": PRINT: GOTO 510
```

Igual à linha 410, mas desta vez o computador guarda um ano em Z$.

Loop para comparar Z$ com todos os anos que estão em ANO. Como não se pode comparar uma variável string com uma variável numérica, é preciso usar a instrução VAL, que diz ao computador para tratar os caracteres guardados em Z$ como um número.

```
540 PRINT: PRINT "EM ";Z$: PRINT


550 FOR J=1 TO 10
555 IF M(J,I)=1 THEN PRINT T$(J);" GANHOU A COPA"
560 NEXT J



565 PRINT
570 INPUT "APERTE RETURN PARA VER MENU";X$


580 RETURN
```

Este loop é parecido com o das linhas 450 a 465. Desta vez, o número da coluna é dado pelo índice do ano na matriz ANO e o número da linha muda cada vez que o loop é repetido.

Volta para a linha 130.

```
600 REM SUB-ROTINA PARA TERMINAR PROGRAMA
610 INPUT "TERMINOU? (S/N)";X$
620 IF X$<>"S" THEN RETURN ELSE END
```

Verifica se você quer parar. Se você escreve S, a instrução END diz ao computador para interromper a execução do programa. Se você escreve qualquer outra coisa, o computador volta à linha 130. A palavra ELSE é uma forma de acrescentar outra condição a uma instrução IF...THEN.

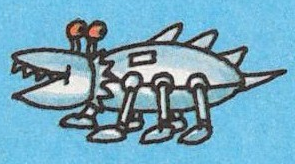
Se o seu micro não usa ELSE, coloque a instrução END na linha seguinte.

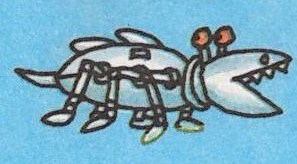
```
1000 FOR I=1 TO 12: READ ANO(I): NEXT I
1010 DATA 1930, 1934, 1938, 1950
1020 DATA 1954, 1958, 1962, 1966
1030 DATA 1970, 1974, 1978, 1982
```

Loop para guardar os dados em ANO.

```
1100 FOR I=1 TO 10: READ T$(I): NEXT I
1110 DATA URUGUAI, ARGENTINA
1115 DATA ITALIA, TCHECOSLOVAQUIA, HUNGRIA
1120 DATA ALEMANHA OCIDENTAL, BRASIL
1125 DATA SUECIA, INGLATERRA, HOLANDA
```

Loop para guardar os dados em T$.





Cuidado na hora de entrar com os dados. Se esquecer alguma vírgula ou número, o programa não funcionará.

```
1200 FOR I=1 TO 10: FOR J=1 TO 12
1205 READ M(I,J)
1210 NEXT J: NEXT I


1215 RETURN
```

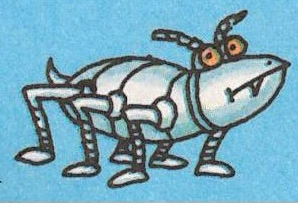
Loops para guardar dados na matriz bidimensional M.

Desta vez, volta para linha 110.

```
1220 DATA 1,0,0,1,0,0,0,0,0,0,0,0
1230 DATA 2,0,0,0,0,0,0,0,0,0,1,0
1240 DATA 0,1,1,0,0,0,0,0,2,0,0,1
1250 DATA 0,2,0,0,0,0,2,0,0,0,0,0
1260 DATA 0,0,2,0,2,0,0,0,0,0,0,0
1270 DATA 0,0,0,0,1,0,0,2,0,1,0,2
1280 DATA 0,0,0,0,0,1,1,0,1,0,0,0
1290 DATA 0,0,0,0,0,2,0,0,0,0,0,0
1300 DATA 0,0,0,0,0,0,0,1,0,0,0,0
1310 DATA 0,0,0,0,0,0,0,0,0,2,2,0
```

Estes são os dados para M.

Verifique se os dados estão corretos lendo várias vezes as linhas e colunas. Se algum dos números estiver errado, o computador obterá uma informação falsa quando consultar a matriz.



A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

```
2000 REM SUB-ROTINA PARA IMPRIMIR O MENU
2010 CLS

2020 PRINT "                MENU"
2030 PRINT "                ****"

2040 FOR I=1 TO 6: PRINT: NEXT I

2050 PRINT "PARA VER LISTA DE TIMES APERTE 1"
2060 PRINT
2070 PRINT "PARA VER LISTA DE ANOS APERTE 2"
2080 PRINT
2090 PRINT "PARA DIZER NOME DE TIME APERTE 3"
2100 PRINT
2110 PRINT "PARA DIZER ANO APERTE 4"
2120 PRINT
2130 PRINT "PARA TERMINAR APERTE 5"
2140 PRINT
2150 PRINT "ESCREVA O NUMERO QUE ESCOLHEU"

2160 INPUT "E DEPOIS APERTE RETURN ";C

2170 IF C<1 OR C>5 THEN PRINT "NUMEROS APENAS
ENTRE 1 E 5,POR FAVOR": GOTO 2150

2180 RETURN
```

Deixe uns 15 espaços para centralizar a palavra menu acima da lista de opções.

Loop para pular 6 linhas.

O número que você escolheu é guardado em C.

Esta linha é uma precaução para o caso de você escrever alguma coisa que não seja em número entre 1 e 5.

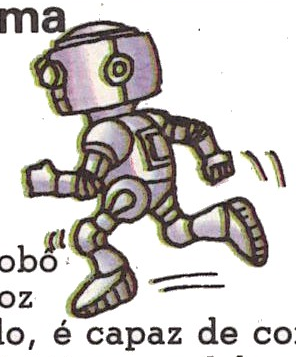
AND, OR e ELSE*

```
IF A=3 AND C$="SIM" THEN LET D=D+1
IF X<0 OR X>100 THEN PRINT "FORA DA TELA"
IF IDADE<18 THEN PRINT "MENOR" ELSE PRINT "MAIOR"
```

Você pode usar AND, OR e ELSE para acrescentar novas condições e opções a uma instrução IF...THEN, como nos exemplos acima. Quando você usa AND, o computador só executa a instrução que vem depois de THEN quando as duas condições são satisfeitas. OR diz ao computador para executar a instrução se pelo menos uma das condições for satisfeita. A palavra ELSE permite fornecer uma instrução para ser executada se a condição global não for satisfeita. Veja se é capaz de escrever um programa simples usando ELSE para resolver o problema ao lado.

*Nem todos os micros usam essas palavras de BASIC.

Problema



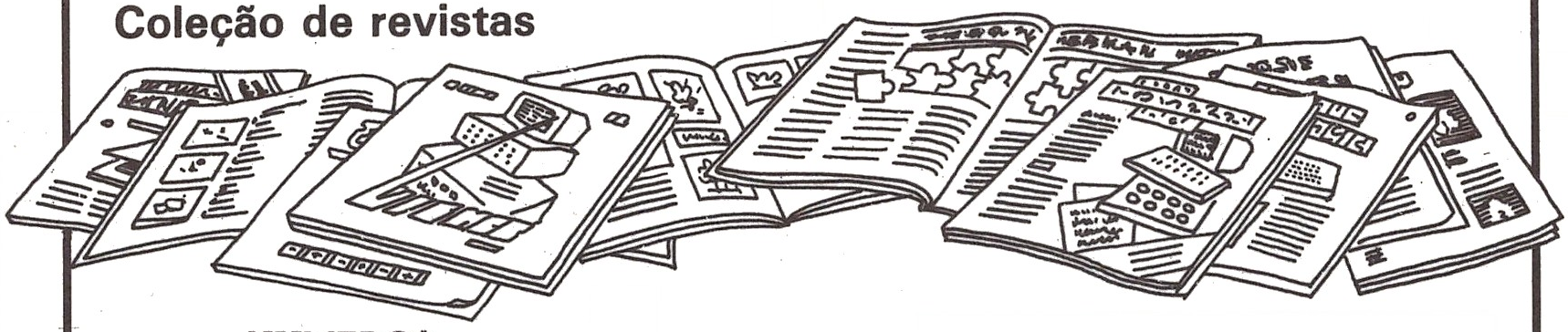
Zak, o robô mais veloz do mundo, é capaz de correr 100 quilômetros em 1 hora. Quando a temperatura é maior ou menor que 25°, porém, Zak corre mais depressa ou mais devagar à razão de 4 quilômetros por hora por grau de diferença. Escreva um programa para calcular a distância percorrida por Zak a uma dada temperatura e por um dado número de horas. (Resposta na página 48.)

Adaptação do programa de base de dados

Se você compreender bem como funciona o programa, poderá facilmente adaptá-lo para armazenar dados referentes a qualquer assunto. Mais abaixo aparecem três sugestões.

Depois que escolher o assunto, prepare uma tabela com os dados, como a da página 18. A tabela pode ter um número diferente de linhas e colunas, mas nesse caso você terá que mudar o tamanho das matrizes do programa. Coloque os dados nas linhas de DATA e reescreva as opções do menu. Não se esqueça de mudar as instruções DIM e o número de vezes que os loops de leitura de dados são executados.

Coleção de revistas

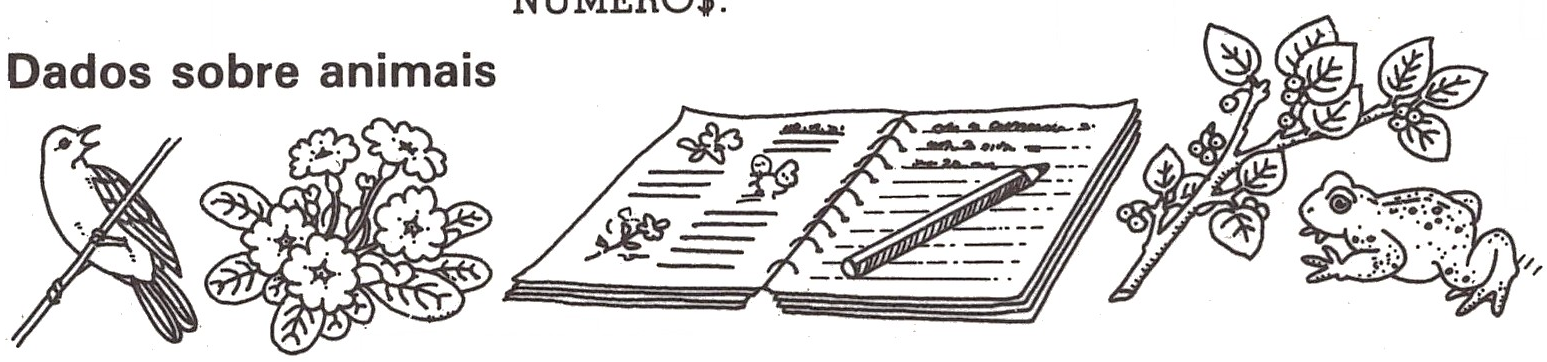


NUMERO$

ARTIGO$ | MATRIZ

Você pode usar uma base de dados para ver em que número de uma revista foi publicado um dado artigo ou quais os artigos publicados em um dado número de revista. Vai precisar, além da matriz de dados, de uma matriz chamada ARTIGO$ e de outra chamada NUMERO$.

Dados sobre animais

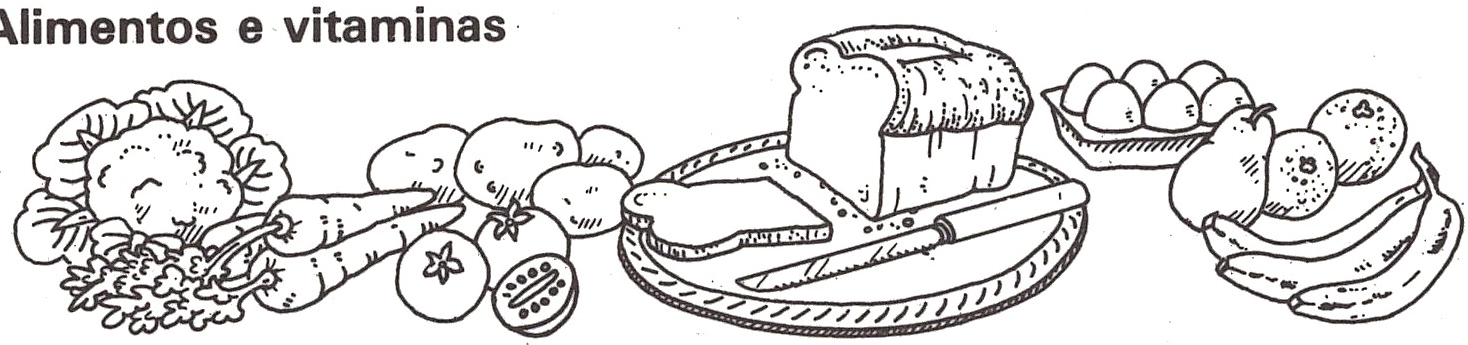


ANIMAL$

LUGAR$ | MATRIZ

Você pode organizar uma base de dados para saber quando ou onde viu um determinado animal ou planta. Vai precisar de uma matriz para os nomes dos animais ou plantas e outra para os lugares ou datas.

Alimentos e vitaminas



ALI$

VITA$ | MATRIZ

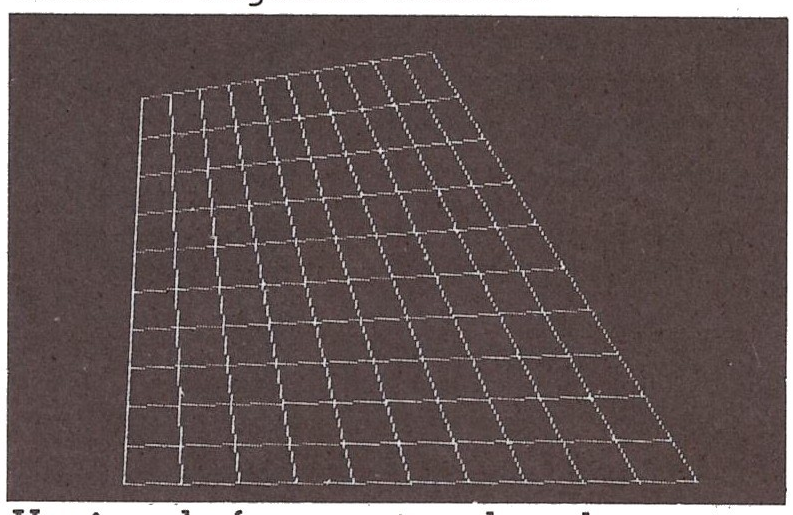
Esta base de dados pode ser usada para verificar quais os alimentos que contêm uma dada vitamina ou quais as vitaminas que estão presentes em um dado alimento. Você também pode organizar uma base de dados de alimentos e calorias, para saber quantas calorias possui um dado alimento ou quais os alimentos que possuem um dado número de calorias.

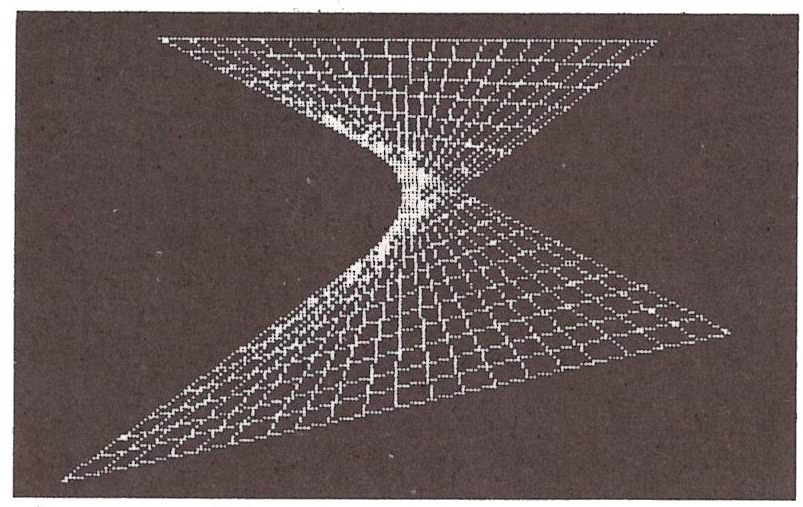
Programa para desenhar

Este programa desenha uma forma simples na tela e a preenche com uma rede de linhas retas. As redes são muito usadas nos desenhos feitos por computador para dar uma impressão de relevo.

O programa usa as instruções gráficas PLOT X,Y para marcar um ponto e DRAW X,Y para traçar uma reta. As coordenadas X e Y são tomadas a partir das margens da tela. Você talvez tenha que substituir essas instruções pelas que se aplicam ao seu micro.*

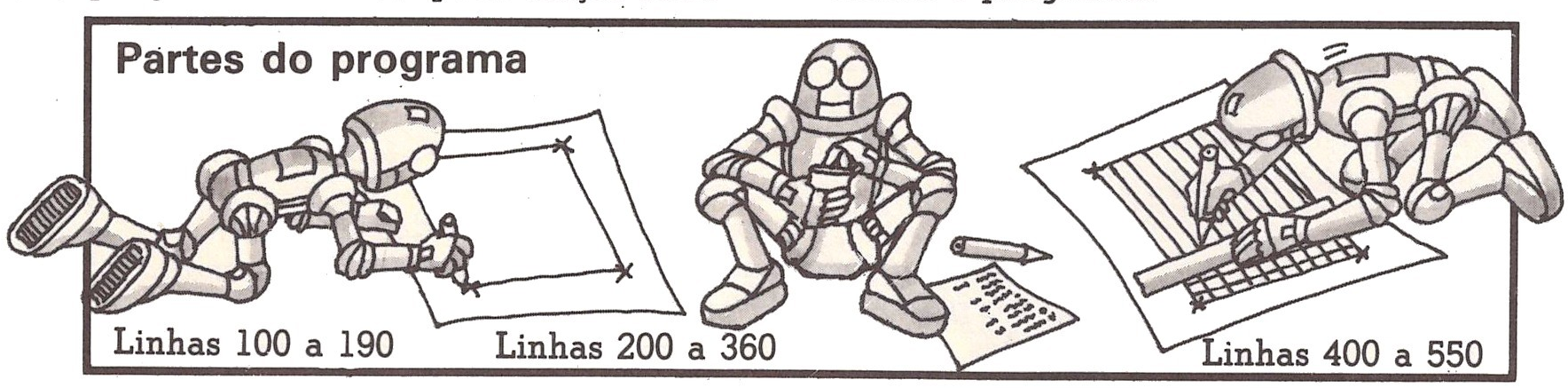
Existem duas formas diferentes de escrever programas para desenhar na tela. Você pode dizer ao computador para calcular e marcar cada ponto de uma vez, construindo o desenho gradualmente na tela, ou pode fazer primeiro todos os cálculos, guardar os resultados em matrizes e no final fazer o desenho de forma quase instantânea. O programa que se segue utiliza o segundo método.





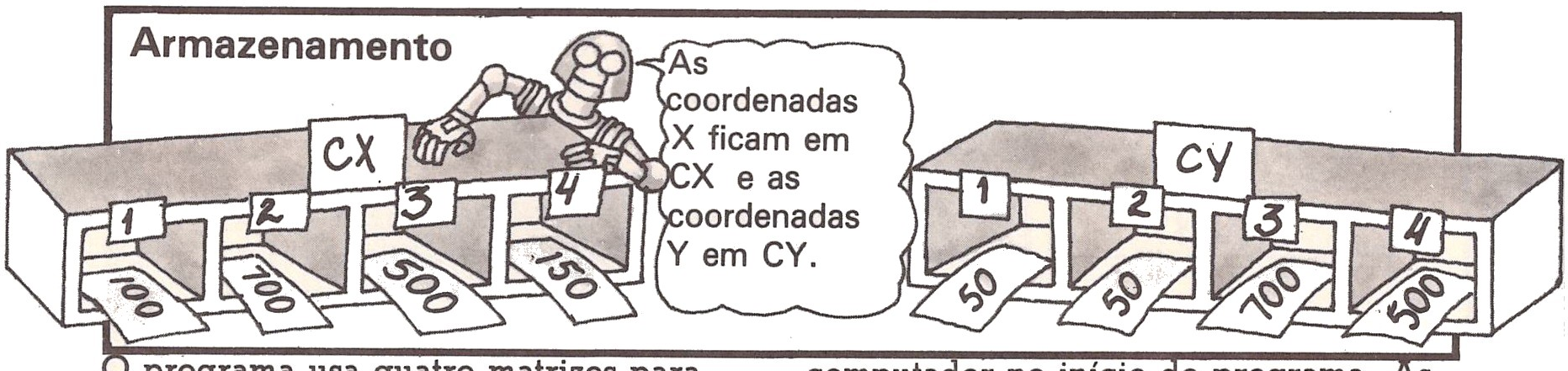
Você pode fazer muitos desenhos diferentes mudando os dados do programa. Também pode traçar retas de cores diferentes. No final da listagem aparecem algumas dicas para alterar o programa.

Partes do programa



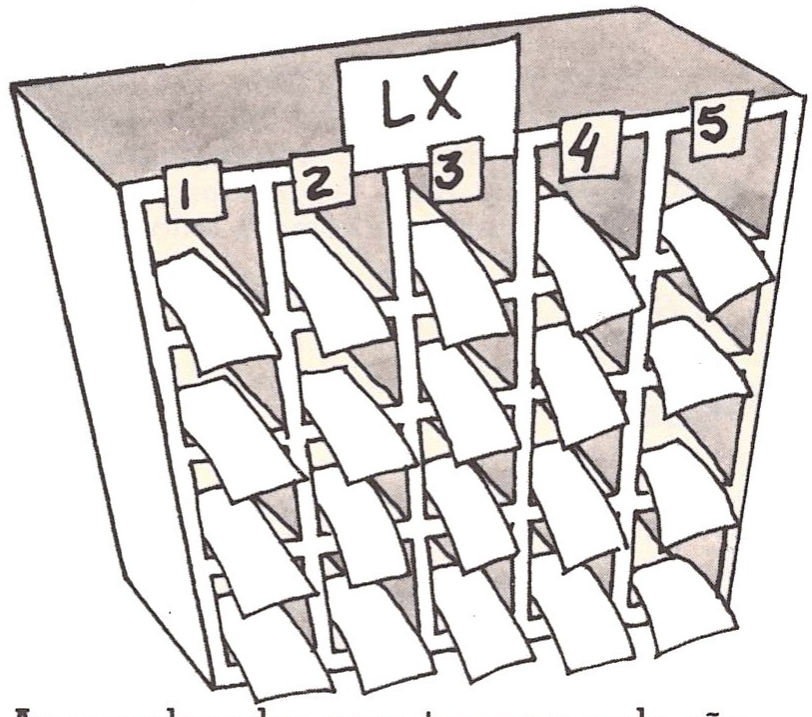
O programa tem três partes principais. A primeira parte (linhas 100 a 190) marca os cantos do desenho e os liga por linhas retas. A segunda parte (linhas 200 a 360) calcula as coordenadas para traçar as retas da rede e a terceira parte (linhas 400 a 550) traça a rede.

Armazenamento

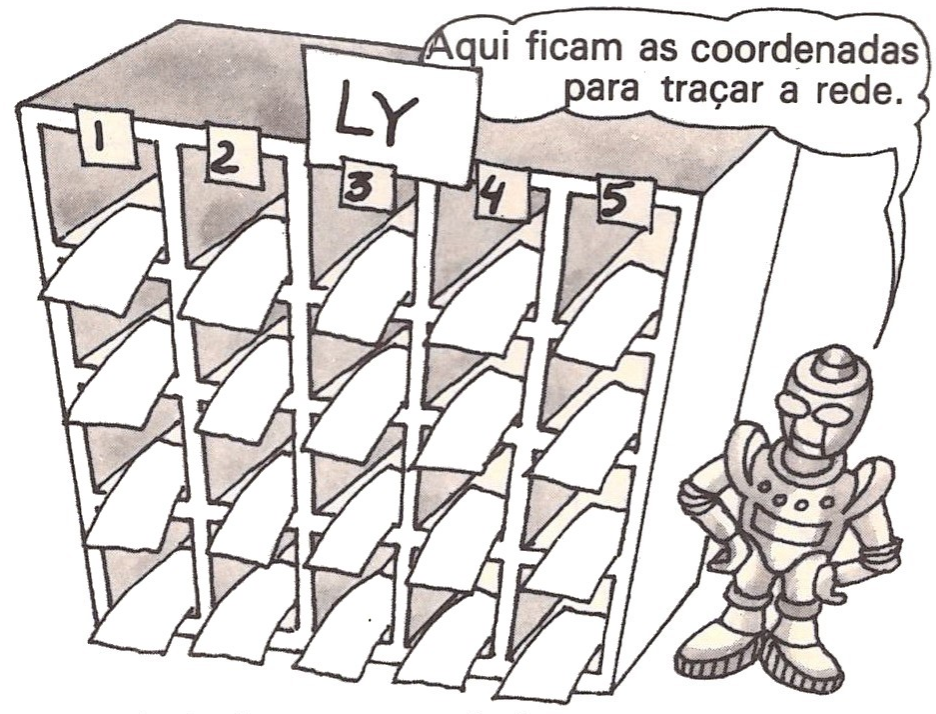


O programa usa quatro matrizes para guardar todos os dados. As coordenadas X e Y dos quatro cantos da rede são guardadas em CX e CY. Você fornece esses dados ao computador no início do programa. As coordenadas do primeiro canto da rede ficam guardadas em CX(1) e CY(1), as do segundo em CX(2) e CY(2) e assim por diante.

*Em alguns micros, como o TK-85, as coordenadas X e Y para traçar uma reta são medidas a partir do último ponto marcado. Para converter o programa para esses micros, vide página 47.



As coordenadas para traçar a rede são guardadas nas matrizes LX e LY. Ambas são matrizes bidimensionais com quatro linhas. O número de colunas depende do número de retas desejado.



Em cada linha são guardadas as coordenadas para as retas de um dos lados do desenho. O computador calcula os valores das coordenadas e os armazena em LX e LY.

O programa

```
100 INPUT "NUMERO DE RETAS ";N

110 DIM CX(4),CY(4)
115 DIM LX(4,N),LY(4,N)
120 CLS
130 REM DESENHA MOLDURA
140 FOR I=1 TO 4
145 READ CX(I),CY(I)
150 NEXT I

160 DATA 100,50,700,50,500,700,150,500

170 PLOT CX(4),CY(4)
180 FOR I=1 TO 4
185 DRAW CX(I),CY(I)
190 NEXT I
```

Use 20 nos micros com gráficos de alta resolução e 5 nos de baixa resolução.

Informa ao computador qual o tamanho das matrizes.

Loop para guardar os dados dos cantos em CX e CY. Cada vez que o loop é repetido, dois números da linha 160 são guardados em CX e CY.

Os números que aparecem aqui são para o desenho da esquerda da página anterior. Mude-os para obter desenhos diferentes.

Estas são as coordenadas dos cantos. Você pode ter que mudar os números para que a figura caiba na tela do seu micro.

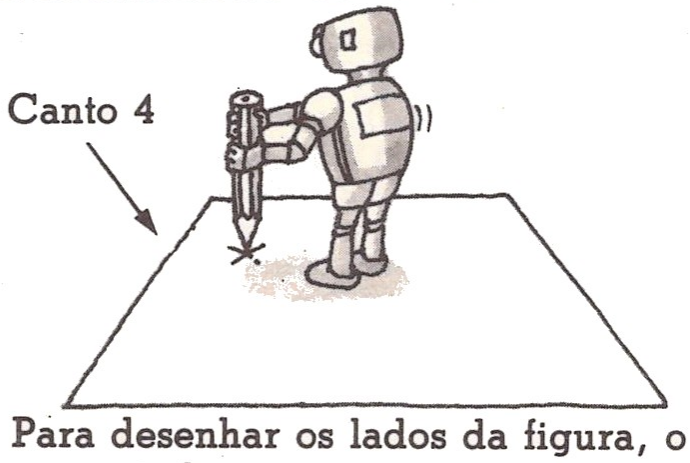
Use as instruções equivalentes a PLOT e DRAW no seu micro.

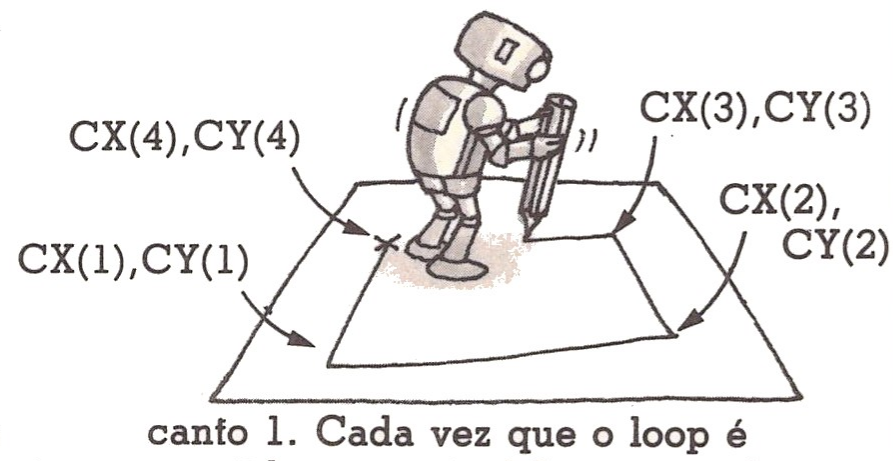
Marca o canto 4 usando os números guardados em CX(4) e CY(4).

Loop para desenhar os lados da figura.

A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

Desenhando os lados

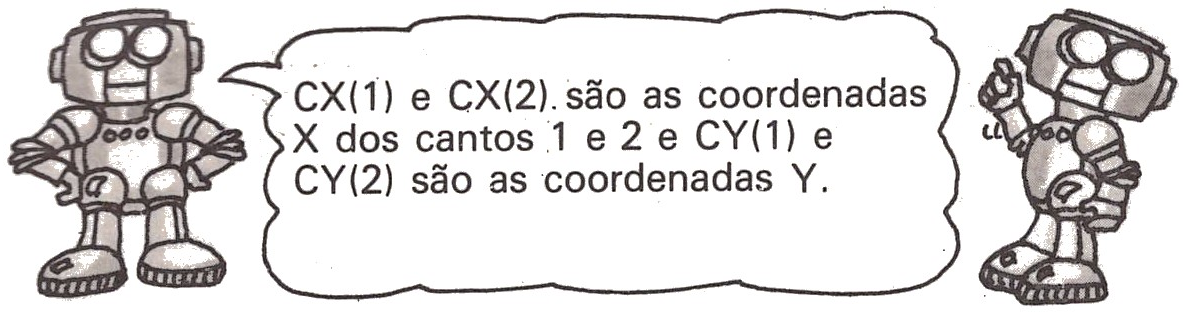




Para desenhar os lados da figura, o computador marca primeiro o canto 4 (linha 170). Depois, o loop das linhas 180 a 190 o faz traçar uma reta até o canto 1. Cada vez que o loop é repetido, a variável I aumenta de uma unidade e o computador traça uma reta até o canto seguinte.

```
200 REM CALCULA CÓORDENADAS
```

A parte seguinte do programa é composta por quatro loops para calcular as coordenadas das retas da rede. As figuras abaixo mostram como é feito o cálculo.




```
210 FOR I=1 TO N
```

N é o número de retas escolhido.

```
220 LET LX(1,I)=CX(1)+(CX(2)-CX(1))*I/N
```

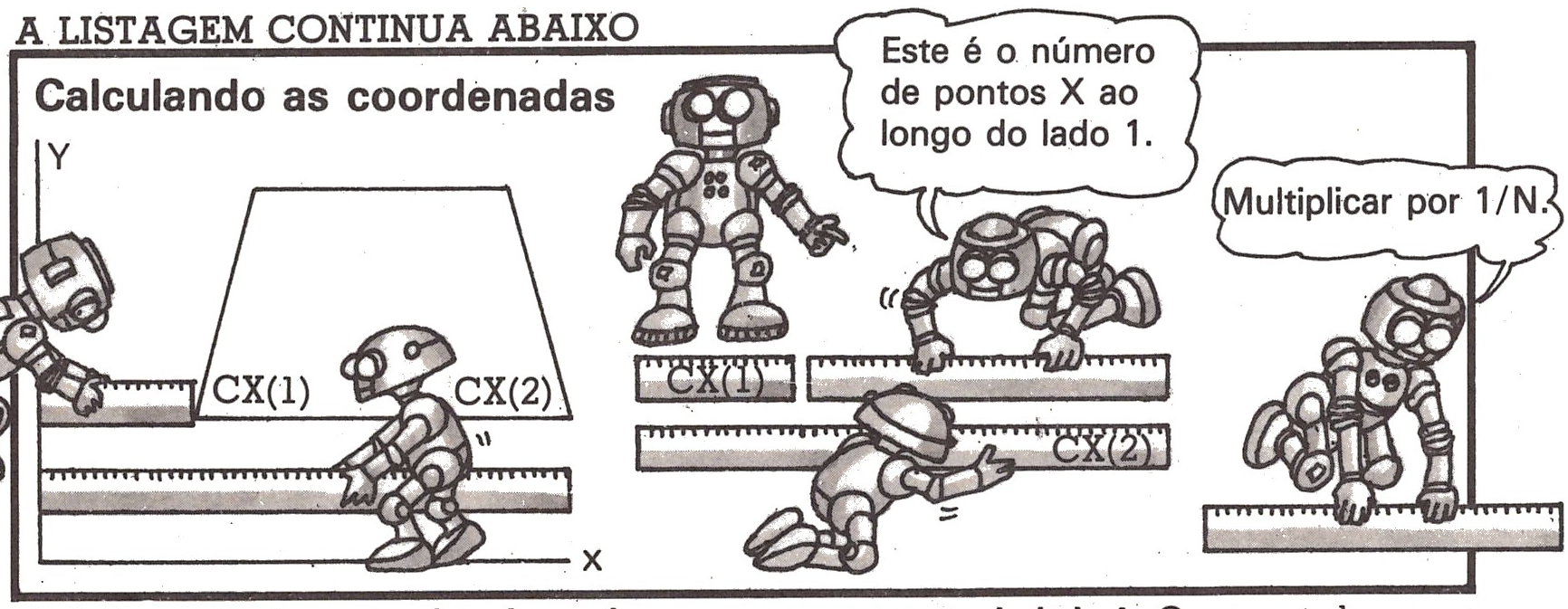
Calcula as coordenadas X das retas ao longo do lado 1 e as guarda em LX, linha 1, colunas 1 a N.

```
230 LET LY(1,I)=CY(1)+(CY(2)-CY(1))*I/N
```

Calcula as coordenadas Y para o lado 1 e as guarda em LY, linha 1, colunas 1 a N.

```
240 NEXT I
```

A LISTAGEM CONTINUA ABAIXO



Cada loop calcula as coordenadas ao longo de um dos lados do desenho. Assim, por exemplo, as linhas 210 a 240 calculam as coordenadas para o lado 1. Na linha 220, o computador subtrai CX(1) de CX(2). O resultado é o número de pontos X ao longo do lado 1. Na primeira passagem pelo loop, este número é multiplicado por 1/N (N é o número de retas que você escolheu). Assim, por exemplo, se N é 5 o resultado é 1/5 do comprimento do lado 1. O computador soma este número a CX(1) e guarda o resultado em LX(1,1). Na segunda passagem pelo loop, I=2, de modo que ele multiplica por 2/5 e guarda o resultado em LX(1,2). O computador faz isso para todos os valores de I de 1 a N, para calcular as coordenadas X de todos os pontos da rede ao longo do lado 1. A linha 230 usa o mesmo método para calcular as coordenadas Y.

```
250 FOR I=1 TO N
260 LET LX(3,I)=CX(4)+(CX(3)-CX(4))*I/N
270 LET LY(3,I)=CY(4)+(CY(3)-CY(4))*I/N
280 NEXT I
```

Loop para calcular as coordenadas para o lado 3. Para guardá-las na mesma ordem que para o lado 1 (da esquerda para a direita), o computador tem que fazer a soma na ordem oposta. Ele subtrai as coordenadas do canto 4 das do canto 3 e soma o resultado às coordenadas do canto 4.

Experimente mudar estas linhas para ficarem iguais às dos outros lados e veja o que acontece.

```
290 FOR I=1 TO N
300 LET LX(2,I)=CX(2)+(CX(3)-CX(2))*I/N
310 LET LY(2,I)=CY(2)+(CY(3)-CY(2))*I/N
320 NEXT I
```

Loop para calcular as coordenadas para o lado 2.

A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

```
330 FOR I=1 TO N
340 LET LX(4,I)=CX(1)+(CX(4)-CX(1))*I/N
350 LET LY(4,I)=CY(1)+(CY(4)-CY(1))*I/N
360 NEXT I

400 REM DESENHAR RETAS
410 LET FILA=1: GOSUB 500
420 LET FILA=2: GOSUB 500

430 STOP
500 REM SUB-ROTINA
510 FOR I=1 TO N

520 PLOT LX(FILA,I),LY(FILA,I)
530 PLOT LX(FILA+2,I),LY(FILA+2,I)
540 NEXT I

550 RETURN
```

Loop para calcular as coordenadas para o lado 4.

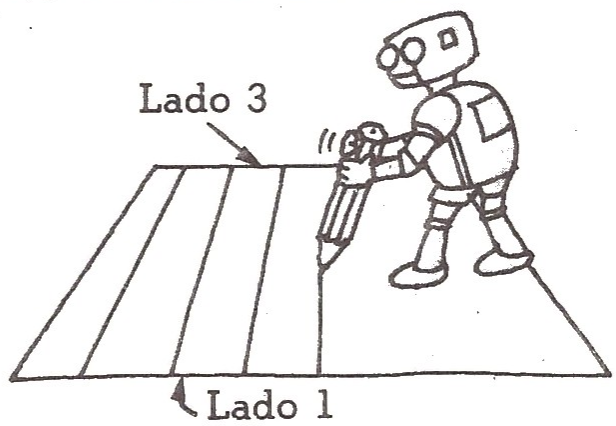
Esta linha define uma variável chamada **FILA**, que recebe o valor 1. Em seguida, o computador vai para a sub-rotina que começa em 500. Depois de passar pela sub-rotina, volta para a linha 420, muda **FILA** para 2 e vai de novo para a sub-rotina.

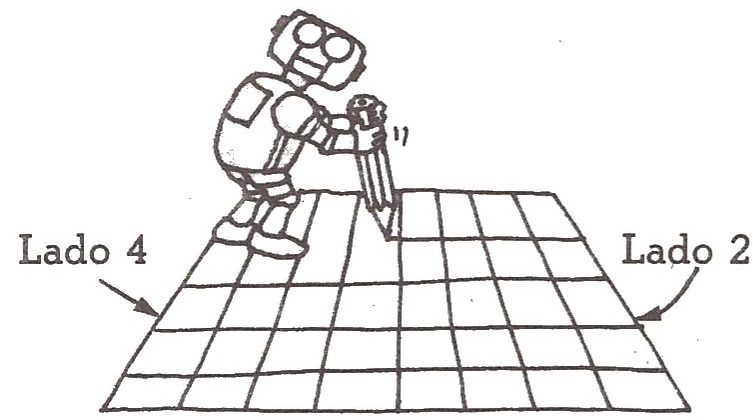
O programa é interrompido depois de executar duas vezes a sub-rotina.

FILA e **I** são os índices de **LX** e **LY**. Essas duas variáveis dizem ao computador em que compartimento deve procurar as coordenadas **X** e **Y** para cada reta da rede. **FILA** é o número da linha e **I** o da coluna. Na linha 1 estão as coordenadas do lado 1, na linha 2 as do lado 2, etc.

Da primeira vez, volta para a linha 420; da segunda, para 430.

Traçando as retas da rede





Na primeira vez em que a sub-rotina é executada, **FILA** = 1, de modo que na linha 520 o computador marca um ponto no lado 1. Na linha 530, ele soma 2 a **FILA** e portanto traça uma reta até o lado 3. Na segunda vez, **FILA** = 2 e portanto ele marca pontos no lado 2 e traça retas até o lado 4.

Idéias para modificar o programa

1. Para mudar o desenho, faça primeiro um esboço no papel da forma desejada. O primeiro par de números da linha 160 são as coordenadas do canto 1, o segundo par do canto 2, etc. Se você usar dois cantos iguais, terá um triângulo. Experimente também fazer os lados se cruzarem, como no desenho da direita da página 26.

2. Você pode usar a instrução **INPUT** dentro de um loop para entrar com os dados no computador. Substitua as linhas 140 a 160 pelas linhas seguintes:

```
140 FOR I=1 TO 4
150 PRINT "DIGA AS COORDENADAS DO VERTICE ";I
155 INPUT CX(I),CY(I)
160 NEXT I
```

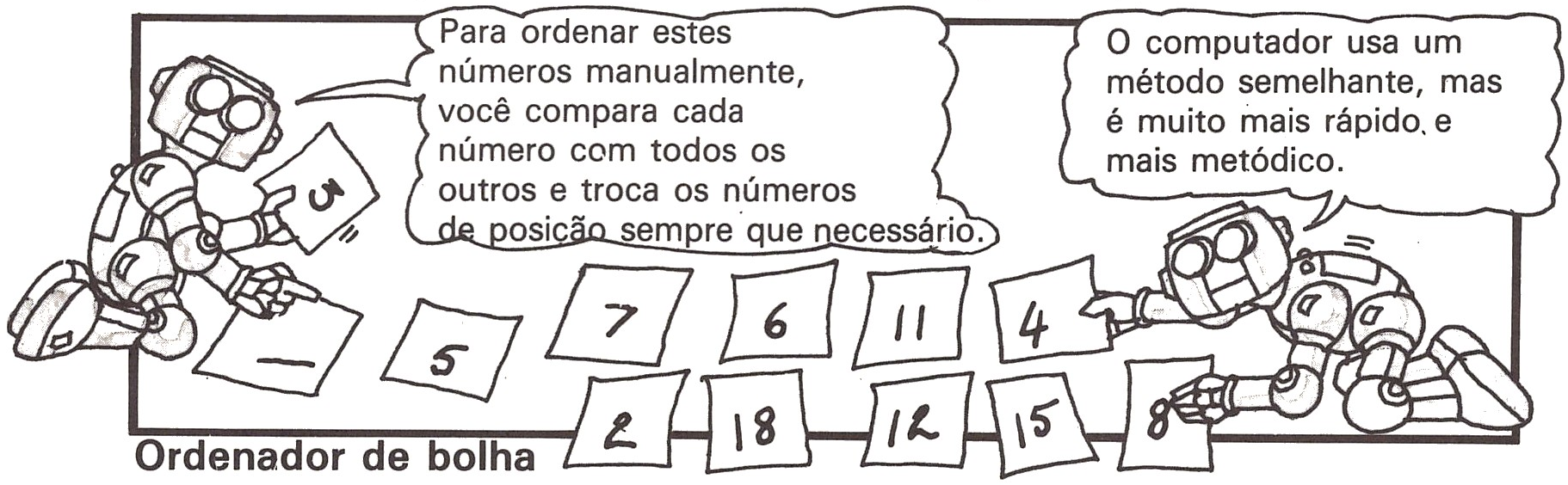
3. Para traçar linhas coloridas, acrescente a instrução de cor do seu micro às linhas 410 e 420, antes de GOSUB. Não se esqueça de colocar dois pontos para separar a instrução de cor do GOSUB.

Programas para ordenar dados

Às vezes surge a necessidade de colocar uma série de dados em ordem numérica ou alfabética. Isso pode acontecer, por exemplo, se você estiver preparando o índice de um livro ou analisando um conjunto de medidas. As listas curtas são fáceis de ordenar manualmente, mas no caso de uma grande quantidade de dados o computador é muito mais rápido, além de não cometer erros.

Os programas especiais para ordenar dados são chamados de "ordenadores". Existem muitos programas ordenadores escritos em BASIC; pode ser que você já tenha visto alguns nas revistas especializadas. Esses programas usam diferentes técnicas de programação e se aplicam a tipos de problemas ligeiramente diferentes.

Nas próximas páginas, vamos analisar dois programas de ordenação. O primeiro é chamado de "ordenador de bolha" (daqui a pouco explicaremos por que) e o segundo de ordenador de Shell (em homenagem ao seu criador). O ordenador de bolha é um dos tipos mais lentos e só costuma ser usado quando o número de dados é muito pequeno. O ordenador de Shell é muito mais rápido. Na página 35 você verá que existe uma maneira simples de comparar as velocidades dos dois tipos de ordenadores.



Ordenador de bolha

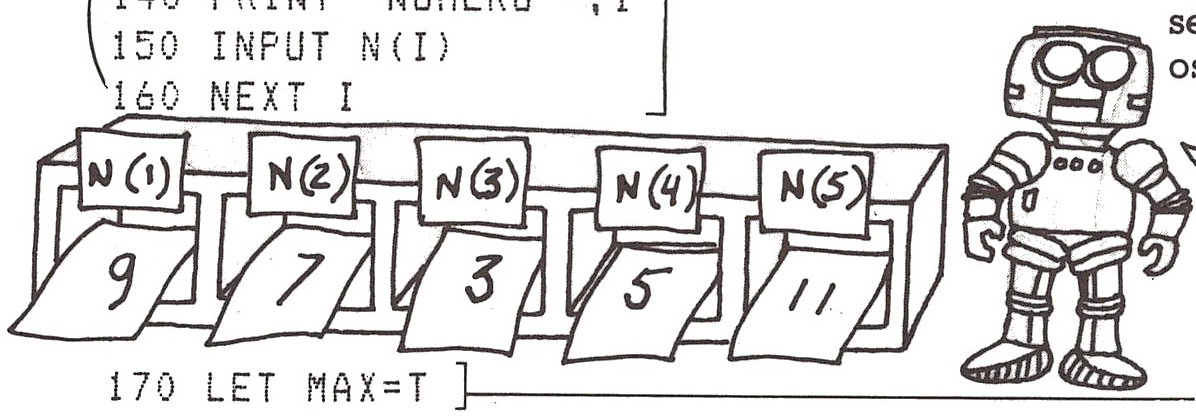
Nos ordenadores de bolha, o computador começa por comparar os dois primeiros elementos da lista. Se estão na ordem errada, troca-os de posição. Em seguida, compara o segundo elemento com o terceiro, e assim por diante. Durante a execução do programa, os números menores sobem como "bolhas" para o topo da lista.

O programa abaixo é um ordenador de bolha para números. Na página 32 mostramos outra versão do programa, desta vez para ordenar palavras.

```
100 REM ORDENADOR DE BOLHA PARA NUMEROS
110 INPUT "QUANTOS NUMEROS PARA SEREM
    ORDENADOS ";T
120 DIM N(T)
130 FOR I=1 TO T
140 PRINT "NUMERO ";I
150 INPUT N(I)
160 NEXT I
```

Define uma matriz chamada N com T compartimentos. T é o número total de números que você quer ordenar.

Pede a você os números a serem ordenados e guarda-os na matriz.




```
170 LET MAX=T
```

Define outra variável chamada MAX para guardar o número total porque o valor de T vai ser alterado pelo programa.

```
175 LET X=0
```

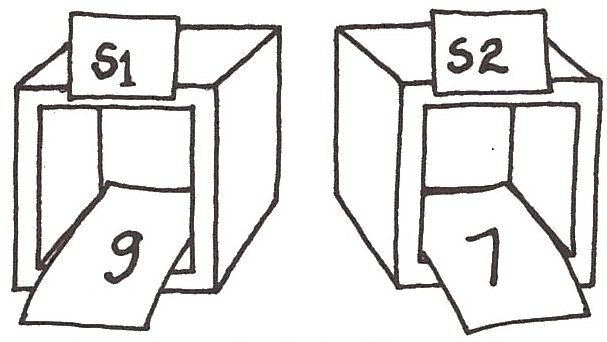
X é uma variável de contagem.

```
180 FOR C=1 TO T-1
```

T-1 é o número de repetições necessário para comparar todos os pares da lista.

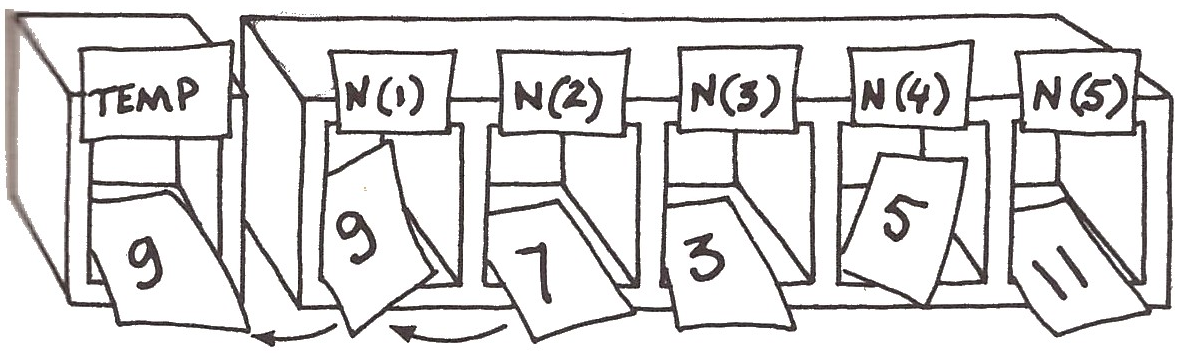
```
190 LET S1=N(C):LET S2=N(C+1)
```

As variáveis S1 e S2 são para guardar os números do par enquanto estão sendo comparados. Na primeira passagem pelo loop C = 1, de modo que N(1) e N(2) são guardados em S1 e S2.




```
200 IF S1<=S2 THEN GOTO 250
```

Compara os dois números. Se S1 é menor que S2, os números estão na ordem correta e a linha 250 manda o computador de volta ao início do loop para comparar o próximo par de números. Se S1 é maior que S2, o computador passa para as linhas seguintes, onde as posições dos dois números são trocadas.




```
210 LET TEMP=N(C)
```

O número que está em N(C) é colocado em uma variável chamada TEMP.

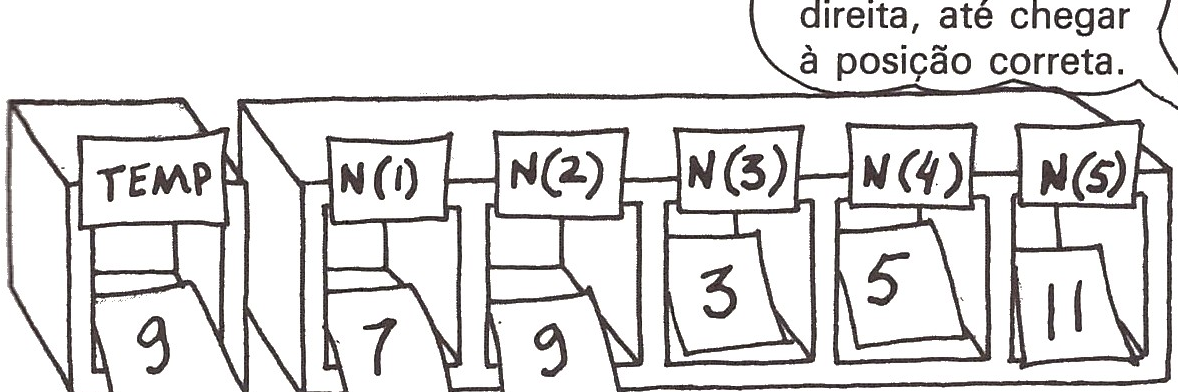
```
220 LET N(C)=N(C+1)
```

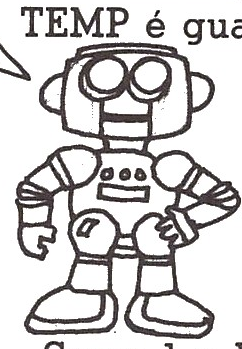
O número que está em N(C + 1) é transferido para N(C).

```
230 LET N(C+1)=TEMP
```

O número que está em TEMP é guardado em N(C + 1).

Cada vez que o loop é repetido, o número é deslocado uma posição para a direita, até chegar à posição correta.





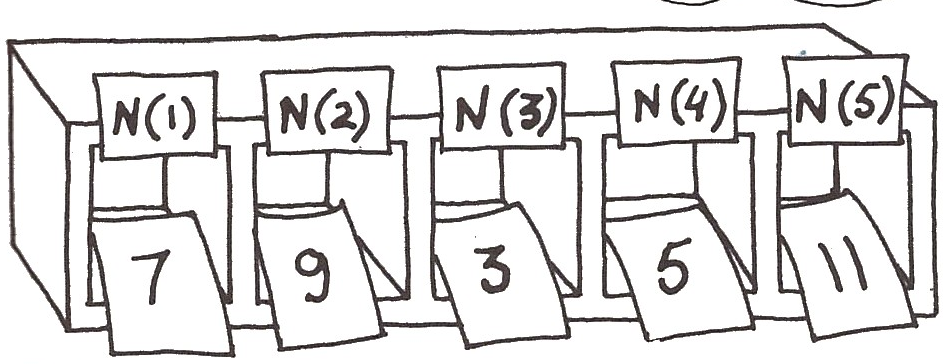
```
240 LET X=X+1
```

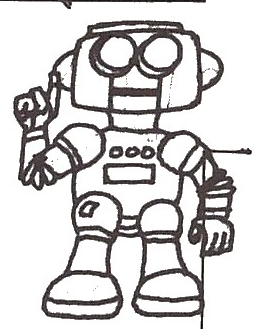
Soma 1 a X para mostrar que houve uma troca.

Da segunda vez, C = 2, de modo que N(C) = N(2) e N(C + 1) = N(3).

```
250 NEXT C
```

Manda o computador de volta para comparar o próximo par de números. Depois de repetir o loop T-1 vezes, o computador já comparou todos os pares e passa para linha 260.





```
260 IF X>0 THEN LET T=T-1:GOTO 175
```

Se X é maior que 0, é sinal de que houve uma troca; o computador subtrai 1 de T, pois sabe que pelo menos o último número está na posição correta, e volta para o início do loop. Se X = 0, os números já estão ordenados e o computador passa para a linha 270.

```
270 PRINT "OS NUMEROS ORDENADOS SAO"
280 FOR I=1 TO MAX
290 PRINT N(I)
300 NEXT I
```

MAX é o número total de números que foram ordenados.

Se colocar o GOTO da linha 260 em uma linha separada, repita o IF...THEN.

Ordenador de bolha para palavras

O programa abaixo é um ordenador de bolha para palavras. Funciona da mesma forma que o programa anterior, mas as variáveis usadas para guardar os dados (N,S1,S2 e TEMP) agora têm que ser do tipo string.*

Para o computador, tanto faz comparar números ou letras. Dentro do micro, as letras e símbolos são representados por números, de modo que quando você manda o computador comparar duas letras, o que ele compara são os códigos numéricos correspondentes. Para comparar palavras, ele compara as primeiras letras das duas palavras; se forem iguais, compara as segundas letras, e assim por diante. Como uma variável string também serve para guardar números, o programa abaixo pode ser usado para ordenar dados que contenham palavras e números, como o índice de um livro ou uma lista de endereços.

```
QUANTAS PALAVRAS PARA SEREM ORDENADAS ?4
PALAVRA 1
?UNIDADE DE DISCO 34 76 82 93
PALAVRA 2
?LINGUAGEM DE MAQUINA 55 72 85
PALAVRA 3
?SONS 32
PALAVRA 4
?GRAFICOS 8 23 45
AS PALAVRAS ORDENADAS SAO
GRAFICOS 8 23 45
LINGUAGEM DE MAQUINA 55 72 85
SONS 32
UNIDADE DE DISCO 34 76 82 93
```

```
QUANTAS PALAVRAS PARA SEREM ORDENADAS ?4
PALAVRA 1
?CELINHA PRAIA DE BOTAFOGO 99
PALAVRA 2
?BEBEL RUA DUVIVIER 88/203
PALAVRA 3
?INGRID LARGO DO MACHADO 111/501
PALAVRA 4
?MARCIA RUA BELA 19
AS PALAVRAS ORDENADAS SAO
BEBEL RUA DUVIVIER 88/203
CELINHA PRAIA DE BOTAFOGO 99
INGRID LARGO DO MACHADO 111/501
MARCIA RUA BELA 19
```

Nos exemplos acima, o computador está ordenando os elementos de um índice e de uma lista de endereços. Repare na ausência de vírgulas; na maioria dos micros, a vírgula é usada para separar dados consecutivos. Se você quiser usar vírgulas dentro de strings, coloque toda a string entre aspas.

O programa

```
100 REM ORDENADOR DE BOLHA PARA PALAVRAS
110 INPUT "QUANTAS PALAVRAS PARA SEREM
    ORDENADAS ";T
120 DIM N$(T)
130 FOR I=1 TO T
140 PRINT "PALAVRA ";I
150 INPUT N$(I)
160 NEXT I
170 LET MAX=T
175 LET X=0
180 FOR C=1 TO T-1
190 LET S1$=N$(C):LET S2$=N$(C+1)
200 IF S1$<=S2$ THEN GOTO 250
210 LET TEMP$=N$(C)
220 LET N$(C)=N$(C+1)
230 LET N$(C+1)=TEMP$
240 LET X=X+1
250 NEXT C
260 IF X>0 THEN LET T=T-1:GOTO 175
270 PRINT "AS PALAVRAS ORDENADAS SAO"
280 FOR I=1 TO MAX
290 PRINT N$(I)
300 NEXT I
```

Define uma matriz chamada N$, com T compartimentos.

Experimente entrar com elementos começando com símbolos, números, letras maiúsculas e minúsculas e veja em que ordem são colocados pelo seu micro.

Os primeiros dois elementos são colocados em S1$ e S2$.

Compara S1$ e S2$.

Troca as posições de dois elementos consecutivos de N$.

Se X > 0, subtrai 1 do número total de elementos e volta ao início do loop.

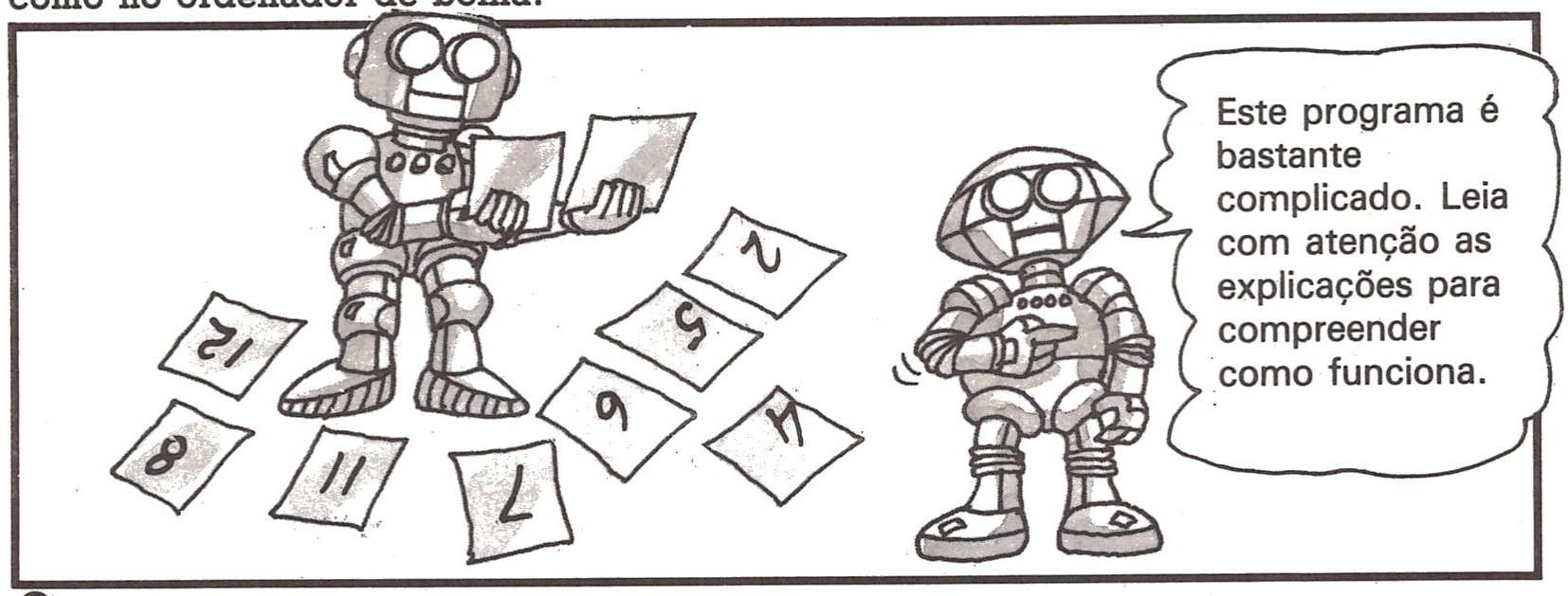
Se colocar o GOTO da linha 260 em uma linha separada, repita o IF...THEN.

*Nos micros da família Sinclair (TK-83, TK-85, CP-200) mude a linha 120 para DIM N$(T,N), onde N é o número de caracteres da maior string que você pretende usar.

Ordenador de Shell

Quando o número de elementos é razoavelmente grande, o ordenador de bolha se torna excessivamente lento. Em alguns micros, ele pode levar quase um minuto para ordenar cinqüenta elementos. O programa abaixo é um ordenador de Shell para números, cerca de três vezes mais rápido que o ordenador de bolha.

No ordenador de Shell, o computador divide os elementos a serem ordenados em dois grupos e compara todos os elementos de um grupo com todos os elementos do outro. Em seguida, divide cada grupo ao meio e torna a realizar novas comparações, trocando os elementos de posição sempre que necessário, como no ordenador de bolha.



O programa

```
100 REM ORDENADOR DE SHELL
    PARA NUMEROS
110 INPUT "QUANTOS NUMEROS
    PARA SEREM ORDENADOS ";T
120 DIM N(T)
130 FOR I=1 TO T
140 PRINT "NUMERO ";I
150 INPUT N(I)
160 NEXT I
170 LET C=T
180 LET C=INT(C/2)
190 IF C=0 THEN GOTO 330
200 LET D=T-C
210 LET E=1
220 LET F=E
230 LET G=F+C
```

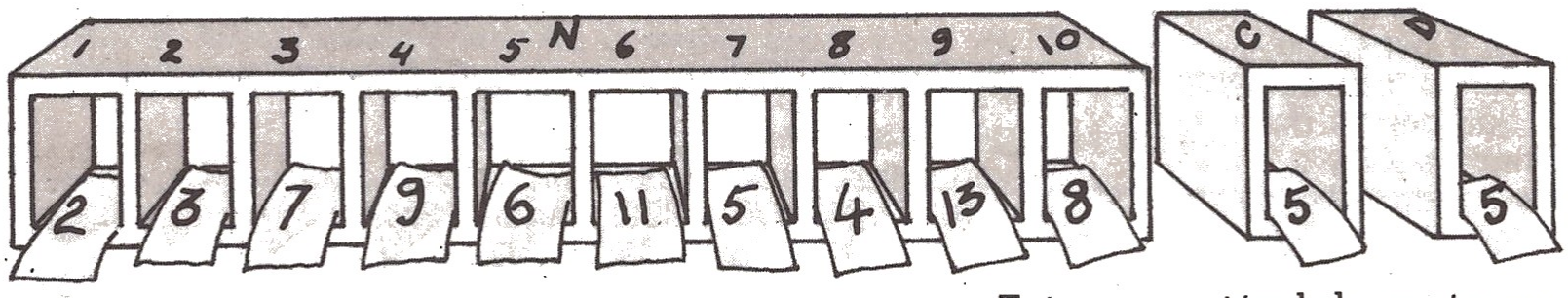
Estas linhas são iguais às do ordenador de bolha. T é o número total de números a serem ordenados e as linhas 120 e 160 pedem a você os números e guardam-nos na matriz N.

O número de elementos é guardado na variável C.

Divide C ao meio para calcular o número de elementos do primeiro grupo. INT faz o computador desprezar a parte fracionária de C, transformando-o em um número inteiro.

O programa divide C ao meio várias vezes; quando C = 0, o computador vai para a linha 330 para imprimir a lista ordenada.

D é o número de elementos do segundo grupo.



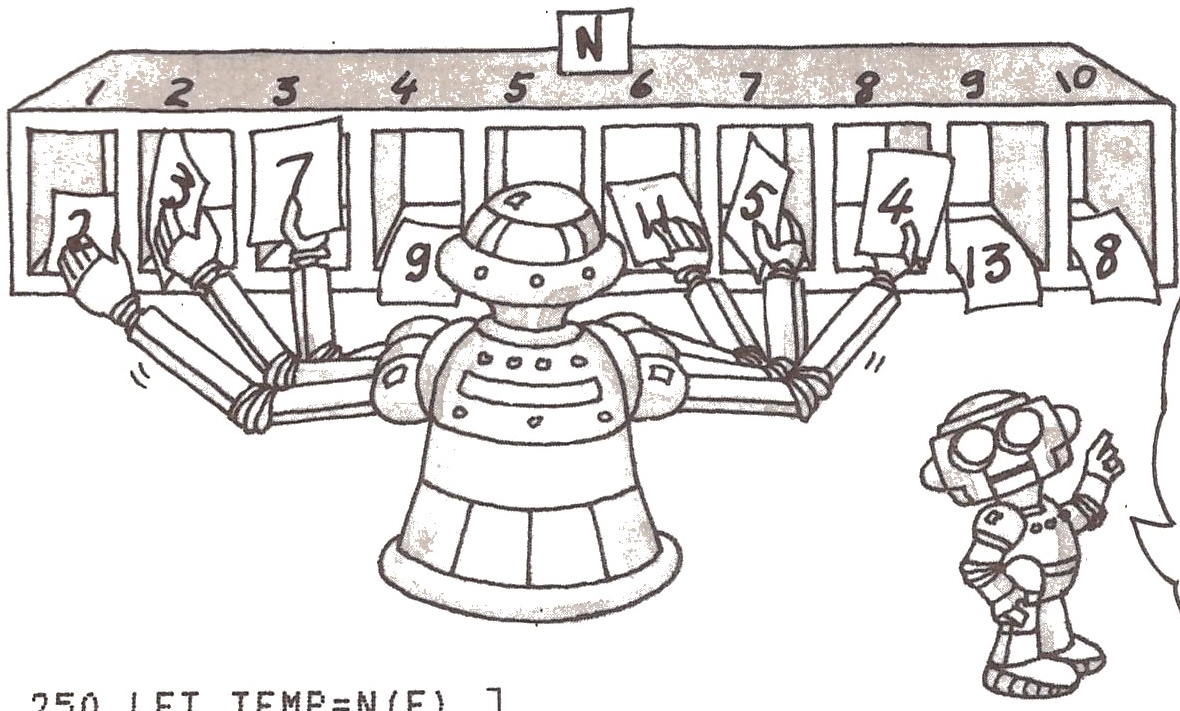
E é uma variável de contagem.

As variáveis F e G são usadas para guardar os índices de N, a matriz onde estão guardados todos os números a serem ordenados.

A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

```
240 IF N(F)<=N(G) THEN GOTO 300
```

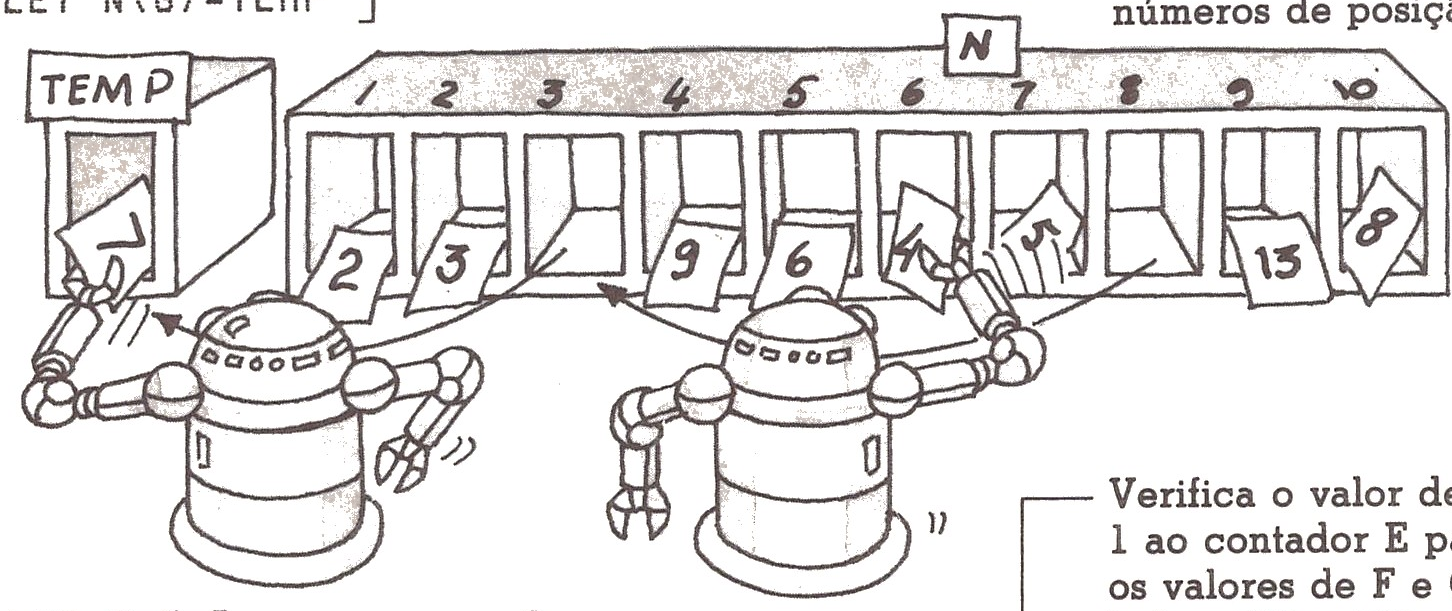
Se o número que está em N(F) é menor ou igual ao que está em N(G), o computador vai para 300, soma 1 a E e volta para modificar os valores de F e G.



Neste exemplo há dez números para serem ordenados. Da primeira vez, C=5, de modo que o computador compara os números que estão em N(1) a N(5) com os que estão em N(6) a N(10).

```
250 LET TEMP=N(F)
260 LET N(F)=N(G)
270 LET N(G)=TEMP
```

Se N(F) é maior que N(G), o computador troca os dois números de posição.




```
280 LET F=F-C
290 IF F>0 THEN GOTO 230
300 LET E=E+1
310 IF E>D THEN GOTO 180
```

Verifica o valor de F e soma 1 ao contador E para mudar os valores de F e G nas linhas 220 e 230.

Verifica se E ainda é menor que D. Se não, volta à linha 180 para subdividir os grupos.

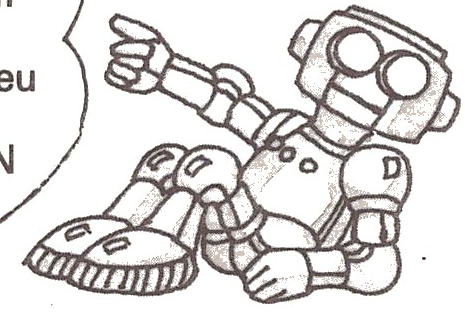
```
320 GOTO 220
330 PRINT "OS NUMEROS ORDENADOS SAO"
340 FOR I=1 TO T
350 PRINT N(I)
360 NEXT I
```

Se E é menor que D, o computador vai para a linha 220 para calcular os índices de mais um par de números.

Imprime a lista ordenada.

Veja se é capaz de acrescentar instruções para que depois de ordenada a lista o computador escreva quantas comparações e quantas trocas teve que executar. A resposta está na página 48.

Para transformar este programa em um ordenador de palavras, mude a variável N para N$ (linhas 120,150,240-270 e 350), mude TEMP para TEMP$ (linhas 250 e 270) e mude as instruções PRINT. Se o seu micro é da família Sinclair, mude a linha 120 para DIM N$(T,N) onde N é o trabalho da maior string que você pretende usar.



Como funciona?

Para ter uma idéia melhor de como funciona o ordenador de Shell, acrescente as linhas abaixo. Elas imprimem os valores das variáveis, para que você possa ver que números estão sendo comparados.

```
233 PRINT
235 PRINT "F=";F;",G=";G
237 PRINT "COMPARE N(";F;") E N(";G;")"
245 PRINT "TROQUE N(";F;") E N(";G;")"
274 PRINT "LISTA=";
275 FOR J=1 TO T
276 PRINT N(J);" ";
277 NEXT J
278 PRINT
279 INPUT "APERTE RETURN PARA CONTINUAR";Z$
```

F e G são os índices dos números a serem comparados.

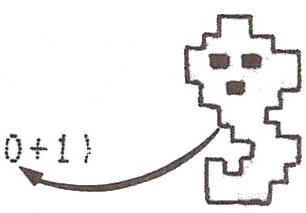
Imprime os índices dos números a serem trocados.

Imprime a lista na ordem em que está no momento.

Comparação entre os ordenadores

Se você testou os ordenadores de bolha e de Shell com uns poucos números, talvez não tenha percebido como o ordenador de Shell é mais rápido. Para comparar os dois ordenadores, você pode gerar uma longa série de números aleatórios e verificar o tempo que cada programa leva para colocá-los em ordem. Quanto maior a lista de números, maior a diferença de tempo entre os dois programas. Na página seguinte você vai aprender a fazer um gráfico que mostra a diferença entre os dois ordenadores.

Como gerar os números



Você pode mudar este número à vontade.

```
140 LET N(I)=INT(RND(1)*200+1)
150 PRINT N(I)
165 INPUT "PREPARE O RELOGIO E
    APERTE RETURN PARA COMECAR";Z$
```

Para fazer os programas gerarem números aleatórios, você precisa substituir as linhas 140 e 150 nos dois programas e inserir uma linha 165 que lhe permita iniciar a ordenação no instante desejado.

A linha 140 gera números aleatórios entre 1 e 200 e os armazena na matriz N. A linha 150 mostra os números na tela e a linha 165 faz o programa esperar até que você aperte **RETURN**.

Como executar o teste

Para testar cada programa, você deve executá-lo várias vezes, com um número diferente de elementos, como 10,20,30, etc. A velocidade varia de acordo com o micro. A tabela abaixo mostra as velocidades para um Ap-II.

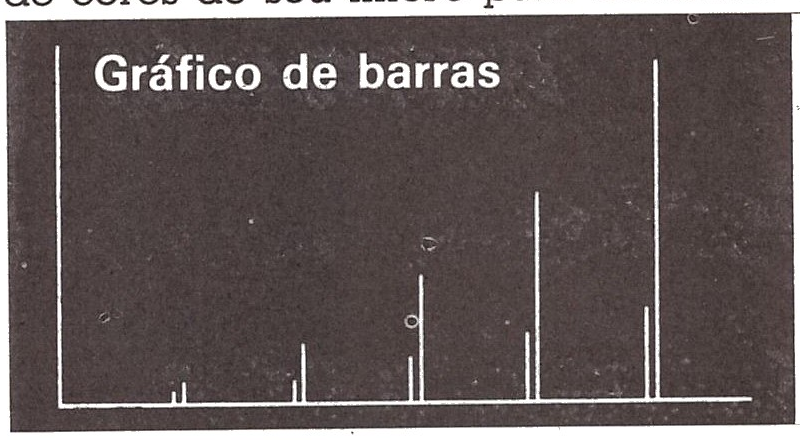
Teste dos programas ordenadores

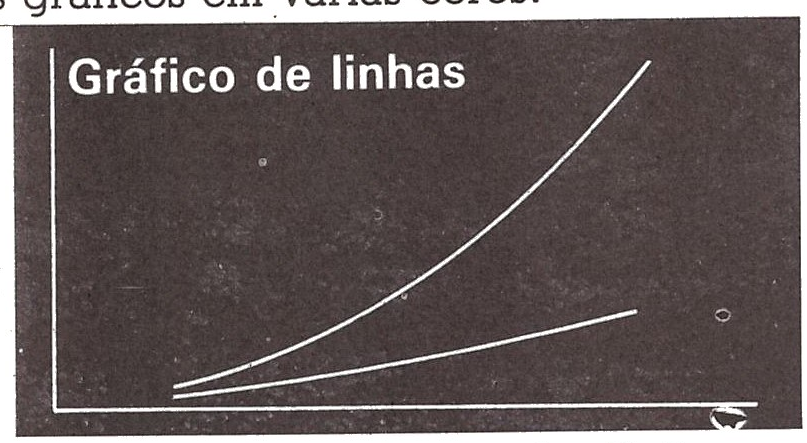
Número de elementos	10	20	30	40	50
Ordenador de bolha	2 segundos	5 segundos	11 segundos	18 segundos	29 segundos
Ordenador de Shell	1 segundo	2 segundos	4 segundos	6 segundos	8 segundos

Programas para fazer gráficos

Os resultados fornecidos por um computador podem ser muito mais fáceis de interpretar se forem apresentados sob a forma de gráficos. O programa abaixo constrói um gráfico de barras para mostrar a diferença entre o ordenador de bolha e o ordenador de Shell. Na página seguinte você vai aprender a modificar o programa para fazer um gráfico de linhas.

Os programas são bastante simples e você pode adaptá-los para o tipo de informação que desejar. Pode também aperfeiçoá-los, usando as instruções de cores do seu micro para desenhar os gráficos em várias cores.





Estas são as saídas dos dois programas. Os dois gráficos comparam o tempo que os dois programas ordenadores levam para ordenar 10, 20, 30, 40 e 50 números. O tempo está representado ao longo do eixo dos Y e o número de números ao longo do eixo dos X. Se o seu micro é capaz de escrever palavras em posições especificadas da tela, você pode colocar legendas nos gráficos.

Programa do gráfico de barras

```
100 DIM B(5): DIM S(5)
110 LET N=0
120 FOR I=10 TO 50 STEP 10
130 LET N=N+1
140 PRINT "PARA ";I;" NUMEROS"
150 INPUT "QUANTOS SEGUNDOS LEVOU
    O ORDENADOR DE BOLHA ";B(N)
160 INPUT "E O ORDENADOR DE SHELL ";S(N)
170 NEXT I
180 INPUT "QUANTAS LINHAS TEM SUA TELA ";L
190 INPUT "E QUANTAS COLUNAS ";C
200 REM USE AS INSTRUCOES GRAFICAS
    DO SEU MICRO .
210 REM DESENHAR EIXOS
220 PLOT 1,L:DRAW 1,1: DRAW C,1
230 REM DESENHAR GRÁFICO
240 LET X=(C*0.75)/5
250 LET Y=(L*0.75)/B(5)
```

Define as matrizes B e S para guardar os dados relativos aos dois ordenadores.

Loop para guardar os dados nas matrizes. N é uma variável de contagem.

Observe que a variável I também é usada para contar os números.

Os números em PLOT e DRAW são medidos a partir das margens da tela. Se no seu micro a medida é feita a partir da última posição do cursor, veja o que fazer na página 47.

Traça os eixos a 1 pixel de distância das margens da tela. L e C são a altura e a largura da sua tela.

Os números em X e Y estabelecem as escalas do gráfico ao longo dos eixos X e Y. X é igual a três quartos da largura dividido por 5, o número de testes de cada ordenador. Y é três quartos da altura dividido pelo maior tempo, isto é, B(5).

A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

```
260 FOR I=1 TO 5
270 PLOT INT(I*X),1
280 DRAW INT(I*X),INT(B(I)*Y)
290 PLOT INT(I*X-4),1
300 DRAW INT(I*X-4),INT(S(I)*Y)
310 NEXT I
```

Você é capaz de mudar o programa para traçar barras mais largas? (Resposta na pág. 48.)

Você pode mudar o -4 para aproximar ou afastar as barras.

Inicia um loop para traçar as barras. Cada vez que o loop é repetido, o computador traça duas barras, uma para cada tipo de ordenador.

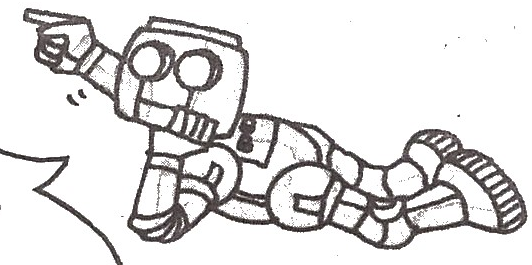
Marca um ponto sobre o eixo dos X, que será a base de uma das barras.

Traça uma reta até B(I) x Y. O comprimento da reta é proporcional ao tempo que o ordenador de bolha levou para ordenar um certo número de números.

Marca um ponto 4 pixels à esquerda da barra que o computador acabou de traçar.

Traça uma reta até S(I) x Y.

Volta ao início do loop para traçar mais um par de barras.



Dependendo do modelo do seu micro, talvez você possa colocar legendas nas barras do gráfico. Para fazer isso no DGT-1000, por exemplo, basta acrescentar as linhas abaixo.

```
193 FOR I=1 TO 5:PRINT @64*INT(13-.4*B(I))
+INT(9.5*I)+64,"B";:NEXT I
194 FOR I=1 TO 5:PRINT @64*INT(13-.4*S(I))
+INT(9.5*I)+126,"S";:NEXT I
```

Programa do gráfico de linhas

Para fazer um gráfico de linhas, você precisa marcar o primeiro ponto do gráfico, traçar uma linha até o ponto seguinte e assim por diante. Vai precisar de loops separados para cada ordenador. Para adaptar o programa do gráfico de barras para fazer um gráfico de linhas, substitua as linhas de 270 em diante pelas linhas abaixo.

```
270 PLOT INT(X),INT(B(1)*Y)
280 FOR N=2 TO 5
290 DRAW INT(N*X),INT(B(N)*Y)
300 NEXT N
310 PLOT INT(X),INT(S(1)*Y)
320 FOR N=2 TO 5
330 DRAW INT(N*X),INT(S(N)*Y)
340 NEXT N
```

Marca o primeiro ponto do ordenador de bolha.

Loop para traçar o gráfico do ordenador de bolha. Cada vez que o loop é repetido, N aumenta de 1 e o computador traça uma reta até o ponto seguinte.

Marca o primeiro ponto do ordenador de Shell.

Loop para traçar o gráfico do ordenador de Shell.

Programa avançado de manipulação de strings

O programa que aparece nas próximas páginas faz com que o computador converse com você. É evidente que todas as palavras e frases para as respostas do computador estão contidas no programa, armazenadas em strings.*

A principal dificuldade do programa é fazer com que o computador escolha palavras e frases que lhe permitam manter um diálogo coerente. O programa contém algumas rotinas de manipulação de strings que fazem com que suas respostas às vezes pareçam quase "inteligentes". O sucesso de um programa desse tipo não está apenas em sua estrutura, mas também nas palavras e frases escolhidas. Você pode mudar à vontade o vocabulário do micro para fazê-lo "conversar" a respeito de diferentes assuntos ou tornar suas respostas mais cordiais... ou desagradáveis.

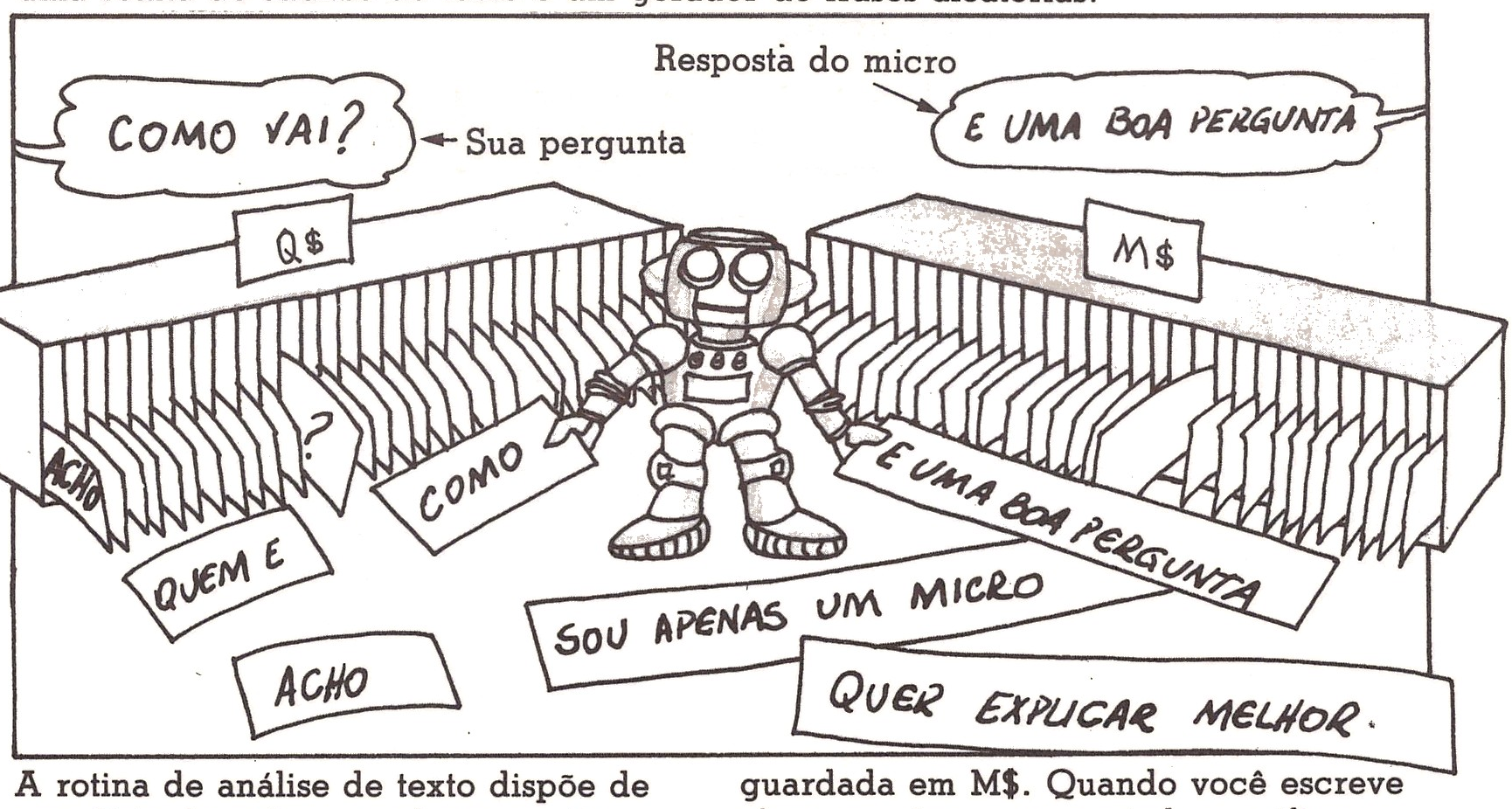
Exemplos

```
BOM DIA. COMO SE CHAMA ?MARISA
DIGA ALGUMA COISA, MARISA
?BOM DIA, COMPUTADOR
QUE ACHA DA VIOLENCIA URBANA?
?ACHO QUE E UM .PROBLEMA GRAVE
EU NAO ACHO
NAO ACHA?
MARISA, ACHO QUE VOCE E TAO
INTERESSANTE QUANTO AS OUTRAS
PESSOAS QUE CONVERSARAM COMIGO
?OBRIGADA
```

```
DE NADA
?SOBRE QUE VAMOS CONVERSAR?
VAMOS DISCUTIR ALGUMA COISA MAIS ESTIMULANTE
?ESTA BEM
NAO ACHA QUE O SER HUMANO E REVOLTANTE?
?CLARO QUE NAO
OUVI FALAR QUE VOCE TEM UMA PERSONALIDADE
EXTREMAMENTE NEUROTICA, MARISA
?QUEM LHE CONTOU?
VOCE SABE MUITO BEM.
?NAO SEI, NAO
ACHA QUE EU SOU FASCINANTE, MARISA?
```

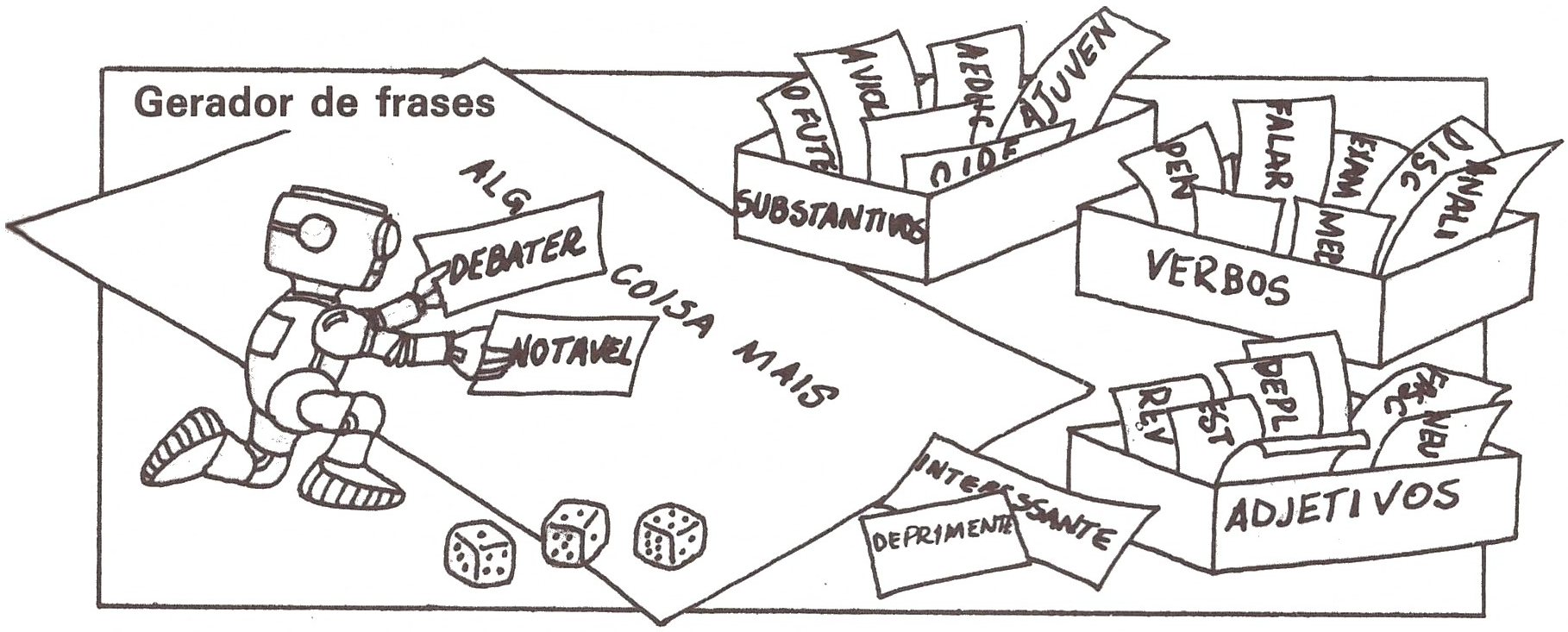
Como funciona

O programa usa dois métodos diferentes para gerar as respostas do computador, uma rotina de análise de texto e um gerador de frases aleatórias.



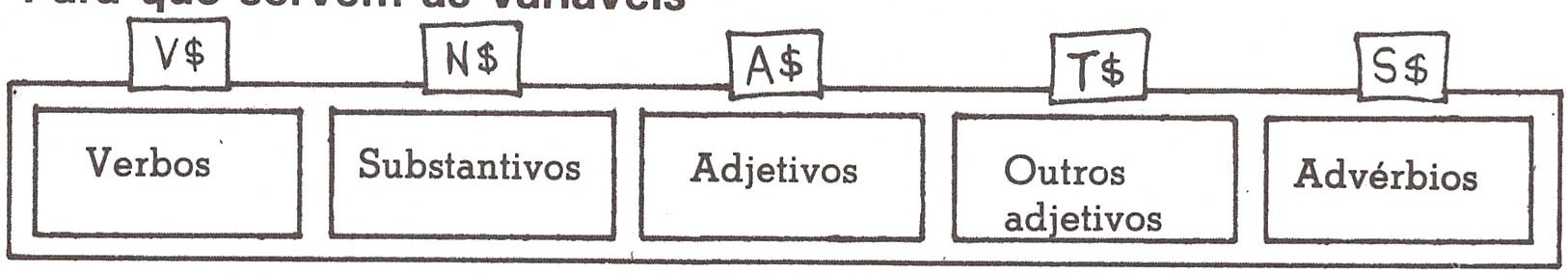
A rotina de análise de texto dispõe de uma lista de palavras e frases muito usadas, guardada em uma matriz chamada Q$. Para cada palavra ou frase existe uma resposta conveniente guardada em M$. Quando você escreve alguma coisa, o computador verifica se você usou alguma das frases de Q$; em caso afirmativo, procura a resposta correspondente em M$.

*As alterações necessárias para que este programa funcione nos micros da família Sinclair estão na página 47.

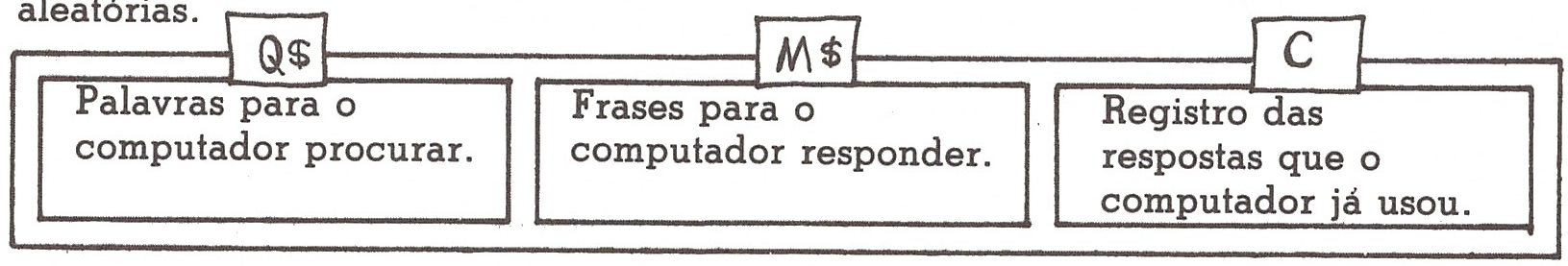


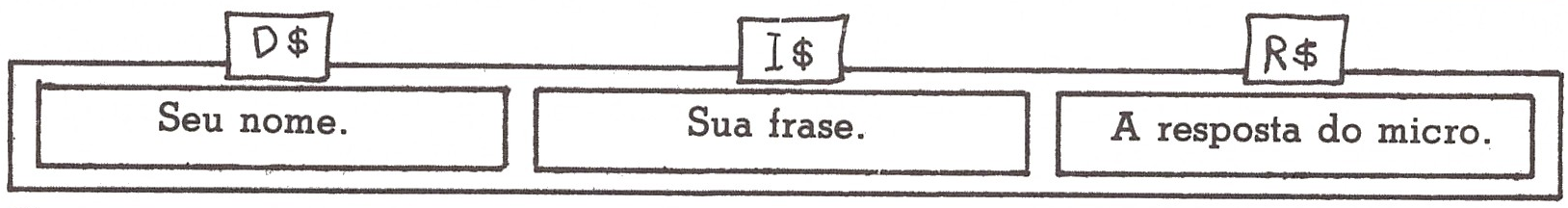
O gerador de frases aleatórias utiliza frases semi-acabadas, que o computador completa com verbos, substantivos, adjetivos e advérbios escolhidos ao acaso. Todas as palavras estão guardadas em matrizes na memória do computador.

Para que servem as variáveis



As matrizes **V$, N$, A$, T$,** e **S$** contêm as palavras usadas para gerar as frases aleatórias.





O programa

```
100 CLS
110 DIM V$(10),N$(10),A$(10)
120 DIM R$(10),S$(10),T$(10)
130 DIM M$(30),Q$(30),C(30)
135 GOSUB 1800
140 REM LEITURA DE DADOS
150 GOSUB 1000
200 REM INICIO DA CONVERSA
210 INPUT "BOM DIA. COMO SE CHAMA ";D$
220 PRINT
230 PRINT "DIGA ALGUMA COISA, ";D$
240 INPUT I$
250 IF I$="" THEN GOTO 220
260 IF I$="TCHAU" THEN GOTO 910
```

Define as matrizes usadas para guardar as palavras e frases. (Em alguns micros, não é necessário dimensionar matrizes com menos de 10 elementos.)

Vai para uma sub-rotina para guardar todas as palavras e frases nas matrizes.

A sua frase é guardada em **I$**.

Precaução para o caso de você apertar **RETURN** sem escrever nada.

Se você escrever **TCHAU**, o micro vai para 910 e pergunta se você quer parar.

A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

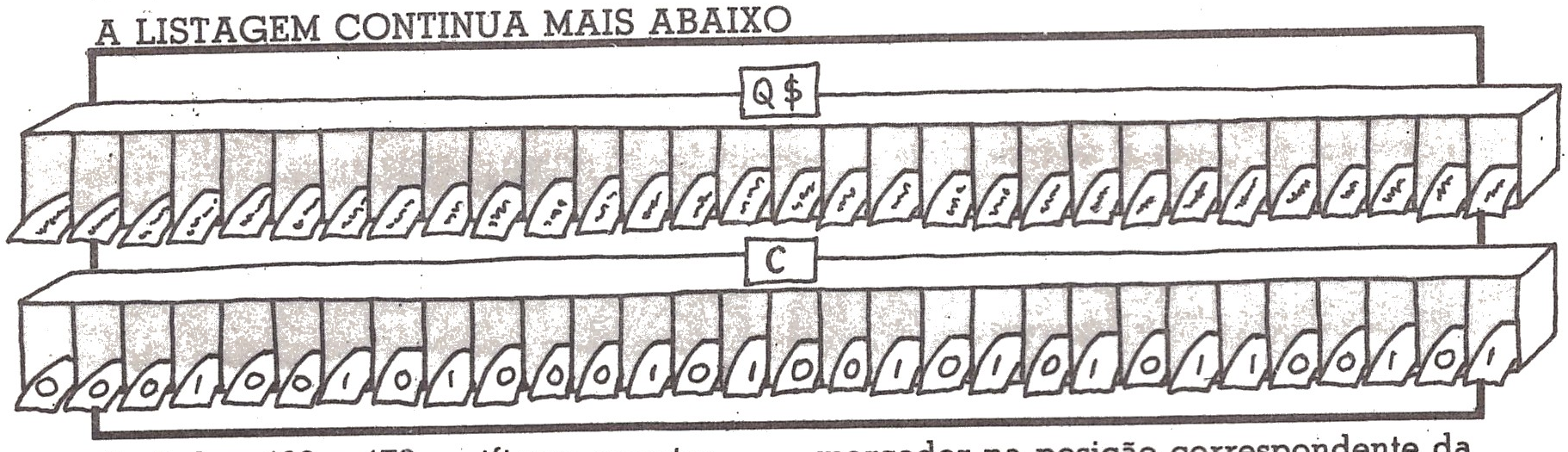
```
300 REM RESPOSTA DO COMPUTADOR
310 LET R=INT(RND(1)*8+1)
320 IF R<6 THEN GOTO 490

330 GOTO 600
340 PRINT
350 FOR I=1 TO 10:PRINT R$(I);: NEXT I
355 GOSUB 1800
360 PRINT
400 REM VERIFICA QUANTAS RESPOSTAS FORAM USADAS
```

O número aleatório da variável R é usado para escolher o modo de resposta do computador. Se R é menor que 6, ele usa a rotina de análise de texto que começa na linha 490.

Se R é maior ou igual a 6, vai para o gerador de frases que começa na linha 600.

O computador mostra a resposta que está guardada em R$ e depois vai para 1800 para esvaziar a matriz R$.

A LISTAGEM CONTINUA MAIS ABAIXO



As linhas 400 a 470 verificam quantas das respostas em M$ foram usadas. Cada vez que o computador encontra uma das palavras de Q$ na sua frase, ele coloca o algarismo 1 como marcador na posição correspondente da matriz C. Isto impede que ele use de novo a mesma resposta. Depois de usadas mais de 12 respostas, os marcadores são todos removidos.

```
405 LET T=0
410 FOR K=1 TO 30
420 LET T=T+C(K)
430 NEXT K
440 IF T<12 THEN GOTO 460
450 FOR K=1 TO 30: LET C(K)=0: NEXT K
460 LET T=0
470 GOTO 240
490 REM ANALISE DO TEXTO
500 FOR P=1 TO 30

510 LET L1=LEN(Q$(P))

520 LET L2=LEN(I$)

530 FOR TT=1 TO L2

540 IF MID$(I$,TT,L1)=Q$(P)
    THEN GOTO 560

550 NEXT TT: NEXT P: GOTO 600
```

Loop para contar quantos marcadores existem na matriz C. O total é guardado em T.

Se T é menor que 12, foram usadas menos de 12 respostas e o computador não remove os marcadores.

Loop para remover os marcadores.

A variável T (total de marcadores) volta para zero.

Volta à linha 240 para esperar a frase da pessoa.

Loop para ser repetido tantas vezes quantas palavras estão guardadas em Q$.

Cada vez que o loop é repetido, o computador mede o tamanho de uma palavra em Q$ e guarda o resultado em L1.

O número de caracteres da sua frase é guardado em L2.

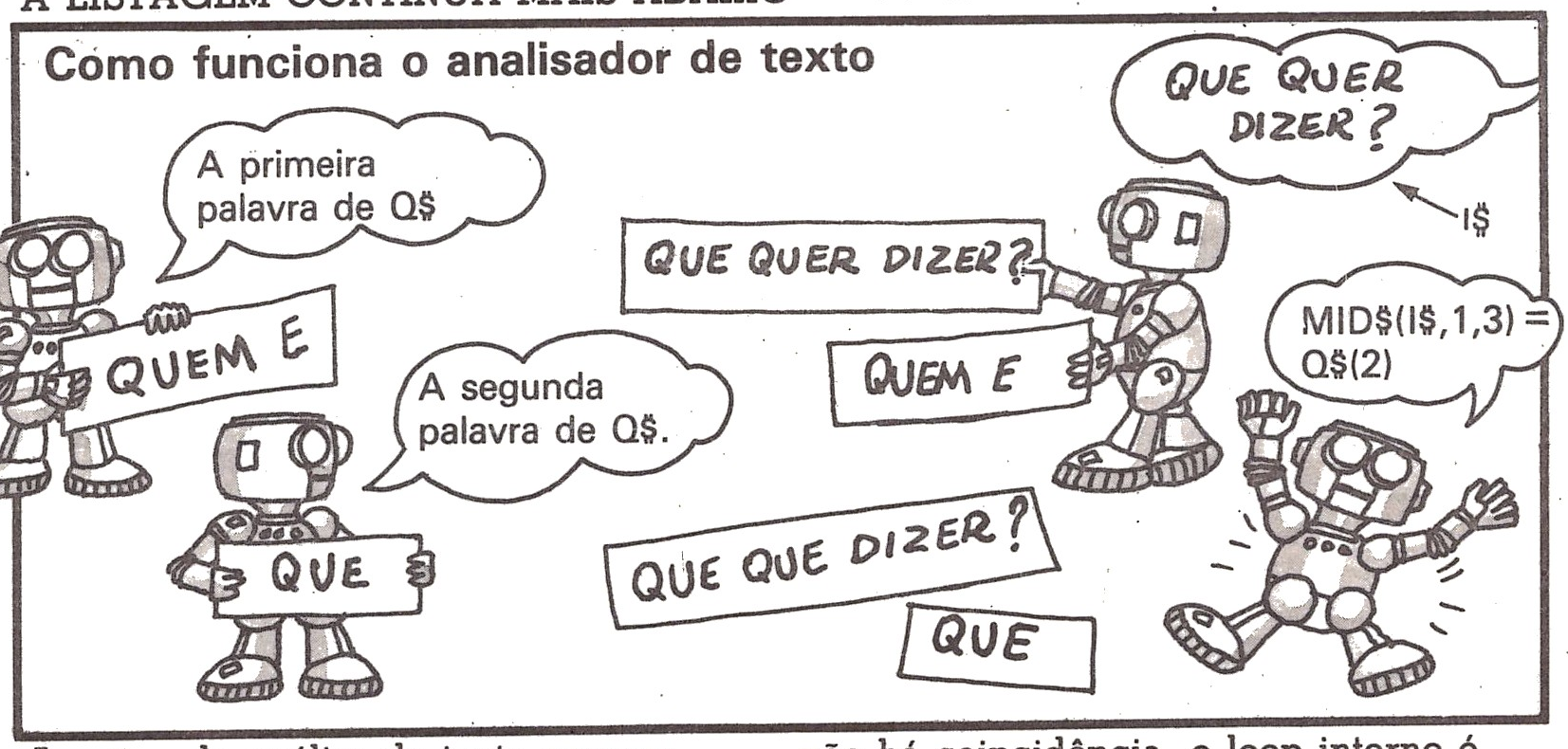
TT é o índice de um loop interno que é executado tantas vezes quantos caracteres existem na sua frase. Cada vez que o loop é repetido, o computador compara os caracteres de I$ com os das palavras de Q$. Em caso de coincidência, vai para a linha 560.

Se depois de esgotados todos os loops o computador não consegue encontrar em I$ nenhuma das palavras de Q$, ele passa para o gerador de frases aleatórias, que começa na linha 600.

```
560 IF C(P)>0 THEN GOTO 550
570 LET C(P)=C(P)+1
580 LET R$(1)=M$(P)
590 GOTO 340
```

Em caso de coincidência, o computador sai do loop e pula para a linha 560. Verifica então o marcador correspondente à palavra. Se o marcador não é 0, volta para os loops para ver se encontra outra palavra coincidente em I$. Se o marcador é 0, muda-o para 1, procura a frase correspondente em M$ e guarda-a em R$(1), pronta para ser impressa na linha 350.

A LISTAGEM CONTINUA MAIS ABAIXO



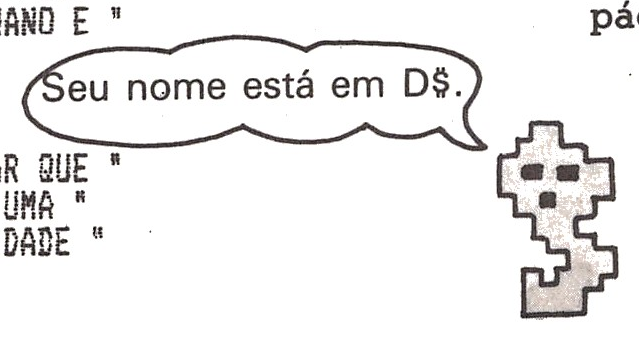
A rotina de análise de texto começa com P = 1, de modo que o computador examina a primeira palavra de Q$. O loop interno começa com TT = 1, de modo que o computador compara a palavra em Q$ com os caracteres de 1 a 6 de I$ (o comprimento de Q$). Se não há coincidência, o loop interno é repetido com TT = 2, isto é, são comparados os caracteres 2 a 7 de I$, e assim por diante. Depois de esgotado o loop interno, o computador volta ao início do loop externo para examinar a palavra seguinte de Q$.

```
600 REM GERADOR DE FRASES
610 LET E=INT(RND(1)*10+1)
620 LET F=INT(RND(1)*10+1)
630 LET G=INT(RND(1)*10+1)
640 LET H=INT(RND(1)*10+1)
650 LET L=INT(RND(1)*10+1)
```

Números aleatórios para escolher as palavras e frases a serem usadas. O número E é usado para escolher a frase, F o verbo, G o adjetivo, H o substantivo e L o advérbio.

```
660 ON E GOTO 700,720,740,760,
    780,800,830,850,870,890
700 LET R$(1)="QUE ACHA D"
701 LET R$(2)=N$(H)
702 LET R$(3)="?"
704 GOTO 340
720 LET R$(1)=D$
721 LET R$(2)=", NAO ACHA QUE "
722 LET R$(3)="O SER HUMANO E "
723 LET R$(4)=A$(G)
724 LET R$(5)=" ?"
725 GOTO 340
740 LET R$(1)="OUVI FALAR QUE "
741 LET R$(2)="VOCE TEM UMA "
742 LET R$(3)="PERSONALIDADE "
743 LET R$(4)=S$(L)
744 LET R$(5)=" "
745 LET R$(6)=T$(G)
746 GOTO 340
760 LET R$(1)=D$
```

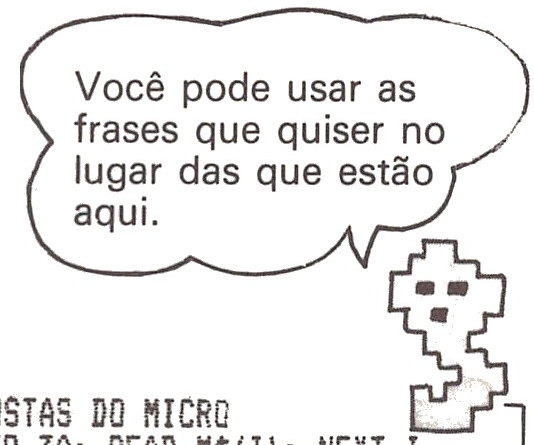
O número que está em E diz ao computador para onde deve ir. Se E = 1, vai para a primeira linha da lista; se E = 2, vai para a segunda etc. A palavra ON já foi vista na página 20.



As linhas 700 a 900 contêm dez frases semi-acabadas, que o computador completa com as palavras de N$, V$, etc. O computador guarda a frase completa na matriz R$ e depois volta à linha 340 para imprimi-la.

A LISTAGEM CONTINUA NA PÁGINA SEGUINTE

```
761 LET R$(2)=" ,ACHO QUE "
762 LET R$(3)="VOCE E TAO "
763 LET R$(4)=A$(G)
764 LET R$(5)=" QUANTO AS OUTRAS"
765 LET R$(6)=" PESSOAS QUE CONVE"
766 LET R$(7)="RSARAM COMIGO"
767 GOTO 340
780 LET R$(1)="ESTOU ME SENTINDO "
781 LET R$(2)=S$(L)
782 LET R$(3)=" "
783 LET R$(4)=A$(G)
784 LET R$(5)=" NO MOMENTO"
785 GOTO 340
800 PRINT: PRINT "PSSSSIU.....ESTOU PENSANDO....."
810 LET R$(1)="VAMOS "
811 LET R$(2)=V$(F)
812 LET R$(3)=" "
813 LET R$(4)=N$(H)
814 LET R$(5)=". EU ACHO QUE "
815 LET R$(6)=N$(H)
816 LET R$(7)=" E "
817 LET R$(8)=A$(G)
818 GOTO 340
830 LET R$(1)="FALE SOBRE "
831 LET R$(2)=N$(H)
832 LET R$(3)=", "
833 LET R$(4)=D$
834 GOTO 340
850 LET R$(1)="ACHA QUE SOU "
851 LET R$(2)=A$(G)
852 LET R$(3)=", "
853 LET R$(4)=D$
854 LET R$(5)="?"
855 GOTO 340
870 LET R$(1)="VAMOS "
871 LET R$(2)=V$(F)
872 LET R$(3)=" ALGUMA COISA"
873 LET R$(4)=" MAIS "
874 LET R$(5)=A$(G)
875 GOTO 340
890 LET R$(1)="ADIVINHE EM QUE"
891 LET R$(2)=" ESTOU PENSANDO, "
892 LET R$(3)=D$
900 GOTO 340
910 REM FINAL
920 PRINT "JA CHEGA?"
930 PRINT "ALGUEM AINDA QUER CONVERSAR COMIGO?"
940 INPUT Z$: IF Z$="SIM" THEN GOTO 210
950 PRINT: PRINT "ENTAO TCHAU"
960 END
1000 REM PALAVRAS PARA O COMPUTADOR PROCURAR
1010 FOR I=1 TO 30: READ G$(I): NEXT I
1020 DATA QUEM E,QUE,COMO,?
1030 DATA POR QUE,SEU,MIM,NAO
1040 DATA ?,EU,SABE,ACHO,?
1050 DATA SAO,?,SIM,ACHO,ESTA BEM
1060 DATA SEI LA,?,MESMO?,OBRIGADO
1070 DATA ACHO,VAMOS,?,NAO DIGA
1080 DATA SABE,E VOCE?,POR QUE,E VOCE?
1100 REM RESPOSTAS DO MICRO
1110 FOR I=1 TO 30: READ M$(I): NEXT I
1120 DATA SOU APENAS UM MICRO
1130 DATA NAO IMPORTA,E UMA BOA PERGUNTA
1140 DATA NAO SEI DIREITO,E POR QUE NAO?
1150 DATA QUE QUER DIZER COM ISSO?
1160 DATA QUER EXPLICAR MELHOR?
1170 DATA NAO SEJA TAO NEGATIVO!
1180 DATA VOCE SABE MUITO BEM!
1190 DATA EU TAMBEM,EU NAO
1200 DATA EU NAO ACHO,QUE PERGUNTA BOBA!
1210 DATA TAMBEM ACHO,SEI LA!
1220 DATA AH!ENTAO CONCORDA!,EU TAMBEM
1230 DATA ENTAO COMECE,ENTAO DE UM PALPITE
1240 DATA NAO SEI,MESMO!
1250 DATA DE NADA,OBRIGADO!
1260 DATA ENTAO COMECE.,ADIVINHE
1270 DATA DIGO SIM!,EU NAO!
1280 DATA EU CONCORDO.,PORQUE SIM.,EU NAO.
1300 REM LEITURA DE SUBSTANTIVOS
1310 FOR I=1 TO 10: READ N$(I): NEXT I
```



Estas são as respostas do computador para cada uma das palavras em Q$. As respostas estão na mesma ordem que as palavras. Assim, por exemplo, a quinta frase de M$ é a resposta adequada para a quinta palavra de Q$.

Se a sua resposta é TCHAU, a linha 260 manda o computador para cá.

Estas linhas contêm as palavras que o computador procura na sua frase.

```
1320 DATA O FUTEBOL PROFISSIONAL,A VIOLENCIA URBANA
1330 DATA A EDUCACAO MODERNA,O IDEAL OLIMPICO
1340 DATA A JUVENTUDE DE HOJE,A IMORTALIDADE DA ALMA
1350 DATA A FAMILIA BRASILEIRA,O CAPITALISMO
1360 DATA A MINHA CPU,O COMUNISMO
1400 REM LEITURA DE VERBOS
1410 FOR I=1 TO 10: READ V$(I): NEXT I
1420 DATA PENSAR SOBRE,FALAR SOBRE
1430 DATA DISCUTIR,ANALISAR
1440 DATA EXAMINAR,CONSIDERAR
1450 DATA DISSECAR,AVALIAR
1460 DATA MEDITAR SOBRE,DEBATER
1500 REM LEITURA DE ADJETIVOS
1510 FOR I=1 TO 10: READ A$(I): NEXT I
1520 DATA INTERESSANTE,REVOLTANTE
1530 DATA DEPRIMENTE,ESTIMULANTE
1540 DATA NOTAVEL,DEPLORAVEL
1550 DATA FASCINANTE,NEUROTIZANTE
1560 DATA ASFIXIANTE,REPELENTE
1600 REM LEITURA DE ADVERBIOS E DE
     OUTROS ADJETIVOS, AOS PARES
1610 FOR I=1 TO 10: READ S$(I),T$(I): NEXT I
1620 DATA EXTREMAMENTE,NEUROTICA
1630 DATA REALMENTE,SENSIVEL
1640 DATA ESTRANHAMENTE,TENSA
1650 DATA LEVEMENTE,SUSCETIVEL
1660 DATA PROFUNDAMENTE,DESONESTA
1670 DATA TREMENDAMENTE,TIMIDA
1680 DATA BASICAMENTE,HONESTA
1690 DATA VERDADEIRAMENTE,PSICOPATICA
1700 DATA FORTEMENTE,AFETUOSA
1710 DATA NITIDAMENTE,AMBICIOSA
1720 RETURN
1800 FOR I=1 TO 10
1810 LET R$(I)="": NEXT I
1820 RETURN
```

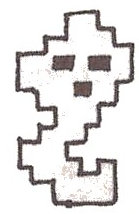
Estes são os substantivos a serem colocados nas frases aleatórias.

Estes são os verbos a serem colocados nas frases aleatórias.

Você pode mudar as palavras à vontade, mas cuidado para não cometer erros de concordância.

Estes são os adjetivos.

Quando escrever sua resposta, não use vírgulas, caso contrário o computador irá ignorar todas as palavras que vierem depois da primeira vírgula.



Cada linha de dados contém um advérbio e um adjetivo. Eles são lidos aos pares para economizar espaço.

Idéias para mudar o programa

1. A maneira mais fácil de alterar o programa é mudar as palavras e frases. Substitua cada palavra nova em todas as frases para ver se faz sentido. Você também pode aumentar o número de palavras, mas para isso terá que mudar o tamanho das matrizes, os loops para ler os dados e os números aleatórios das linhas 610 a 650.
2. Você também pode mudar as palavras em Q$ para fazer o computador reconhecer outras palavras-chave. Não se esqueça de colocar respostas apropriadas nas posições equivalentes de M$.
3. Para fazer o computador usar com maior freqüência o gerador de frases aleatórias, mude o número 6 na linha 320 para um número menor. Você também pode mudar a freqüência com que o computador remove os marcadores de respostas na matriz C. Para isso, mude o número 12 da linha 440.

Rotina para monologar

Você pode fazer o computador conversar consigo mesmo acrescentando as seguintes linhas ao programa:

```
160 PRINT "DIALOGO OU MONOLOGO?
    (APERTE D OU M)"
170 INPUT K$
180 IF K$="M" THEN LET D$=
    "ROM": GOTO 600
470 IF K$="D" THEN GOTO 240
475 IF K$="M" THEN LET I$=R$
480 LET R$="": GOTO 310
```

D$ é a variável onde estava o seu nome. Para monologar, o computador usa a palavra ROM como nome para si mesmo e vai para 600 para gerar uma frase aleatória.

A resposta do computador, R$, se transforma na nova frase, I$. O computador volta à linha 310 para escolher a forma de resposta.

Rotina para responder

Aqui está outra rotina que você pode acrescentar ao programa de conversação. Ele faz o computador responder usando as próprias palavras do interlocutor. As alterações para os micros da família Sinclair estão na página seguinte.

A rotina para responder funciona como a rotina de análise de texto. Existem duas matrizes de dados, U$ e W$. U$ contém palavras que você pode usar e W$ as respostas do computador. Se você usa uma das palavras de U$, a rotina a substitui pela palavra correspondente de W$, mantendo inalterado o resto da frase.

```
135 DIM U$(9),W$(9)
138 DIM Z$(5)
```

Define as matrizes U$ e W$ e outra matriz chamada Z$ para guardar as respostas do computador.

```
155 GOSUB 2200
```

Vai para uma sub-rotina para ler os dados.

```
 325 IF R=7 THEN GOTO 2000
2000 REM ROTINA PARA RESPONDER
2010 LET Z=0
```

Diz ao computador se deve usar a rotina para responder.

Z é uma variável de contagem.

```
2020 LET Q=LEN(I$)
```

Mede o número de caracteres da sua frase.

```
2030 FOR A=1 TO Q
```

Loop para ser executado tantas vezes quantos caracteres tem a sua frase.

```
2040 FOR B=1 TO 9
```

Loop para ser executado quantas vezes quantos caracteres tem a maior palavra de W$.

```
2050 LET L=LEN(U$(B))
```

Cada vez que o loop de B é repetido, o comprimento de uma palavra de U$ é colocado em L.

```
2060 IF MID$(I$,A,L)=U$(B) THEN
     GOTO 2140
```

Compara os caracteres A a L da sua frase com a palavra guardada em U$(B). Se há coincidência, o computador vai para a linha 2140.

```
2070 NEXT B: NEXT A
```

Cada vez que o loop de B é repetido, o computador examina a palavra seguinte de U$. Cada vez que o loop de A é repetido, ele examina um novo grupo de caracteres de I$.

```
2080 IF Z$(1)="" THEN GOTO 600
```

Volta ao gerador de frases aleatórias se não há coincidência.

```
2090 FOR J=1 TO Z
2100 PRINT Z$(J);
2110 LET Z$(J)=""
2120 NEXT J
```

Imprime a resposta que está em Z$.

```
2130 LET R$=I$: GOTO 350
```

Guarda o que resta da sua frase em R$ e volta à linha 350 para imprimir.

```
2140 LET Z=Z+1
```

A variável Z é usada para contar quantas respostas existem em Z$.

```
2150 IF A>1 THEN LET Z$(Z)=LEFT$
     (I$,A-1)+" "+W$(B)+" "
```

Neste ponto, A é o número do primeiro caractere da frase em I$ que coincide com uma das palavras de U$. Se A > 1, a linha 2150 coloca os caracteres à esquerda da palavra em Z$ e depois acrescenta a resposta, obtida em W$.

```
2160 IF A<2 THEN LET Z$(Z)=W$(B)+" "
```

Se A < 2, a palavra fica no começo de I$, de modo que o computador coloca a resposta no começo de Z$.

```
2170 LET I$=MID$(I$,A+L,F)
2180 GOTO 2020
2200 REM DADOS PARA A ROTINA DE RESPONDER
2210 FOR I=1 TO 9
2220 READ U$(I),W$(I)
2230 NEXT I
```

Coloca o resto da sua frase em I$ e volta a 2020 para ver se encontra outra coincidência.

```
2240 DATA EU SOU,VOCE E,VOCE E,EU SOU
2250 DATA EU TENHO,VOCE TEM,VOCE TEM,EU TENHO
2260 DATA MEU,SEU,SEU,MEU
2270 DATA MINHA,SUA,SUA,MINHA
2280 DATA VOCES SAO,NOS SOMOS
2290 RETURN
```

Estes são os dados para U$ e W$.

Rotina para responder nos micros da família Sinclair

O programa abaixo é para o TK-83 e o TK-85. No caso do TK-83, porém, você terá que usar o método que aparece na página 47 para entrar com os dados. Coloque o loop de leitura entre as linhas 1000 e 1720 e as instruções DIM antes da linha 1000.

```
135 DIM U$(9,10)
136 DIM W$(9,10)
137 DIM Z$(5,20)
2042 LET P$=""
2044 FOR I=1 TO LEN(U$(B))
2046 IF U$(B)(I TO I)<>"*"THEN LET P$=P$+U$(B)(I TO I)
2048 NEXT I
2050 LET L=LEN(P$)
2055 IF L+A-1>P THEN GOTO 2070
2060 IF I$(A TO A+L-1)=P$ THEN GOTO 2140
2080 IF Z=0 THEN GOTO 600
2150 IF A>1 THEN LET Z$(Z)=I$(TO A-1)+" "+W$(B)+" "
2160 IF A<2 THEN LET Z$(Z)=W$(B)+" "
2170 LET I$=I$(A+L TO)
2240 DATA "EU SOU****","VOCE E","VOCE E****","EU SOU"
2250 DATA "EU TENHO**","VOCE TEM","VOCE TEM**","EU TENHO"
2260 DATA "MEU*******","SEU","SEU*******","MEU"
2270 DATA "MINHA*****","SUA","SUA*******","MINHA"
2280 DATA "VOCES SAO*","NOS SOMOS"
2290 RETURN
```

Coloca uma palavra de U$ em P$, usando os asteriscos para localizar o final da palavra.

Se P$ é maior que a frase I$, passa para a palavra seguinte.

Se há coincidência, pula para 2140.

Se não há coincidência, vai para 600.

Se A > 1, coloca em Z$ o início de I$ e acrescenta a palavra de W$(B). Se A < 2, limita-se a colocar em Z$ a palavra de W$(B).

Conversão dos programas

Estas duas páginas mostram como converter os programas para que funcionem no TK-83 e no TK-85 e como converter os programas de desenhar para que funcionem em micros que traçam retas a partir do último ponto marcado. Além de inserir as linhas apropriadas, você também terá que fazer as mudanças necessárias para o seu micro, isto é, usar os comandos gráficos corretos, alterar a instrução RND e mudar os nomes das variáveis, se necessário.

Jogo de procurar palavras nos micros da família Sinclair

Mude TESTE$ para T$ e mude as linhas 170 e 200 para:

```
170 LET L$=P$(R TO R)
200 LET T$=T$(2 TO)+L$
```

Base de dados no TK-85

```
10 DIM T$(10,14): DIM A(12)
15 DIM M$(10,12)
120
417
425
```

Como no TK-83

Não se esqueça de colocar entre aspas todos os dados das linhas de DATA.

Base de dados no TK-83

```
 10 DIM T$(10,14)
 12 DIM A(12)
 14 DIM M$(10,12)
120 GOSUB 100+100*C
417 LET L=LEN(Z$)
425 IF Z$=T$(I)(1 TO L) THEN GOTO 440
455 IF VAL(M$(I,J))=1 THEN PRINT "GANHOU
    A COPA DO MUNDO EM ";A(J)
460 IF VAL(M$(I,J))=2 THEN PRINT "FOI
    SEGUNDO LUGAR EM ";A(J)
555 IF VAL(M$(J,I))=1 THEN PRINT T$(J);
    " GANHOU A COPA"
2170 IF C>0 AND C<6 THEN GOTO 2180
2175 PRINT "NUMEROS APENAS ENTRE 1 E 5,
     POR FAVOR"
2176 GOTO 2150
```

O segundo índice é o tamanho da maior string.

Usa C para calcular o número da linha.

Diz ao computador quais os caracteres que deve verificar.

Os dados estão guardados em uma matriz string, de modo que você precisa de VAL para fazer com que o computador tome o valor numérico.

Em vez de entrar com todos os anos, você pode fazer o computador calculá-los, acrescentando as seguintes linhas:

```
1000 FOR I=1 TO 3
1010 LET A(I)=1926+I*4
1020 NEXT I
1030 FOR I=0 TO 8
1040 LET A(I+4)=1950+4*I
1050 NEXT I
```

Substitua as linhas 1100 a 1120 por dez instruções LET, como por exemplo:

```
1100 LET T$(1)="URUGUAI"
1110 LET T$(2)="ARGENTINA"
```

Substitua as linhas 1200 a 1310 por outras dez instruções LET, como por exemplo:

```
1200 LET M$(1)="100100000000"
1210 LET M$(2)="200000000010"
```

Acrescente uma nova linha:

```
1310 RETURN
```

Os micros da família Sinclair

Os micros da família Sinclair podem ser divididos em duas "subfamílias", uma baseada no micro ZX81 e outra no Spectrum. No Brasil, os micros compatíveis com o ZX81 são o TK-83, o CP-200, o Ringo e o AS-1000, enquanto que o único micro nacional compatível com o Spectrum é o TK-85. Assim, tudo que dissemos até agora a respeito do TK-83 também se aplica ao CP-200, Ringo e AS-1000, e tudo que dissemos a respeito dos micros da família Sinclair se aplica a esses micros mais o TK-85.

Programas para desenhar e para fazer gráficos

No caso de micros (como o TK-85) que traçam retas a partir do último ponto marcado na tela e não a partir das margens da tela, substitua as linhas abaixo nos programas para desenhar e para fazer gráficos. (Você terá que substituir DRAW e PLOT por instruções apropriadas para o seu micro.)

Programa para desenhar

```
175 DRAW CX(1)-CX(4),CY(1)-CY(4)
180 FOR I=2 TO 4
185 DRAW CX(I)-CX(I-1),CY(I)-CY(I-1)
530 DRAW LX(FILA+2,I)-LX(FILA,I),
    LY(FILA+2,I)-LY(FILA,I)
```

Para calcular as coordenadas da extremidade das linhas, o computador subtrai as coordenadas do último ponto marcado.

Gráfico de Barras

```
220 PLOT 1,H:DRAW 0,-H+1:DRAW W,0
300 DRAW 0,INT(B(I)*Y)
320 DRAW 0;INT(S(I)*Y)
```

Gráfico de Linhas

```
290 DRAW X,INT((B(N)-B(N-1))*Y)
330 DRAW X,INT((S(N)-S(N-1))*Y)
```

Programa de conversação para o TK-85

Faça as seguintes mudanças para o TK-85:

```
110 DIM V$(10,13):DIM N$(10,22):DIM A$(10,12)
120 DIM R$(10,24):DIM S$(10,15):DIM T$(10,11)
130 DIM M$(30,24):DIM Q$(30,8):DIM C(30)
245 PRINT I$
510 LET P$=""
512 FOR I=1 TO LEN(Q$(P))
514 IF Q$(P)(I TO I)<>" " THEN
    LET P$=P$+Q$(P)(I TO I)
516 NEXT I
```

Guarda a palavra Q$ em P$.

```
530 FOR T=1 TO L2-LEN(P$)+1
540 IF I$(T TO T+LEN(P$)-1)=P$ THEN
    GOTO 60
550 NEXT T:NEXT P:GOTO 600
660 GOTO ((E<7)*(680+E*20))+((E)6)*
    (810+(E-6)*20))
```

Verifica se há coincidência.

Isto é: se E < 7, vá para a linha número 680 + Ex20 e se E > 6 vá para a linha 810 + (E-6)x20.

Programa de conversação para o TK-83

O TK-83 não possui a instrução READ. Por isso, os dados têm que ser introduzidos através de instruções INPUT. Para rodar o programa no TK-83, faça as seguintes mudanças:

1. Faça as mesmas mudanças que para o TK-85, mas coloque as instruções DIM nas linhas 970 a 990.

2. Substitua as linhas de READ e DATA por instruções INPUT, como por exemplo:

```
1000 REM PALAVRAS PARA O
     COMPUTADOR PROCURAR
1010 FOR I=1 TO 30
1020 INPUT Q$(I)
1030 NEXT I
```

3. Mude a linha 1720 para:

```
1720 STOP
```

4. Entre com o programa, depois escreva RUN 970 e entre com todos os dados.

5. Para rodar o programa, escreva GOTO 100. Não aperte RUN para não perder os dados.

6. Agora você pode guardar o programa em fita cassete. Para executá-lo, basta escrever GOTO 100.

Respostas

Problema do robô (página 24)

```
10 INPUT "QUAL E A TEMPERATURA ";TEMP
20 INPUT "QUANTAS HORAS ";H
30 IF TEMP<>25 THEN LET D=H*(100+4*(TEMP-25))
   ELSE LET D=100*H
40 IF D<1 THEN PRINT "FRIO DEMAIS PARA ZAK":
   STOP
50 PRINT "A ";TEMP;" GRAUS, ZAK PODE CORRER ";
   D;" QUILOMETROS EM ";H;" HORAS"
```

O computador primeiro calcula o valor de 4x(TEMP—25), para saber qual o aumento ou redução na velocidade. (Se TEMP é menor que 25, o resultado é negativo.) Somando o resultado a 100, ele calcula a velocidade de Zak em quilômetros por hora e multiplicando por H obtém a distância percorrida.

Estatística do ordenador de Shell (página 34)

```
 90 LET X=0
 95 LET S=0
231 LET X=X+1
271 LET S=S+1
365 PRINT "HOUVE ";X;" COMPARACOES
    E ";S;" TROCAS"
```

Barras mais largas (página 37)

```
285 FOR J=1 TO 8 STEP 2
290 PLOT INT(I*X+J),1
300 DRAW INT(I*X+J),INT(B(I)*Y)
310 PLOT INT(I*X-4-J),1
320 DRAW INT(I*X-4-J),INT(S(I)*Y)
325 NEXT J
```

Os números da linha 285 dependem do seu micro.

Índice Remissivo

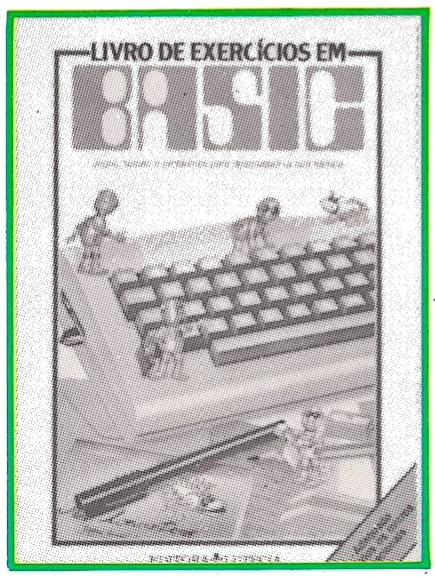
Título original inglês
USBORNE GUIDE TO BETTER BASIC

EDITORA LUTÉCIA LTDA.
Rua Argentina 171 —
20921 Rio de Janeiro, RJ —
Tel.: 580-3668
que se reserva a propriedade
literária desta tradução

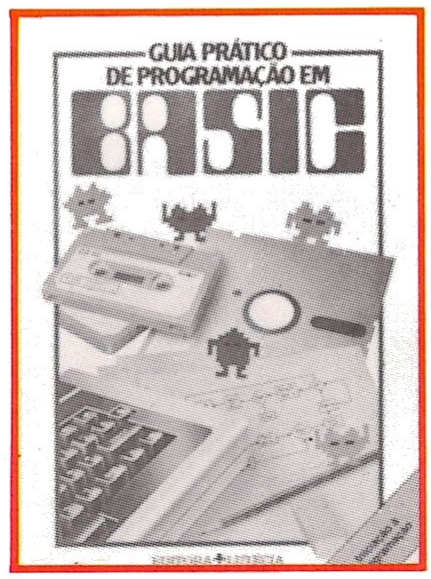
Impresso no Brasil

Guias Práticos de Microcomputadores

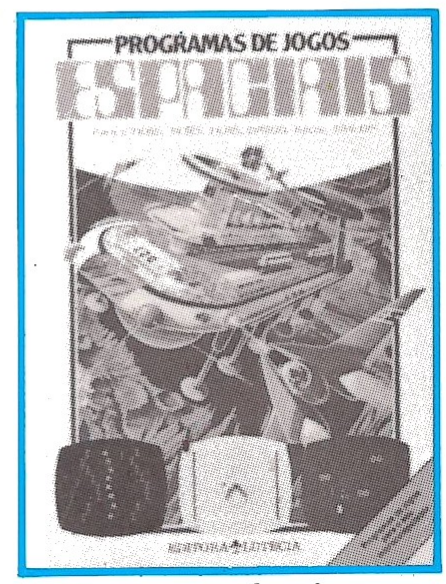
O que é capaz de fazer um microcomputador! Calcular, evidentemente, mas também organizar perguntas, escrever poemas, jogar uma quantidade de jogos cada qual mais palpitante, até mesmo compor música... Pequenos guias práticos de introdução à microinformática, as obras desta nova coleção nos permitem descobrir todas as possibilidades que os microcomputadores nos oferecem. Eles nos iniciam na linguagem e no funcionamento do computador, na aprendizagem da programação e — por que não? — na criação de programas originais: a clareza dos textos, a alegria das cores, a graça dos desenhos, tudo é concebido nestes livros para fazer desta iniciação um prazer.



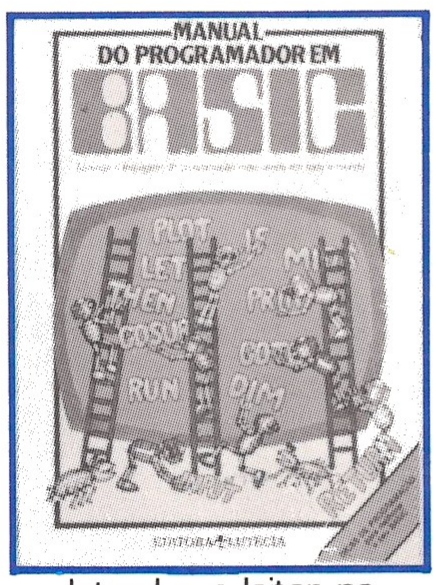
Um colorido guia para melhor compreendermos os microcomputadores — como trabalham e o que fazem, apresentando idéias de coisas que podem ser feitas com o micro. Apresenta um utilíssimo guia comparativo dos diversos micros existentes no mercado brasileiro.



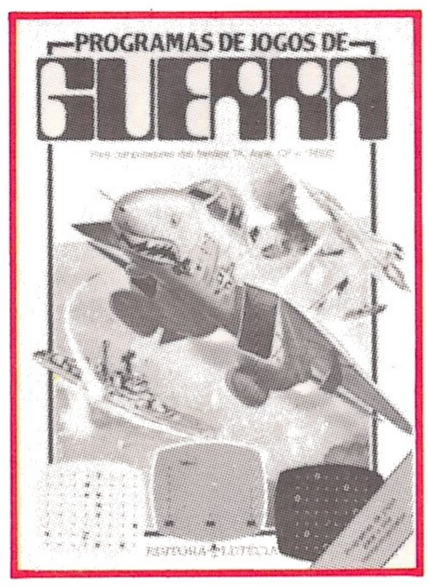
Um guia passo a passo de programação para os absolutamente iniciantes. Apresenta as diferentes linguagens e aprofunda no BASIC, incluindo pequenos programas, simples e divertidos. Profusamente ilustrado a cores.



13 programas fascinantes e ao mesmo tempo simples, para todo tipo de microcomputadores; Jogos Intergaláticos, Módulo Lunar, Viagem ao Futuro, Salvamento no Espaço e muitos outros, com respostas dos problemas.



Introduz o leitor na programação BASIC, através de exemplos didáticos, e é sobretudo indicado ao jovem que começa a programar. Substitui com vantagens os manuais dos fabricantes. Apresenta dezenas de dicas utilíssimas para aprimorar a técnica de programação.



13 listagens de programas de jogos, para serem "rodados" nos micros pessoais mais difundidos no Brasil. Muito útil para ensinar o leitor a desenvolver seus próprios programas. Além disso, é um passatempo fascinante.

Um verdadeiro caderno a cores de problemas e exercícios, com jogos e quebra-cabeças, ideais para todo curso de programação em BASIC. Todas as respostas e explicações estão incluídas, para ajudar a esclarecer todo e qualquer problema