CURSO PRÁTICO 12 DE PROGRAMAÇÃO DE COMPUTADORES

INPUT

PROGRAMAÇÃO BASIC - PROGRAMAÇÃO DE JOGOS - CÓDIGO DE MÁQUINA

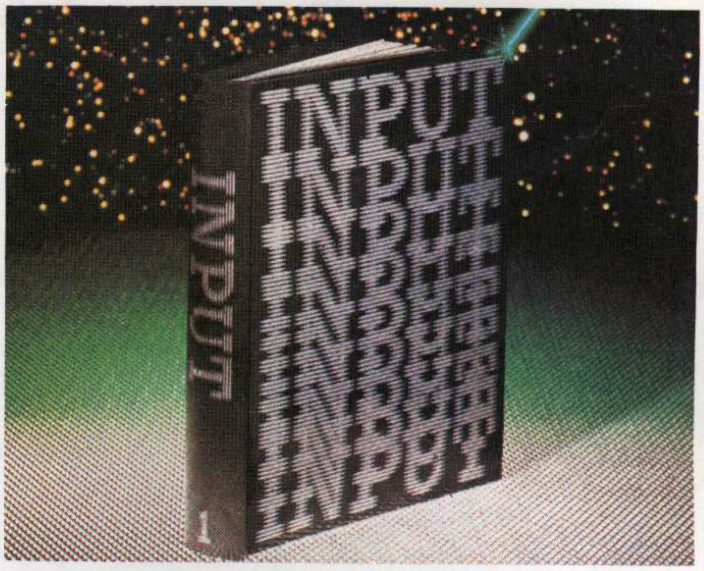
NOVA CULTURAL

Cz$ 20,00

INPUT

Vol. 1 N.º 12

NESTE NÚMERO



PLANO DA OBRA

"INPUT" é uma obra editada em fascículos semanais, e cada conjunto de 15 fascículos compõe um volume. A capa para encadernação de cada volume estará à venda oportunamente.

COMPLETE SUA COLEÇÃO

Exemplares atrasados, até seis meses após o encerramento da coleção, poderão ser comprados, a preços atualizados, da seguinte forma: **1. Pessoalmente** — por meio de seu jornaleiro ou dirigindo-se ao distribuidor local, cujo endereço poderá ser facilmente conseguido junto a qualquer jornaleiro de sua cidade. Em São Paulo os endereços são: Rua Brigadeiro Tobias, 773 (Centro); Av. Industrial, 117 (Santo André); e, no Rio de Janeiro: Rua da Passagem, 93 (Botafogo). **2. Por carta** — Poderão ser solicitados exemplares atrasados também por carta, que deve ser enviada para DINAP — Distribuidor Nacional de Publicações — Números Atrasados — Estrada Velha de Osasco, 132 (Jardim Tereza) — CEP 06000 — Osasco — São Paulo. **3. Por telex** — Utilize o nº (011) 33670 ABSA. Em **Portugal**, os pedidos devem ser feitos à Distribuidora Jardim de Publicações Ltd. — Qta. Pau Varais, Azinhaga de Fetais — 2685, Camarate — Lisboa; Tel. 257-2542 — Apartado 57 — Telex 43 069 JARLIS P.
Não envie pagamento antecipado. O atendimento será feito pelo reembolso postal e o pagamento, incluindo as despesas postais, deverá ser efetuado ao se retirar a encomenda na Agência do Correio. **Atenção**: Após seis meses do encerramento da coleção, os pedidos serão atendidos, dependendo da disponibilidade de estoque. **Obs.:** Quando pedir livros, mencione sempre o título e/ou o autor da obra, além do número da edição.

COLABORE CONOSCO

Encaminhe seus comentários, críticas, sugestões ou reclamações ao Serviço de Atendimento ao Leitor — Caixa Postal 9442, São Paulo — SP.

Editor
VICTOR CIVITA

REDAÇÃO

Diretora Editorial: Iara Rodrigues

Editor chefe: Paulo de Almeida
Editor de texto: Cláudio A.V. Cavalcanti
Editor de Arte: Eduardo Barreto
Chefe de Arte: Carlos Luiz Batista
Assistentes de Arte: Ailton Oliveira Lopes, Dilvacy M. Santos, José Maria de Oliveira, Grace A. Arruda, Monica Lenardon Corradi
Secretária de Redação/Coordenadora: Stefania Crema
Secretários de Redação: Beatriz Hagström, José Benedito de Oliveira Damião, Maria de Lourdes Carvalho, Marisa Soares de Andrade, Mauro de Queiroz
Secretário Gráfico: Antonio José Filho

COLABORADORES
Consultor Editorial Responsável: Dr. Renato M.E. Sabbatini (Diretor do Núcleo de Informática Biomédica da Universidade Estadual de Campinas)
Execução Editorial: DATAQUEST Assessoria em Informática Ltda. Campinas, SP.

Tradução: Maria Fernanda Sabbatini

Adaptação, programação e redação: Abílio Pedro Neto, Aluísio J. Dornellas de Barros, Marcelo R. Pires Therezo, Raul Neder Porrelli,
Coordenação geral: Rejane Felizatti Sabbatini
Assistente de Arte: Dagmar Bastos Sampaio

COMERCIAL
Diretor Comercial: Roberto Martins Silveira
Gerente Comercial: Flávio Ferrucio Maculan
Gerente de Circulação: Denise Maria Mozol

PRODUÇÃO
Gerente de Produção: João Stungis
Coordenador de Impressão: Atilio Roberto Bonon
Preparador de Texto/Coordenador: Eliel Silveira Cunha

Preparadores de Texto: Ana Maria Dilguerian, Antonio Francelino de Oliveira, Karina Ap. V. Grechi, Levon Yacubian, Maria Teresa Galluzzi, Paulo Felipe Mendrone
Revisor/Coordenador: José Maria de Assis
Revisoras: Conceição Aparecida Gabriel, Isabel Leite de Camargo, Ligia Aparecida Ricetto, Maria do Carmo Leme Monteiro, Maria Luiza Simões, Maria Teresa Martins Lopes.

Edição organizada pela Editora Nova Cultural Ltda.
(Artigo 15 da Lei 5 988, de 14/12/1973).
Esta obra foi composta na AM Produções Gráficas Ltda. e impressa na Divisão Gráfica da Editora Abril S.A.

COMO ESTRUTURAR SEUS PROGRAMAS

- O QUE É PROGRAMAÇÃO ESTRUTURADA?
- OS FLUXOGRAMAS
- COMO ESTRUTURAR PROGRAMAS EM BASIC

A estruturação de um programa é uma técnica que facilita a compreensão de seu funcionamento e permite tornar mais eficiente e simples sua operação. Aprenda a utilizá-la em BASIC.

O interpretador BASIC padrão, disponível na maioria dos micros, é simples de usar, mas possui poucas *estruturas de programação*, o que dificulta a elaboração de programas estruturados.

As estruturas são "blocos de construção" da lógica de um programa, e são utilizadas para proporcionar-lhe uma forma coerente e de fácil compreensão. No BASIC padrão, as principais estruturas de programação lógica são representadas pelas declarações **IF...THEN**, **FOR...NEXT**, **GOTO** e **GOSUB**. Já vimos como empregar individualmente a maioria delas; examinaremos agora como colocá-las juntas de uma maneira legível e ordenada.

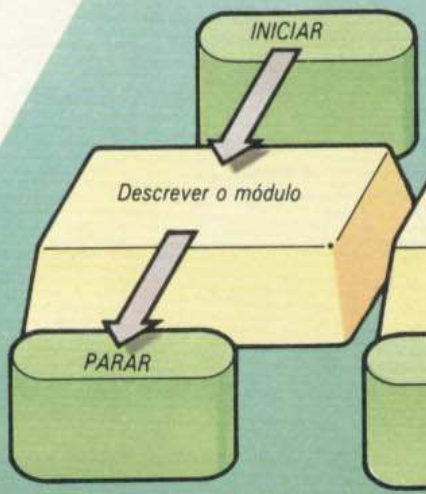
Em programas curtos, a estruturação não é uma questão muito importante. O problema aparece quando se pretende elaborar um programa longo e complexo, do ponto de vista lógico (desvios). Nesse caso, se você escrever uma parte do programa e depois acrescentar mais e mais coisas a ele, provavelmente acabará se perdendo.

Para evitar que isso aconteça, habitue-se a projetar detalhadamente seu programa, antes de começar a digitá-lo. Comece por escrever o que pretende que o computador faça, de um modo bem geral. Por exemplo:

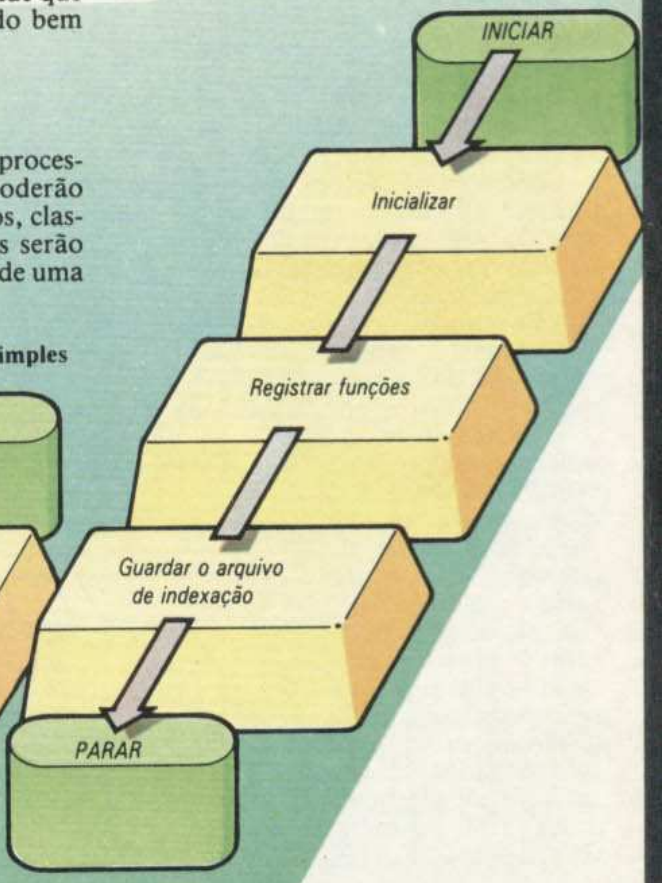
Sistema de indexação

O programa tem por objetivo processar em memória registros que poderão ser criados, atualizados, apagados, classificados e listados. Os registros serão acessíveis por meio da utilização de uma

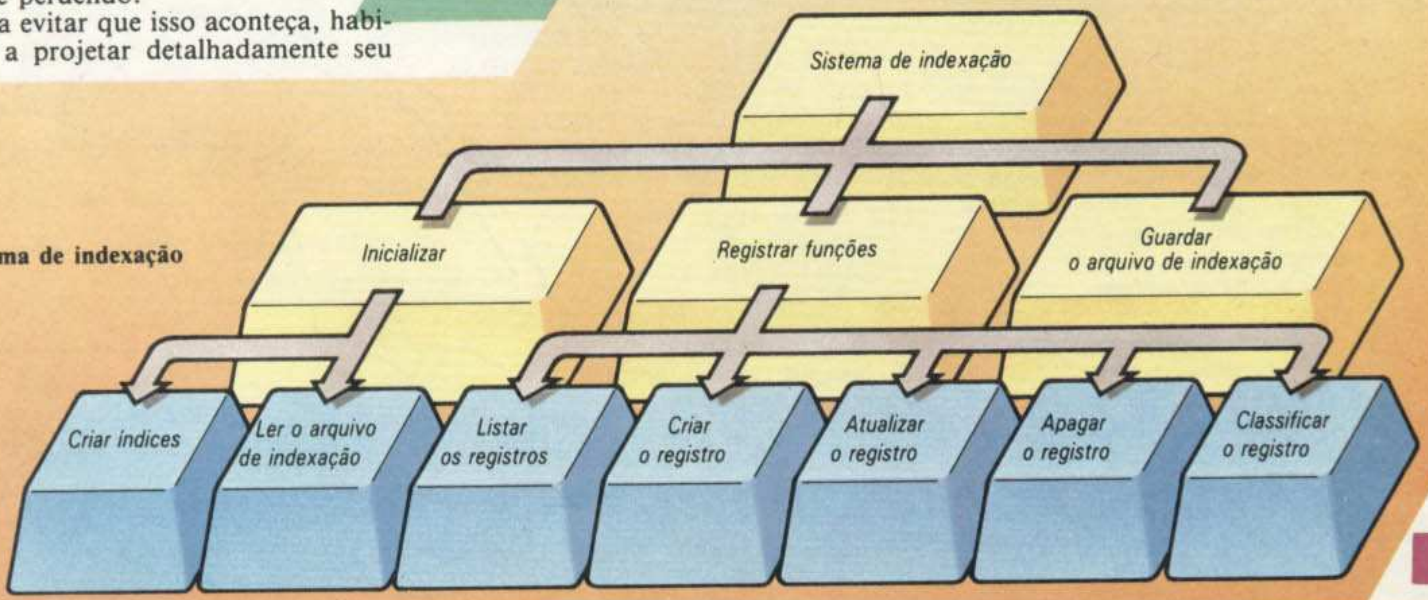
2. Módulo simples



3. Série de módulos



1. Sistema de indexação

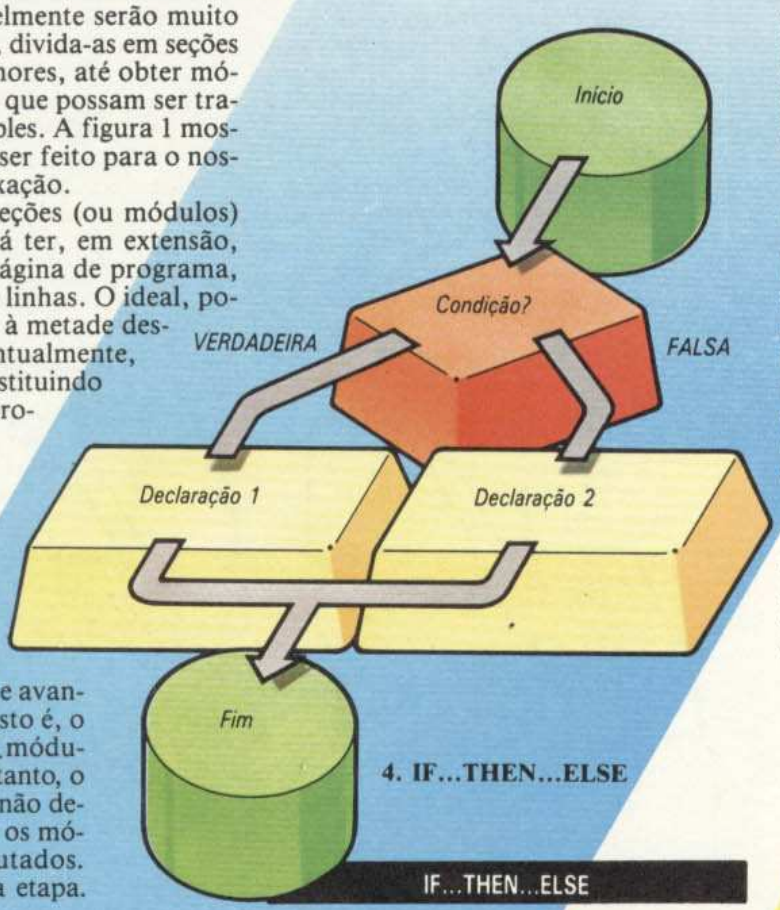


palavra-chave. Todo o conjunto de registros poderá ser armazenado e carregado em um arquivo.

Este primeiro passo é conhecido como especificação do sistema. O passo seguinte consistirá em dividir as especificações gerais, acima, em etapas lógicas ou módulos. Estes, por sua vez, também terão suas funções especificadas de maneira geral. As operações envolvidas em cada módulo provavelmente serão muito complicadas; assim, divida-as em seções sucessivamente menores, até obter módulos tão pequenos que possam ser tratados de forma simples. A figura 1 mostra como isso pode ser feito para o nosso sistema de indexação.

Cada uma das seções (ou módulos) menores não deverá ter, em extensão, mais do que uma página de programa, ou seja, cerca de 60 linhas. O ideal, porém, será limitá-las à metade deste tamanho. Eventualmente, elas terminarão constituindo sub-rotinas do seu programa.

Esse método de subdividir um problema é conhecido como *modelo de cima para baixo (top-down)*. Começa-se do alto (ou seja, da descrição geral do programa) e avança-se até embaixo (isto é, o nível mais baixo de módulos). Até aqui, entretanto, o programador ainda não decidiu em que ordem os módulos serão executados. Esta será a próxima etapa.



4. IF...THEN...ELSE

IF...THEN...ELSE

Embora os diversos computadores utilizem de maneira diferente a declaração **IF...THEN...ELSE**, ela é sempre a base da tomada de decisões dentro de um programa. Representada no fluxograma por um losango (veja figura 4), esta declaração é escrita em BASIC da seguinte maneira:

```
100 IF condição THEN
    declaração 1 ELSE
    declaração 2.
```

Ou seja: se a condição for verdadeira, será executada a declaração 1; caso contrário, a declaração 2.

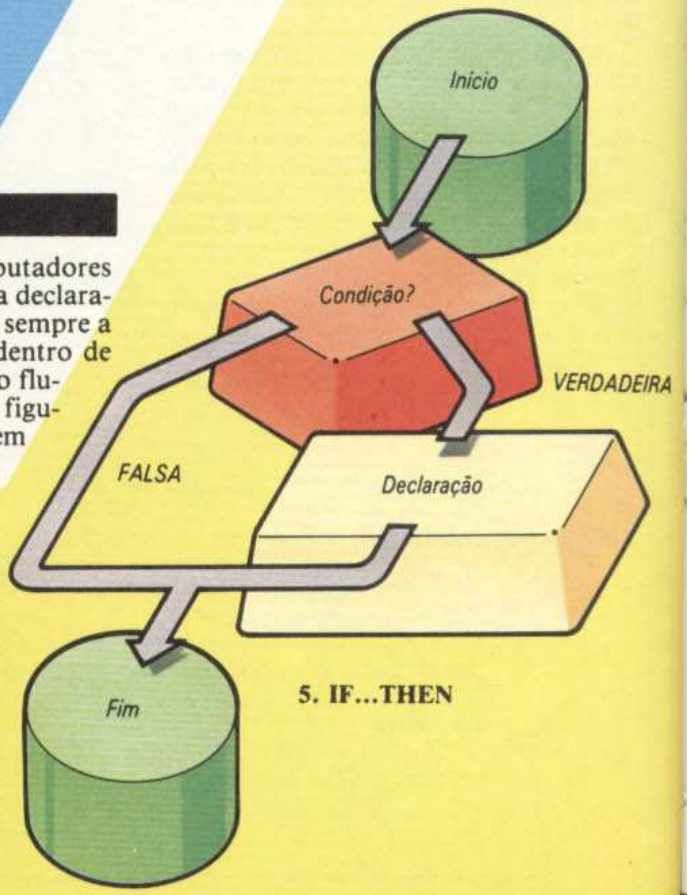
Note que, no fluxograma, só há um ponto de saída para a estrutura **IF...THEN...ELSE**. De fato, cada seção de código deve ter apenas uma entrada e uma saída. Esta regra, muito importante para a programação estruturada, auxilia bastante a testar e a eliminar as falhas.

Nem todas as versões em BASIC, porém, possuem a partícula **ELSE** — ela não está disponível, por exemplo, nos computadores Spectrum, ZX-81, Apple ou TK-2000. Nesses casos, para evitar que a estrutura passe a ter mais de uma saída, o **ELSE** pode ser simulado utilizando-se o **GOTO**:

```
100 ...
110 IF condição THEN GOTO 140
120 declaração 2
130 GOTO 150
140 declaração 1
150 ...
```

É claro que os números das linhas não precisam ser iguais aos do exemplo. Além disso, pode-se incluir mais de uma declaração nas partículas **THEN** e **ELSE**. Veja, a seguir, uma seção de programa para classificar dois números dentro de determinada ordem. Ela constitui a base para uma rotina de classificação alfabética que será dada adiante. No caso específico, a partícula **ELSE** possui quatro declarações:

```
100 IF  primeiro<=segundo  THEN
    GOTO 160
110 LET temporario=primeiro
120 LET primeiro=segundo
130 LET segundo=temporario
140 LET ordem$="errado"
```



5. IF...THEN

DIAGRAMAS DE BLOCOS

A ordem (ou lógica) de execução dos módulos de um programa pode ser especificada por meio dos *diagramas de blocos* ou *fluxogramas*. O uso destes permite que se ordene o programa de uma maneira bem estruturada e clara. Para isto, basta seguir um conjunto simples de regras.

A figura 2 mostra como especificar um módulo. O programa acompanha as linhas na direção das flechas e os retângulos descrevem o que acontece em cada estágio. Se quiser executar uma série de módulos em seqüência, acrescente mais retângulos na ordem correta, entre o início e o fim (veja a figura 3).

Um fluxograma como este é ideal para um programa no qual nenhuma decisão é tomada. No entanto, o poder do

```
150 GOTO 170
160 LET ordem$="certo"
170 ...
```

Esta seção também pode ser escrita utilizando-se o **ELSE**, mas ele deverá estar totalmente contido em apenas uma linha. As declarações múltiplas serão permitidas desde que estejam separadas por dois pontos:

```
100 IF primeiro>segundo THEN
    temporario=primeiro :
    primeiro=segundo: segundo=
    temporario:ordem$="errado"
    ELSE ordem$="certo"
```

Não é fácil ler ou entender programas que utilizam declarações longas como esta — assim, sempre que possível, deve-se evitá-las.

Finalmente, em alguns casos, não será necessário usar partícula **ELSE**. O fluxograma se parecerá com o da figura 5 e será redigido desta maneira:

```
100 IF condição THEN declara-
    ção.
```

que é, simplesmente, a própria declaração **IF...THEN**.

ESTRUTURAS EMBUTIDAS

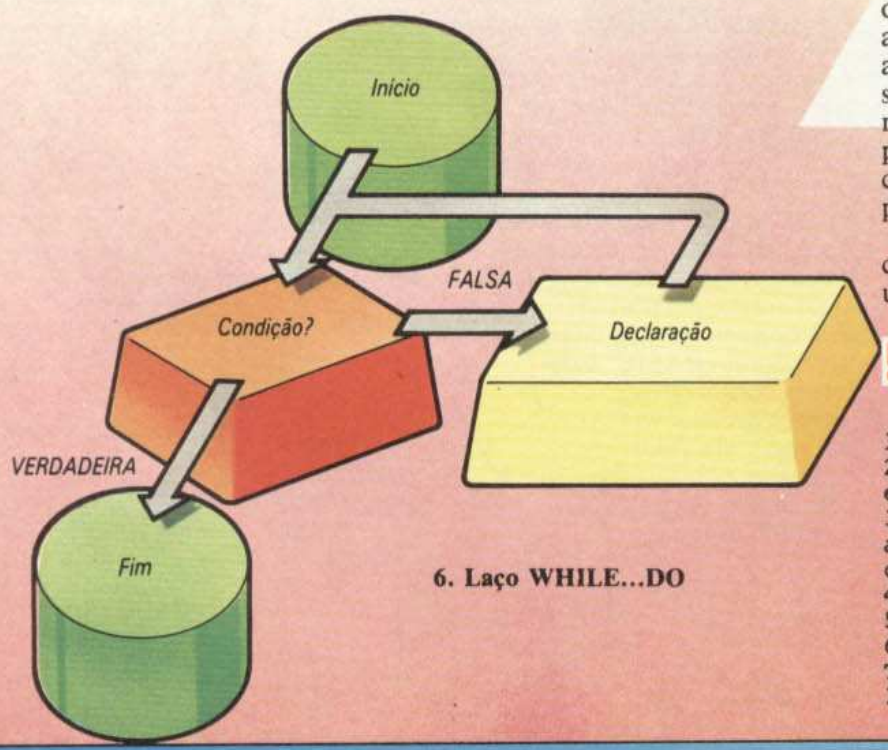
É possível embutir uma linha **IF... THEN...ELSE** dentro de outra declaração **IF...THEN...ELSE**. Ou seja, a conseqüência de um **THEN** ou de um **ELSE** também poderá ser um outro **IF**, e assim por diante. É o caso da seção de programa abaixo, que executa a contagem de quantas partidas dois jogadores ganharam e imprime os resultados após cada jogo:

```
100 IF T1<>T2 THEN GOTO 130
110 PRINT"EMPATOU!"
120 GOTO 190
130 IF T1<T2 THEN GOTO 170
140 PRINT"O JOGADOR 1 GANHA"
150 LET P1=P1+1
160 GOTO 190
170 PRINT"O JOGADOR 2 GANHA"
180 LET P2=P2+1
190 ...
```

Todas as estruturas podem ser embutidas em qualquer combinação e, na teoria, em qualquer profundidade. No entanto, quanto mais você as embute, menos legível se torna o programa; assim, é razoável estabelecer um limite de profundidade de três ou quatro estruturas. Se precisar embutir mais estruturas, subdivida o programa em módulos ou subrotinas menores.

Examinando o último programa, você observará que é difícil acompanhá-lo, ainda que esteja perfeitamente estruturado. Uma maneira prática de tornar mais legíveis as declarações embutidas é recuar (indentar) as linhas do programa. Isso só é viável nos computadores das linhas TRS-80, TRS-Color, Spectrum e MSX; neles, você poderá reescrever o último programa da forma que se segue:

```
100 IF T1<>T2 THEN GOTO 130
110     PRINT"EMPATOU!"
120 GOTO 190
130 IF T1<T2 THEN GOTO 170
140     PRINT"O JOGADOR 1 GANHA"
150     LET P1=P1+1
160 GOTO 190
170     PRINT"O JOGADOR 2 GANHA"
180     LET P2=P2+1
190 ...
```



6. Laço WHILE...DO

A introdução de linhas em branco em diferentes seções e a utilização de declarações **REM** constituem outras maneiras de tornar mais clara a estrutura de um programa. Para entrar as linhas em branco nos computadores Spectrum, basta digitar o número da linha seguida de um espaço e, em seguida, pressionar **<ENTER>** ou **<RETURN>**. Ao escrever programas no TRS-80, TRS-Color ou MSX, você poderá obter um efeito semelhante digitando um apóstrofo (’), ao invés de dar o espaço.

WHILE...DO

Outra declaração importante em programação estruturada é a **WHILE...DO**, que permite a repetição em um programa e é um dos recursos mais úteis para se criar um laço. Esta declaração diz ao computador que faça **(DO)** repetidamente uma coisa enquanto **(WHILE)** uma certa condição for verdadeira. Se seu computador não possui as palavras **WHILE** e **DO**, você pode inventar uma estrutura que faça a mesma coisa, utilizando **IF...THEN** e **GOTO**. A redação, em BASIC, é a seguinte (veja o fluxograma correspondente na figura 6):

```
100 ...
110 IF NOT condição THEN GOTO
    140
120 declaração
130 GOTO 110
140 ...
```

Note que a linha 100 lê a condição **IF NOT...**, ou seja, verifica se a condição não é verdadeira. O procedimento é oposto do normal, ao qual você está acostumado, mas não tem problema. Se a condição for **A=B** então **NOT (A=B)**, será equivalente a **A<>B**. Do mesmo modo, **NOT (A<B)** será **A>=B** e assim por diante. Na verdade, você poderá escrever **NOT (A=B)** se quiser, e o computador entenderá o que você quer dizer.

Eis aqui o exemplo de um programa curto (para cronometrar o cozimento de um ovo) que utiliza um laço **WHILE**:

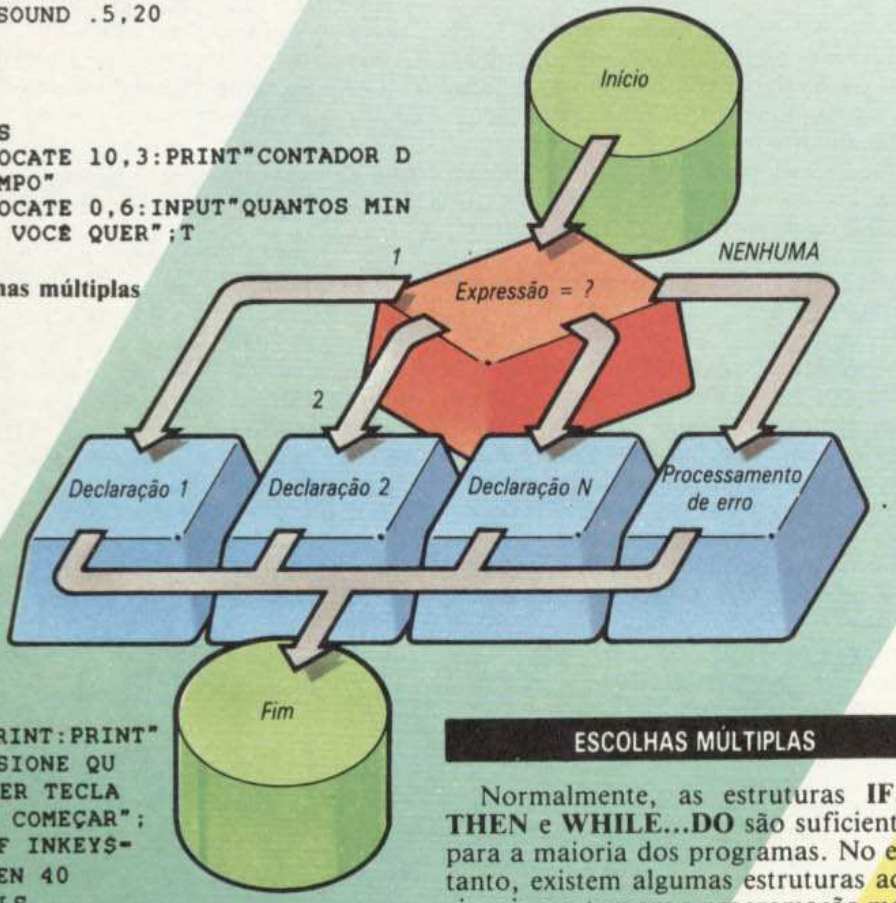
```
 5 CLS
10 PRINT AT 3,11;"CONTADOR"
20 INPUT "Quantos minutos voc
e quer?",t
30 PRINT AT 7,5;"Pressione qu
alquer tecla        para come
car"
40 PAUSE 0
50 CLS
60 PRINT  FLASH 1;AT 10,10;"
T E M P O "
70 POKE 23672,0: POKE 23673,0
```

```
 80 LET time=PEEK 23672+256*
 PEEK 23673: IF time>t*50*60
 THEN  GOTO 110
 90 PRINT AT 14,10;INT (time/
 50);" segundos"
100 GOTO 80
105 REM fim do loop WHILE
110 PRINT  FLASH 1;AT 14,10;"
TERMINADO! "
120 SOUND .5,20
```

```
5 CLS
10 LOCATE 10,3:PRINT"CONTADOR D
E TEMPO"
20 LOCATE 0,6:INPUT"QUANTOS MIN
UTOS VOCÊ QUER";T
30 PRINT:PRINT"
PRESSIONE QU
ALQUER TECLA
PARA COMEÇAR";
40 IF INKEY$=
""THEN 40
50 CLS
60 LOCATE 10,15:PRINT"CONTANDO.
.."
70 TIME=0
75 REM COMEÇA O LOOP WHILE
80 IF TIME>T*3600 THEN GOTO 110
90 LOCATE 12,17:PRINTINT(TIME/6
0);"seg."
100 GOTO 80
105 REM FIM DO LOOP WHILE
110 LOCATE 10,19:PRINT"ESTA PRO
NTO!"
120 BEEP
```

```
5 CLS
10 PRINT @43,"CONTADOR"
15 LET E$=CHR$(27)
20 PRINT @129,;:INPUT"QUANTOS M
INUTOS VOCE QUER ?";T
30 PRINT @196,"PRESSIONE QUALQU
ER TECLA          PARA COMECAR"
40 A$=INKEY$:IF A$="" THEN GOTO
 40
50 CLS
60 PRINT @238,"TEMPO"
70 TIMER = 0
75 REM COMECO DO LACO "WHILE"
80 IF TIMER>T*3000 THEN GOTO 11
0
90 PRINT @298,INT(TIMER/50);"SE
GUNDOS"
100 GOTO 80
105 REM FIM DO LACO "WHILE"
110 PRINT @364,"TERMINADO!"
120 SOUND 180,3
```

7. Escolhas múltiplas



ESCOLHAS MÚLTIPLAS

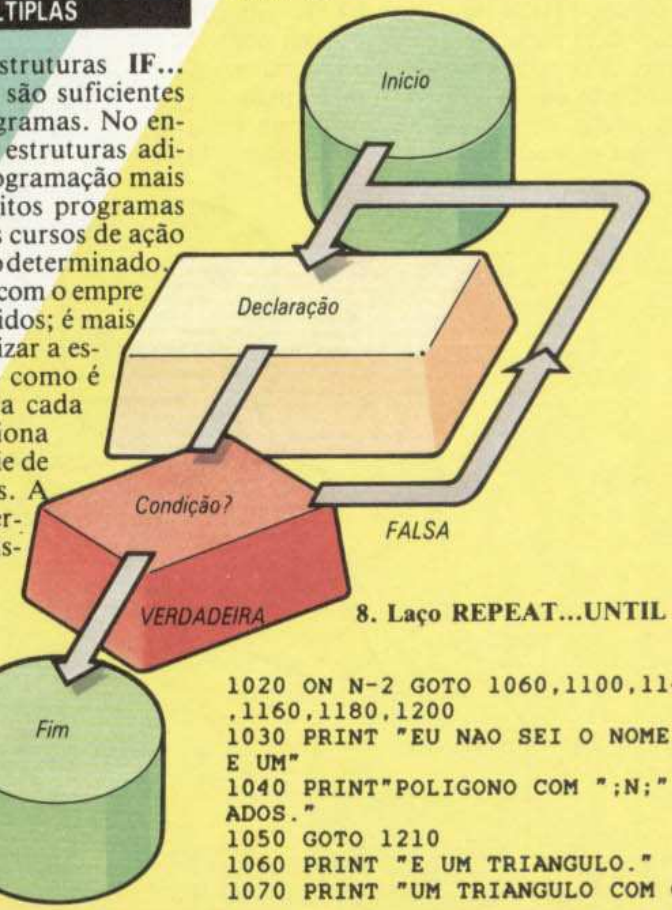
Normalmente, as estruturas **IF... THEN** e **WHILE...DO** são suficientes para a maioria dos programas. No entanto, existem algumas estruturas adicionais que tornam a programação mais fácil. Por exemplo, muitos programas apresentam mais de dois cursos de ação possíveis para um ponto determinado. Isso poderia ser resolvido com o emprego de **IF...THEN** embutidos; é mais conveniente, porém, utilizar a estrutura **CASE** (caso) — como é conhecida. Ela classifica cada opção em ordem e direciona o computador a uma série de cursos de ação possíveis. A grande maioria dos interpretadores BASIC existentes para micros não inclui a declaração **CASE**, mas ela pode ser simulada por outros comandos padrão.

O fluxograma para uma estrutura **CASE** é mostrado na figura 7 e, em BASIC, um programa típico seria:

```
100 REM COMANDO CASE
110 IF C$="C" THEN GOTO 170
120 IF C$="M" THEN GOTO 190
130 IF C$="A" THEN GOTO 210
140 IF C$="L" THEN GOTO 230
150 PRINT "comando nao reconhec
ido"
160 GOTO 240
170 PRINT "CRIAR REGISTRO":GOSU
B 1000
180 GOTO 240
190 PRINT "MODIFICAR REGISTRO":
GOSUB 2000
200 GOTO 240
210 PRINT "APAGAR REGISTRO":GOS
UB 3000
220 GOTO 240
230 PRINT "LISTAR REGISTROS":GO
SUB 4000
240 REM FIM DO CASE
```

Pode-se também fazer múltiplas escolhas por meio das declarações **ON...GOTO** e **ON...GOSUB**. Entretanto, quando utilizar **ON...GOTO**, verifique se cada uma das opções direciona você ao final da seção. Na sub-rotina abaixo, por exemplo, foi necessário usar um **GOTO 1210** após cada opção, para direcionar o programa para o fim da rotina. Com uma declaração **CASE** tal procedimento seria desnecessário.

```
1000 REM SUBROTINA PARA POLIGON
OS
1010 INPUT"QUANTOS LADOS VOCE Q
UER?";N
```



8. Laço REPEAT...UNTIL

```
1020 ON N-2 GOTO 1060,1100,1140
,1160,1180,1200
1030 PRINT "EU NAO SEI O NOME D
E UM"
1040 PRINT"POLIGONO COM ";N;" L
ADOS."
1050 GOTO 1210
1060 PRINT "E UM TRIANGULO."
1070 PRINT "UM TRIANGULO COM OS
```

```
 LADOS IGUAIS CHAMA-SE"
1080 PRINT "TRIANGULO EQUILATER
O."
1090 GOTO 1210
1100 PRINT "E UM QUADRILATERO."
1110 PRINT "UM QUADRILATERO COM
 OS LADOS"
1120 PRINT "E ANGULOS IGUAIS CH
AMA-SE QUADRADO."
1130 GOTO 1210
1140 PRINT "E UM PENTAGONO."
1150 GOTO 1210
1160 PRINT "E UM HEXAGONO."
1170 GOTO 1210
1180 PRINT "E UM HEPTAGONO."
1190 GOTO 1210
1200 PRINT "E UM OCTOGONO."
1210 PRINT
1220 RETURN
```

Como se trata de uma sub-rotina, você ainda não poderá rodá-la. O programa para chamar a rotina será dado na próxima seção.

REPEAT...UNTIL

A declaração **REPEAT... UNTIL** também é uma estrutura muito útil para fazer o programa repetir um grupo de comandos (se seu computador não possui esses comandos, poderá simulá-los por meio de outros). Nesse caso, ao contrário do **WHILE...DO**, o laço sempre é executado pelo menos uma vez. Veja o símbolo correspondente no fluxograma da figura 8 e compare-o à figura 6. Em BASIC, será escrito assim:

```
110 declaração
120 IF  NOT condição THEN  GOTO
    110
130 ...
```

Utilizando a sub-rotina do último exemplo, você poderá escrever um programa utilizando um laço do tipo **REPEAT**, como o seguinte:

```
10 PRINT "VOU DIZER OS NOMES"
20 PRINT "DE ALGUNS POLIGONOS."
30 REM COMECO DO LACO
40 GOSUB 1000
50 INPUT "VOCE QUER OUTRO NOME
?";A$
60 IF LEFT$(A$,1)="S" THEN GOTO
 30
70 PRINT "TCHAU !":END
```

A linha 1000 é a sub-rotina de polígono dada na seção anterior.

LAÇOS FOR...NEXT

Talvez você não tenha percebido, mas o familiar laço **FOR...NEXT** é apenas um caso especial do laço **WHILE... DO**. Ele pode ser utilizado quando se conhece antecipadamente o número de vezes que o laço deverá ser repetido, pois esta informação deve ser especificada logo no início. A variável que mantém a contagem do número de vezes ao redor do laço é conhecida como *variável de controle*.

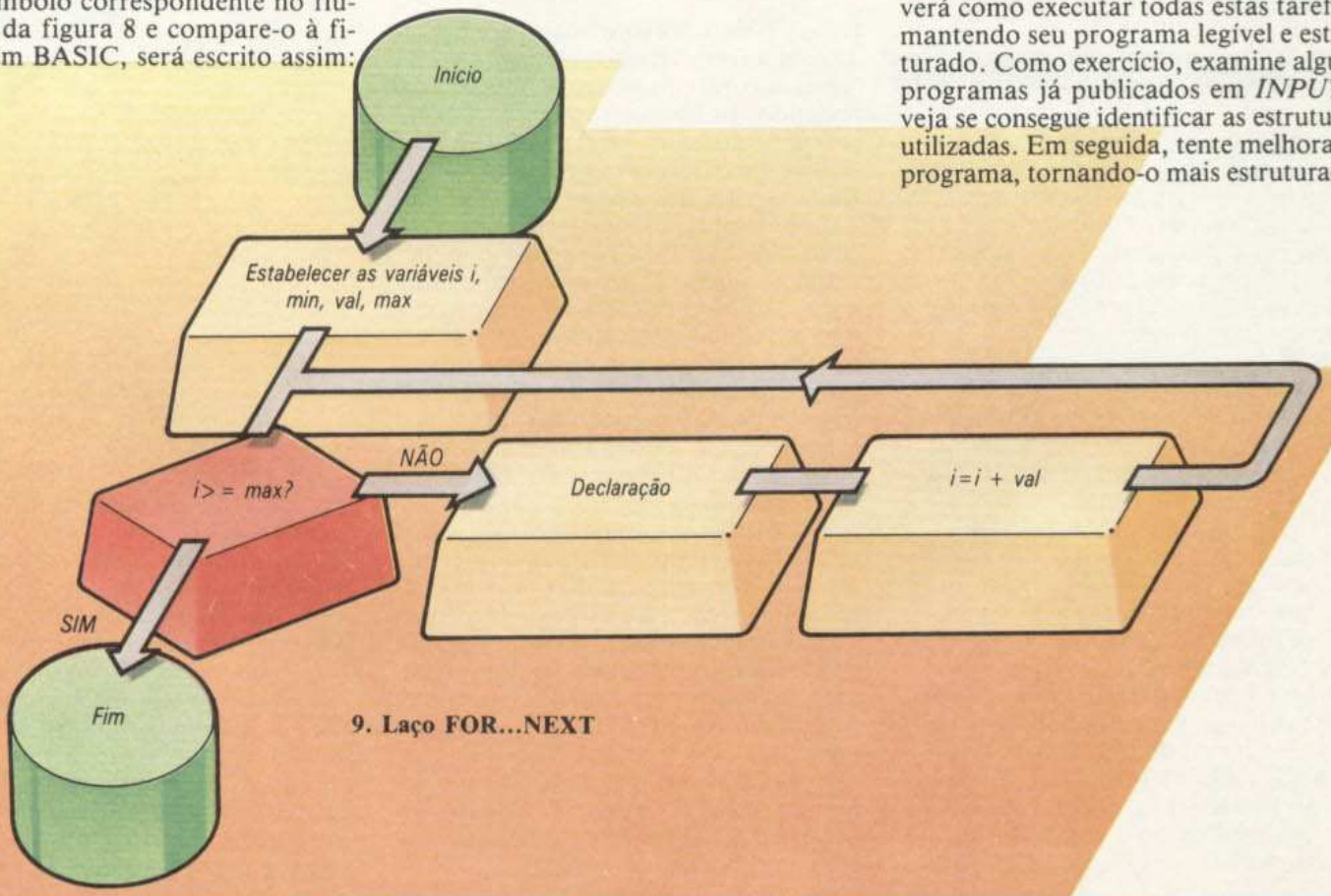
Um fluxograma para o laço **FOR... NEXT** é semelhante ao que se vê na figura 9. Comparando-o com o laço **WHILE** da figura 6, observa-se que tem a mesma estrutura geral. Em BASIC é descrito assim:

```
100 FOR i=min TO max STEP val
110 declaração
120 NEXT i
```

Não se deve saltar para fora de um laço **FOR**, utilizando uma declaração **GOTO**, já que o interpretador BASIC não tem como identificar o que foi feito. Será fácil saltar de volta dentro de um laço, no programa, mas a compreensão de tal estrutura é muito difícil. Assim, evite este procedimento.

COMO COLOCAR TUDO JUNTO

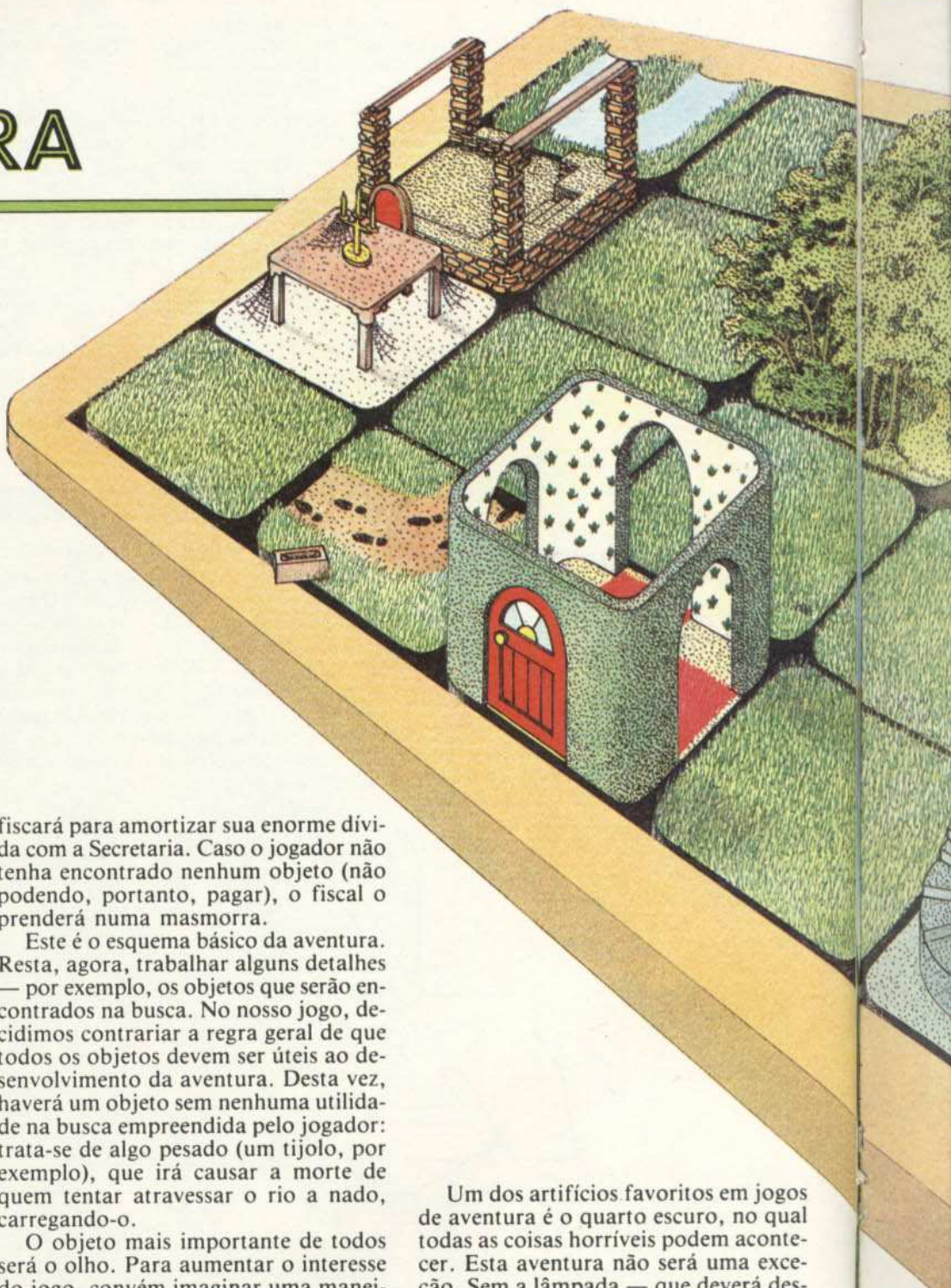
As estruturas vistas neste artigo são as mais conhecidas e importantes, e podem ser utilizadas em qualquer tipo de programa. Mas, para completar o programa, você deverá ainda especificar as variáveis que irá utilizar e verificar se as variáveis dos diferentes módulos não entrarão em choque umas com as outras. Além disso, será preciso especificar todas as entradas e saídas necessárias, testar cada um dos módulos e encadeá-los todos juntos. No próximo artigo você verá como executar todas estas tarefas, mantendo seu programa legível e estruturado. Como exercício, examine alguns programas já publicados em *INPUT* e veja se consegue identificar as estruturas utilizadas. Em seguida, tente melhorar o programa, tornando-o mais estruturado.



9. Laço FOR...NEXT

O MAPA DA AVENTURA

Acompanhe a criação de uma aventura e, depois, crie seu próprio jogo. Aqui, os primeiros passos: o desenvolvimento do roteiro e a confecção do mapa dos locais.



Antes de programar um jogo, é importante elaborar minuciosamente o roteiro, para evitar dificuldades posteriores, como erros e pontos obscuros a serem esclarecidos.

Para que você se familiarize com esta etapa do trabalho, acompanhe conosco o desenvolvimento de um programa de aventuras típico. A história que tomamos como exemplo se passa em um país distante, onde o jogador deve encontrar o olho perdido de um legendário totem inca. Se você seguir passo a passo as etapas percorridas para a montagem desta aventura, sentirá maior facilidade ao projetar seu próprio jogo.

O ROTEIRO

Em primeiro lugar o programador precisa criar um "mundo" que combine com a estrutura básica do roteiro. Este mundo compõe-se de uma série de objetos, cada qual destinado a desempenhar um papel específico. Além disso, deve criar alguns mistérios e problemas para o jogador resolver.

Não é necessário fazer tudo isso de uma vez: à medida que se pensa na história, seus contornos vão se tornando mais nítidos, o que facilita a definição dos detalhes. Comece, portanto, rascunhando o roteiro básico.

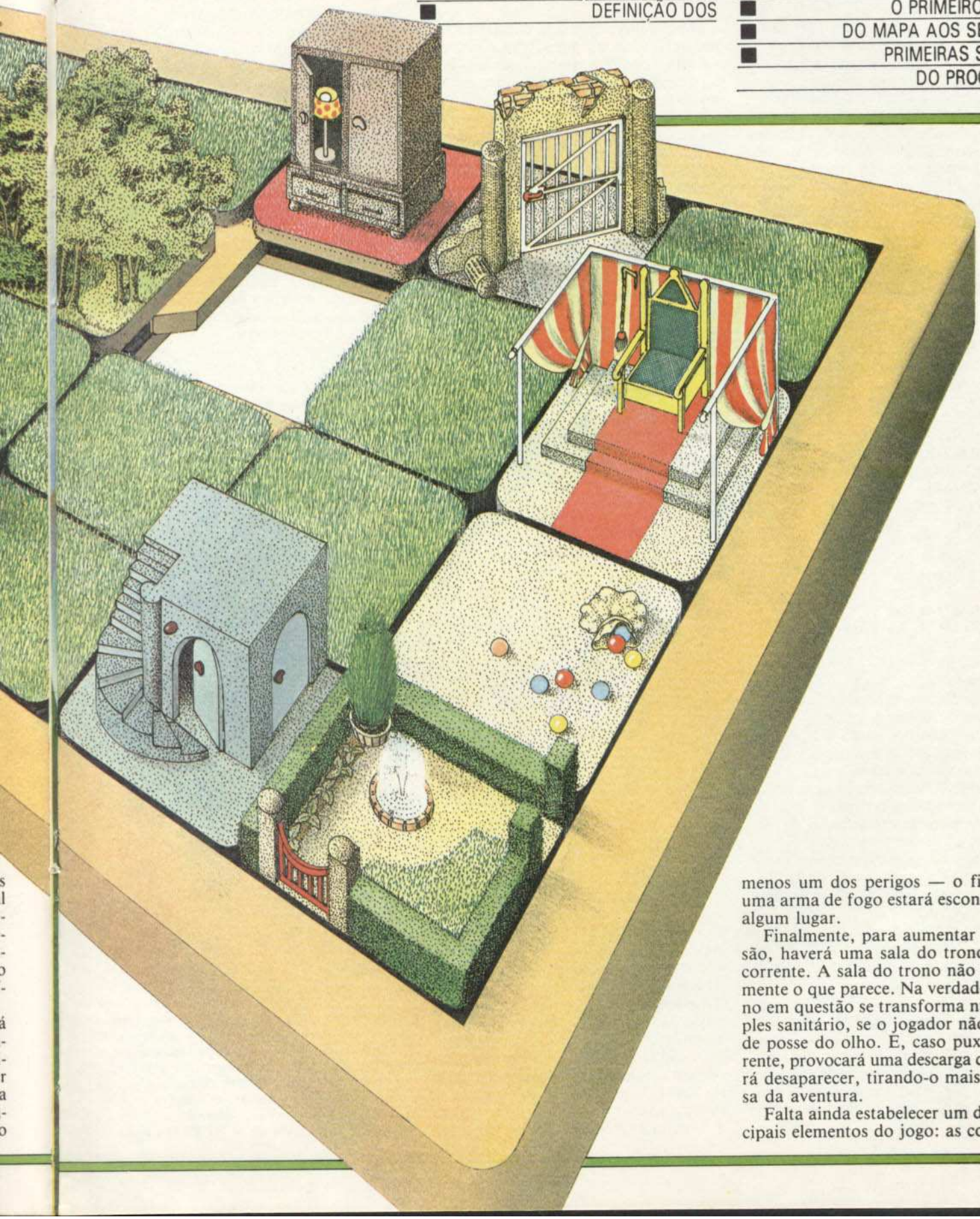
Nosso jogador encontra-se em péssima situação financeira e, por isso, sai em busca do fabuloso olho inca (de altíssimo valor), escondido em algum lugar do mundo da aventura. Infelizmente, a Secretaria da Receita enviou um fiscal para acompanhar o caso. O papel desse personagem é bastante semelhante ao do pirata de outras aventuras — ou seja, cabe-lhe arruinar o infeliz jogador. Sua aparição pode acarretar dois acontecimentos. Se o jogador estiver levando consigo um objeto, o fiscal o confiscará para amortizar sua enorme dívida com a Secretaria. Caso o jogador não tenha encontrado nenhum objeto (não podendo, portanto, pagar), o fiscal o prenderá numa masmorra.

Este é o esquema básico da aventura. Resta, agora, trabalhar alguns detalhes — por exemplo, os objetos que serão encontrados na busca. No nosso jogo, decidimos contrariar a regra geral de que todos os objetos devem ser úteis ao desenvolvimento da aventura. Desta vez, haverá um objeto sem nenhuma utilidade na busca empreendida pelo jogador: trata-se de algo pesado (um tijolo, por exemplo), que irá causar a morte de quem tentar atravessar o rio a nado, carregando-o.

O objeto mais importante de todos será o olho. Para aumentar o interesse do jogo, convém imaginar uma maneira de disfarçá-lo ou escondê-lo. Poderíamos colocá-lo dentro de um baú ou de uma catacumba, mas existem soluções mais sutis para enganar o jogador. Assim, em vez de esconder o olho num lugar que obviamente contém algo de valor, vamos deixá-lo dentro de um saquinho de bolas de gude. O aventureiro não irá a lugar nenhum se tentar jogar com as bolinhas de gude!

Um dos artifícios favoritos em jogos de aventura é o quarto escuro, no qual todas as coisas horríveis podem acontecer. Esta aventura não será uma exceção. Sem a lâmpada — que deverá descobrir em algum misterioso lugar — o pobre aventureiro não encontrará as saídas, ficando em péssima situação.

Talvez isso seja um tanto injusto, já que o jogador não receberá nenhum aviso de perigo iminente, nem terá condições de sair do quarto escuro, a não ser que tenha encontrado a lâmpada. Para dar uma melhor oportunidade ao aventureiro, permitindo-lhe enfrentar pelo



menos um dos perigos — o fiscal —, uma arma de fogo estará escondida em algum lugar.

Finalmente, para aumentar a diversão, haverá uma sala do trono e uma corrente. A sala do trono não é exatamente o que parece. Na verdade, o trono em questão se transforma num simples sanitário, se o jogador não estiver de posse do olho. E, caso puxe a corrente, provocará uma descarga que o fará desaparecer, tirando-o mais depressa da aventura.

Falta ainda estabelecer um dos principais elementos do jogo: as condições

necessárias à vitória. Não existe saída evidente do mundo da aventura e boa parte do quebra-cabeças consiste em descobrir como escapar com a peça do totem. Assim, para ganhar o jogo, o aventureiro obviamente precisará ter encontrado o olho — não apenas o saquinho de bolas de gude. Para dificultar um pouco mais, ele deverá também estar na sala do trono. Puxar a corrente desta vez não fará com que ele entre pelo cano!

Transformar a saída em risco, sob condições diferentes, apresenta a vantagem de desencorajar o jogador a tentá-la constantemente; desta maneira prolonga-se o jogo todo.

A AVENTURA ATÉ AQUI

Antes de começar o mapeamento, convém fazer uma recapitulação, listando os elementos até agora envolvidos na aventura. Assim, não se corre o risco de perder o fio da meada.

PERSONAGENS
- Aventureiro
- Fiscal da Receita — deve surgir aleatoriamente

OBJETOS
- Globo ocular — escondido no saquinho de bolas de gude.
- Tijolo — peso que mata o aventureiro caso este tente atravessar o rio a nado
- Lâmpada — necessária para se achar a saída do quarto escuro
- Arma — para matar o fiscal da Receita
- Corrente — tira o aventureiro do jogo se este chegar à sala do trono e puxá-la, sem estar de posse do olho

LOCAIS
- Rio
- Quarto escuro
- Sala do trono

Até aqui, fixamos apenas três dos locais da aventura, em função do que neles deve acontecer. Podemos muito bem, neste ponto, tomar novas decisões. Mas, seja como for, o próximo passo consistirá em encaixar todos os elementos num mesmo mapa do mundo da aventura.

COMO FAZER O MAPA

O primeiro mapa será composto por uma série de quadrados ligados entre si por setas, como mostra a ilustração desta página. Cada quadrado representará um ambiente ou local da aventura — local talvez seja o termo mais adequado, por sua abrangência. Pode designar desde a cabeça de um alfinete escondido na barra do vestido da rainha, até uma imensa planície. Todos os locais devem ser incluídos no mapa: os da lista preliminar e outros que se façam necessários para completar o jogo.

Ao esboçar o mapa, não se esqueça de assinalar a direção em que se pode ir a partir de cada local. Se quiser, inclua saídas que só funcionem em uma direção — acompanhadas de mensagens, como por exemplo:

A PORTA BATE,
FECHANDO-SE ATRÁS DE VOCÊ

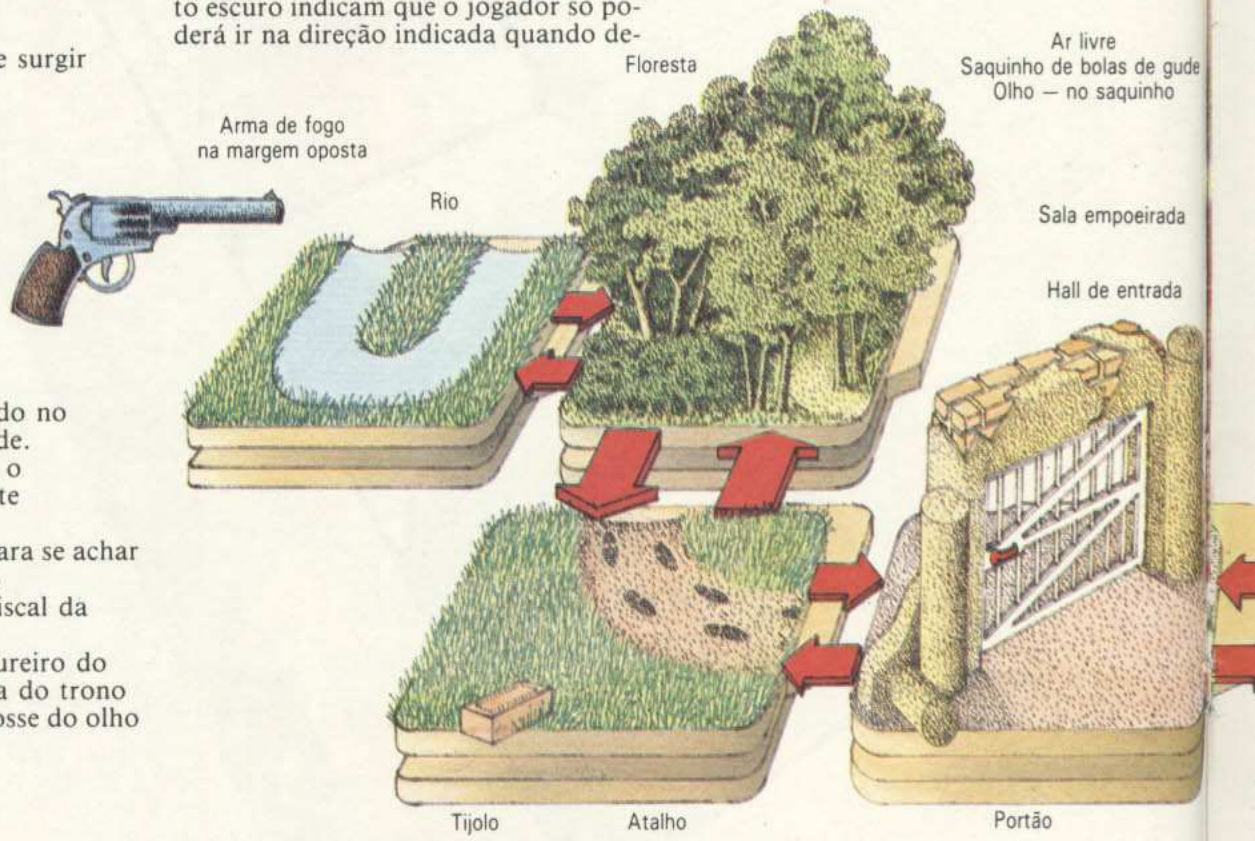
As faixas listradas que saem do quarto escuro indicam que o jogador só poderá ir na direção indicada quando determinadas condições forem satisfeitas. No nosso caso, a condição é estar de posse da lâmpada acesa, para enxergar as saídas.

Não é fácil prever o número de locais que determinada quantidade de memória RAM pode conter. A dificuldade está no próprio programa de aventura, que inclui muitos elementos, todos precisando de memória — descrições de locais, palavras que o programa deve reconhecer, o número de objetos e o papel de cada um deles, o número de enigmas e sua complexidade, etc.

Depois de ter programado algumas pequenas aventuras e verificado o espaço que ocupam, ficará mais fácil calcular o que é possível encaixar na memória do seu computador.

Quem tiver um micro de 16K logo descobrirá que é impossível escrever uma aventura em grande escala dispondo de tão pouca memória RAM. No entanto, a aventura que desenvolveremos aqui inclui um número reduzido de locais — doze, ao todo — e, portanto, não dará esse tipo de dor de cabeça.

Um mapa para a Busca do Olho Inca pode ser semelhante ao de nossa ilustração. As ligações entre os locais foram todas previstas, e o ponto de partida estabelecido. Isso é muito importante, pois afeta a maneira de se abordar a aventura, a ordem em que os objetos são descobertos e também a tentativa de



O primeiro mapa da aventura mostra todos os locais planejados em suas posições relativas. As setas vermelhas indicam as saídas que estão constantemente abertas; as setas listradas indicam saídas que só podem ser utilizadas sob condições especiais — neste caso, quando o aventureiro estiver de posse da lâmpada.

se resolver o quebra-cabeças.

Os objetos também estão assinalados em seus locais. Os que aparecerão posteriormente — como, por exemplo, o olho inca — devem ser listados à margem do mapa.

DO MAPA AOS SETORES

Uma vez concluído o mapa, os dados podem ser transferidos para os setores.

Em geral, no planejamento de jogos de aventura, os setores são organizados em dois tipos de conjunto: um que tem por base quadrados e outro que tem por base octógonos (veja as ilustrações na página 230). A escolha de qual utilizar dependerá do número de saídas de cada local.

O tipo mais simples de aventura inclui saídas em quatro direções: norte, sul, leste e oeste (como na Busca do Olho). Nesse caso, deve-se transferir os dados para um conjunto de setores quadrados, de modo que se enquadrem nas condições do mapa proposto. A maneira de fazê-lo será detalhadamente explicada mais adiante.

Se a aventura incluir saídas a noroeste, nordeste, sudeste e sudoeste, deve-se utilizar o conjunto de setores octogonais. O emprego desse tipo de conjunto é, porém, muito complicado.

Existe ainda a possibilidade de se incluir, na aventura, deslocamentos para cima ou para baixo. Nesse caso, a melhor solução é a utilização de conjuntos distintos de setores para cada "nível" da aventura.

A Busca do Olho tem como base um conjunto do tipo quadrado — ou seja, permite saídas somente para norte, sul, leste e oeste. A não ser que haja uma real necessidade de outras direções, esse tipo de aventura é bastante satisfatório, permitindo, inclusive, que se introduza uma certa confusão no direcionamento das saídas. Por exemplo, pode-se acrescentar a seguinte frase a uma descrição:

EXISTE UMA ESCADA
DESCENDO PARA O OESTE

e, para descer as escadas, bastará usar a resposta normal OESTE.

OS SETORES

Nossa aventura utilizará um conjunto de setores composto de seis por quatro quadrados — verifique o mapa proposto, para cima e para baixo, para a esquerda e para a direita. Antes de iniciar a transferência de todos os detalhes para os setores, certifique-se de que cada quadrado foi devidamente numerado. Deve-se começar com o número 1, no alto à esquerda, e prosseguir até a parte inferior à direita.

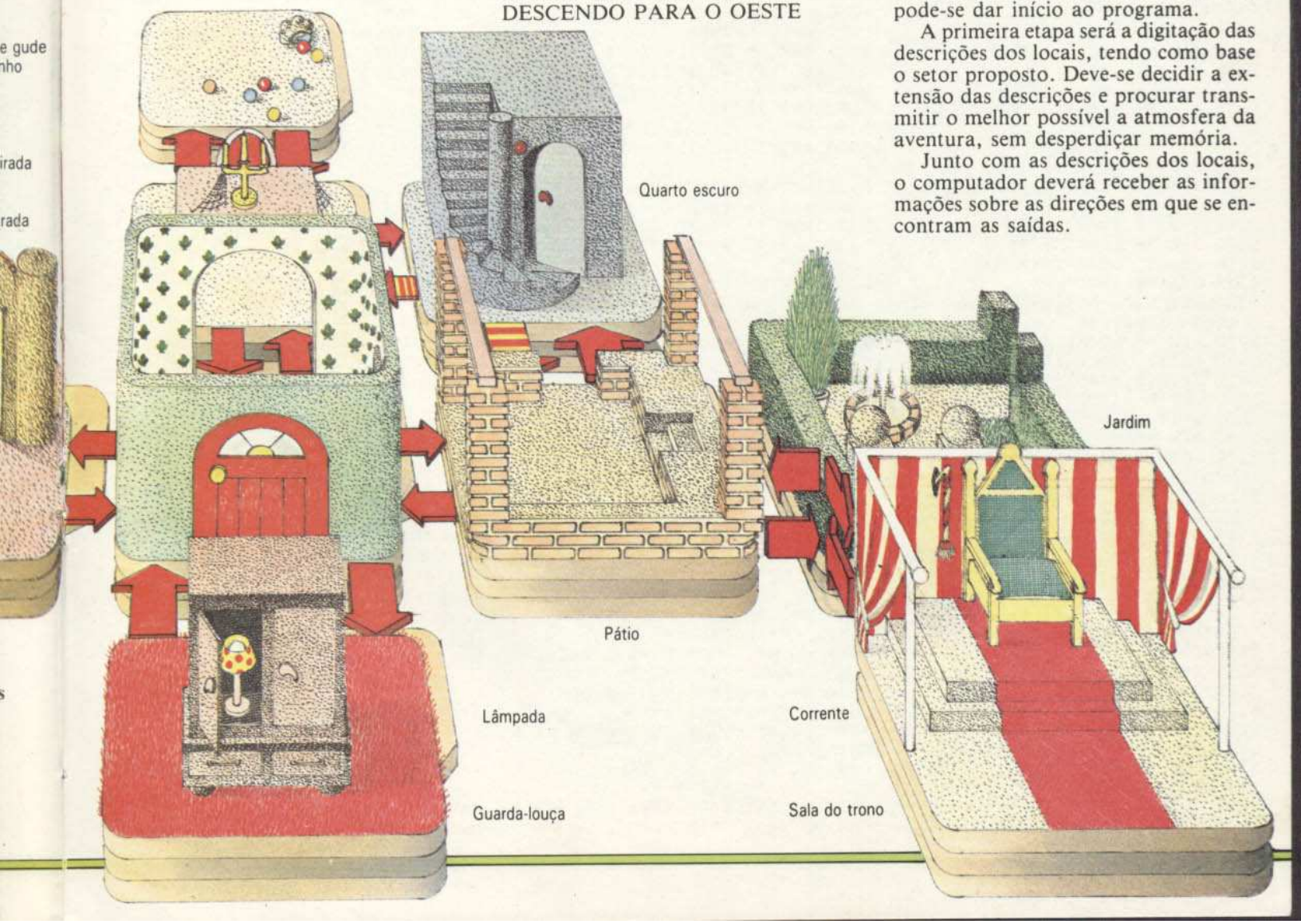
Uma vez numerados os quadrados e transferidos os detalhes, o conjunto dos setores terá uma aparência semelhante à da ilustração da página 231.

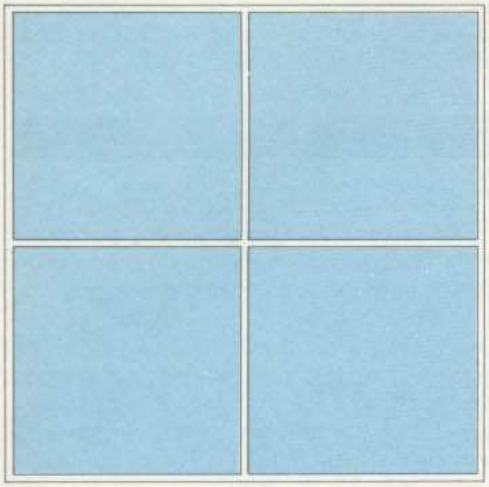
COMO COMEÇAR O PROGRAMA

Depois de estruturar o roteiro básico da história e de completar os setores, pode-se dar início ao programa.

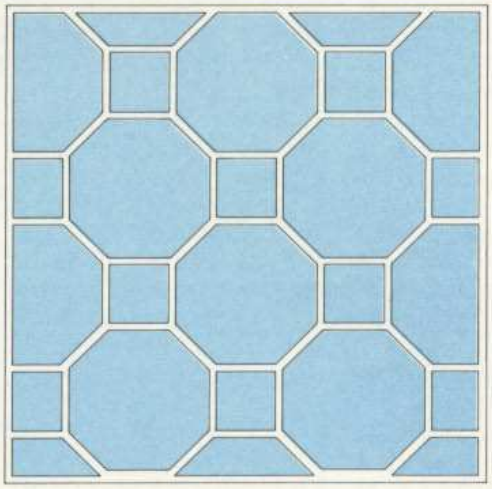
A primeira etapa será a digitação das descrições dos locais, tendo como base o setor proposto. Deve-se decidir a extensão das descrições e procurar transmitir o melhor possível a atmosfera da aventura, sem desperdiçar memória.

Junto com as descrições dos locais, o computador deverá receber as informações sobre as direções em que se encontram as saídas.





Parte de um conjunto de setores quadrados. Neste tipo de conjunto, pode-se planejar aventuras utilizando quatro saídas: para norte, sul, leste e oeste.



Parte de um conjunto de setores octogonais — utilizado quando se quer incluir saídas para noroeste, nordeste, sudoeste e sudeste.

Veja, abaixo, as primeiras seções do programa. O número alto das linhas tem a finalidade de assegurar espaço para as seções anteriores, à medida que o jogo for se desenvolvendo. Digite a seção e preserve-a em fita:

```
5000 REM ** DESCRICAO DO LOCAL
**
5010 REM  ** LOCAL 4 **
5020 PRINT "VOCE ESTA DO LADO D
E FORA DE UM GRANDE PREDIO"
5030 LET N=0: LET E=0: LET S=1:
 LET W=0: RETURN
5040 REM ** LOCAL 7 **
5050 PRINT "VOCE ESTA A BEIRA D
E UM GRANDE  RIO"
5060 LET N=0: LET E=1: LET S=0:
 LET W=0: RETURN
5070 REM ** LOCAL 8 **
5080 PRINT "VOCE ESTA NUMA FLOR
ESTA PETRIFI-CADA"
5090 LET N=0: LET E=0: LET S=1:
 LET W=1: RETURN
5100 REM ** LOCAL 10 **
5110 PRINT "VOCE ESTA NUMA SALA
 EMPOEIRADA"
5120 LET N=1: LET E=1: LET S=1:
 LET W=0: RETURN
5130 REM ** LOCAL 11 **
5140 PRINT "VOCE ESTA NUMA SALA
 ESCURA"
5150 IF LA<>1 THEN  LET N=0: LE
T E=0: LET S=0: LET W=0: PRINT
" ESTA MUITO ESCURO PARA VER AS
   SAIDAS": LET DA=1: RETURN
5160 LET N=0: LET E=0: LET S=1:
 LET W=1: RETURN
5170 REM ** LOCAL 14 **
5180 PRINT "VOCE ESTA EM UM ATA
LHO ENLAMEADO"
5190 LET N=1: LET E=1: LET S=0:
 LET W=0: RETURN
5200 REM ** LOCAL 15 **
5210 PRINT "VOCE ESTA NA ENTRAD
A DA CIDADE OCULTA"
5220 LET N=0: LET E=1: LET S=0:
 LET W=1: RETURN
5230 REM ** LOCAL 16 **
5240 PRINT "VOCE ESTA NO HALL D
E ENTRADA"
5250 LET N=1: LET E=1: LET S=1:
 LET W=1: RETURN
5260 REM ** LOCAL 17 **
5270 PRINT "VOCE ESTA NO PATIO"
5280 LET N=1: LET E=1: LET S=0:
 LET W=1: RETURN
5290 REM ** LOCAL 18 **
5300 PRINT "VOCE ESTA NO JARDIM
"
5310 LET N=0: LET E=0: LET S=1:
 LET W=1: RETURN
5320 REM ** LOCAL 22 **
5330 PRINT "VOCE ESTA NO GUARDA
-LOUCAS"
5340 LET N=1: LET E=0: LET S=0:
 LET W=0: RETURN
5350 REM ** LOCAL 24 **
5360 PRINT "VOCE ESTA NA SALA D
O TRONO"
5370 LET N=1: LET E=0: LET S=0:
 LET W=0: RETURN
```

```
5000 REM**DESCRICAO DOS LOCAIS*
*
5010 REM**LOCAL 4**
5020 PRINT" VOCE ESTA DO LADO D
E FORA DE    UMA GRANDE CONSTRU
CAO"
5030 N=0:E=0:S=1:W=0:RETURN
5040 REM**LOCAL 7**
5050 PRINT" VOCE ESTA A BEIRA D
E UM GRANDE  RIO"
5060 N=0:E=1:S=0:W=0:RETURN
5070 REM**LOCAL 8**
5080 PRINT" VOCE ESTA NUMA FLOR
ESTA PETRIFI CADA"
5090 N=0:E=0:S=1:W=1:RETURN
5100 REM**LOCAL 10**
5110 PRINT" VOCE ESTA NUMA SALA
 MUITO SUJA"
5120 N=1:E=1:S=1:W=0:RETURN
5130 REM**LOCAL 11**
5140 PRINT" VOCE ESTA NUM QUART
O ESCURO"
5150 IF OB(6)<>-1 OR LA<>1 THEN
 N=0:E=0:S=0:W=0:PRINT" ESTA MU
ITO ESCURO PARA VER AS   SAIDAS
":RETURN
5160 N=0:E=0:S=1:W=1:RETURN
5170 REM**LOCAL 14**
5180 PRINT" VOCE ESTA NUM ATALH
O ENLAMEADO"
5190 N=1:E=1:S=0:W=0:RETURN
5200 REM **LOCAL 15**
5210 PRINT" VOCE ESTA NA ENTRAD
A DA CIDADE OCULTA"
5220 N=0:E=1:S=0:W=1:RETURN
5230 REM**LOCAL 16**
5240 PRINT" VOCE ESTA NO HALL D
E ENTRADA"
5250 N=1:E=1:S=1:W=1:RETURN
5260 REM**LOCAL 17**
5270 PRINT" VOCE ESTA NO PATIO"
5280 N=1:E=1:S=0:W=1:RETURN
5290 REM**LOCAL 18**
5300 PRINT" VOCE ESTA NO JARDIM
"
5310 N=0:E=0:S=1:W=1:RETURN
5320 REM**LOCAL 22*
5330 PRINT" VOCE ESTA NO GUARDA
-LOUCAS"
5340 N=1:E=0:S=0:W=0:RETURN
5350 REM**LOCAL 24**
5360 PRINT" VOCE ESTA NA SALA D
O TRONO"
5370 N=1:E=0:S=0:W=0:RETURN
```

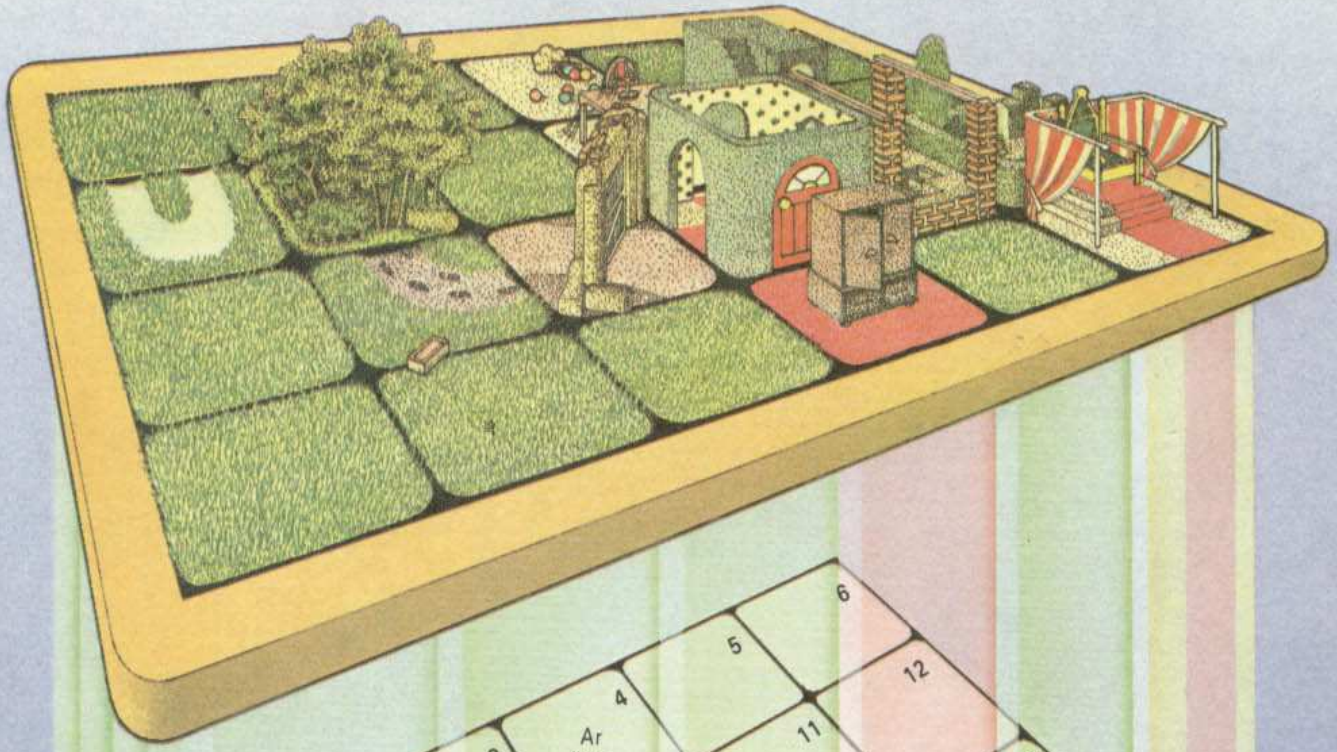
Não se preocupe com o uso repetido de comentários REM incluídos na memória. Nesta etapa inicial do desenvolvimento do programa, o importante é saber o que cada trecho do programa faz, ou qual o número do local a que determinada descrição se refere. Os comentários sempre podem ser eliminados posteriormente. Após cada linha de descrição do local, existe outra linha contendo dados sobre suas possíveis saídas. As variáveis N, S, L e O referem-se a norte, sul, leste e oeste. Elas podem ter um ou dois valores — 0 significa que não há saída naquela direção, ao passo que 1 significa que há uma saída.

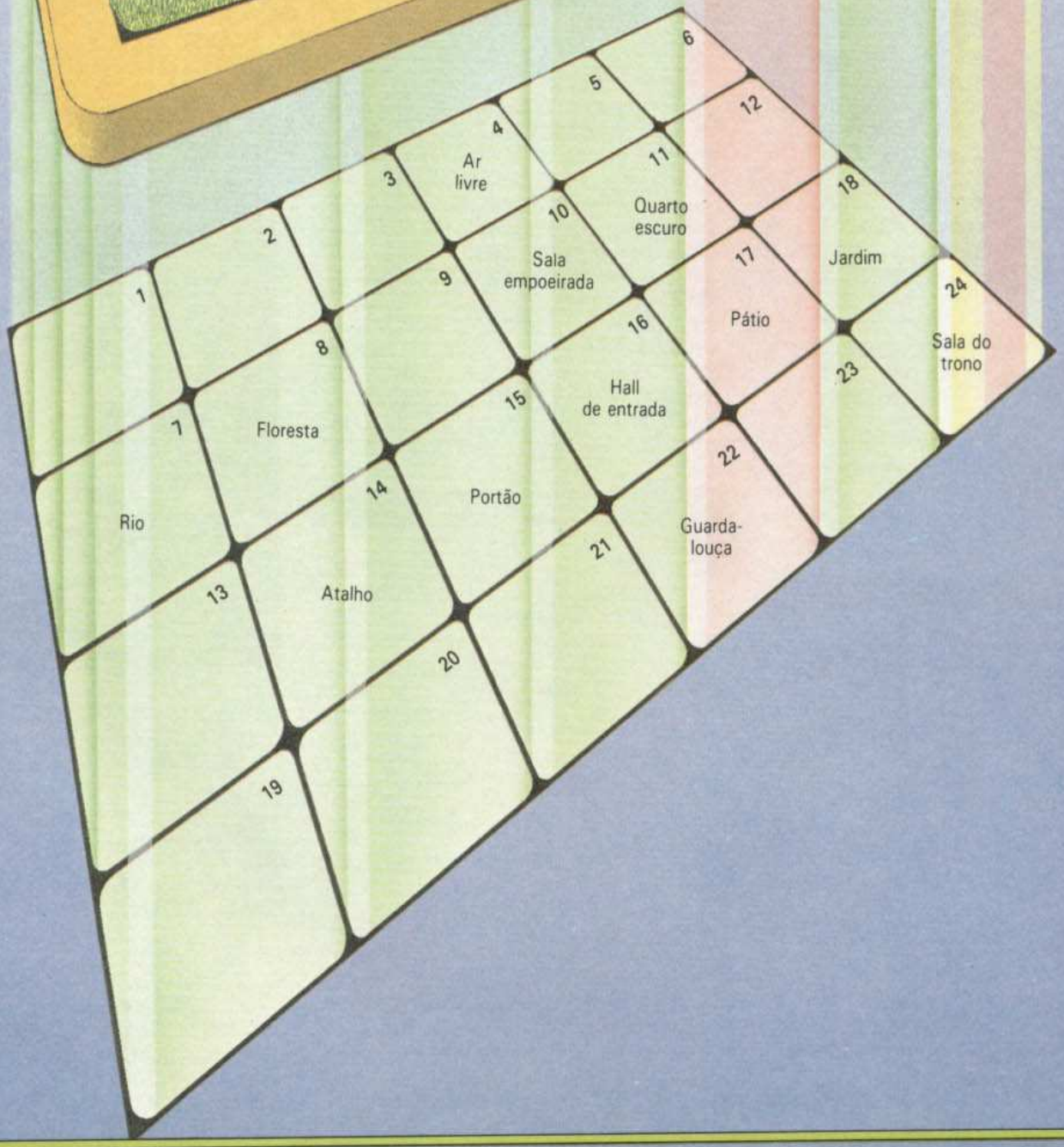
Finalmente, existe um **RETURN** após as seções do programa, pois cada descrição de local será chamada por uma instrução **GOSUB**.

Um **IF...THEN** extra na seção do quarto escuro verifica se o aventureiro possui a lâmpada; porém, a descrição das variáveis será tratada mais adiante, quando analisarmos os objetos.

No próximo artigo veremos como movimentar o jogador pelo mundo que criamos.

Última etapa antes da programação. O conjunto de setores é uma cópia direta do mapa, numa forma que facilita o trabalho.




1
2
3
4
Ar livre
5
6
7
Rio
8
Floresta
9
10
Sala empoeirada
11
Quarto escuro
12
13
14
Atalho
15
Portão
16
Hall de entrada
17
Pátio
18
Jardim
19
20
21
22
Guarda-louça
23
24
Sala do trono

RECURSOS GRÁFICOS SOFISTICADOS

Os computadores domésticos oferecem vários recursos ao artista amador. Aprenda novas formas de utilizar os comandos gráficos em BASIC e crie livremente suas ilustrações.

Tendo já dominado os princípios básicos do desenho na tela, você pode começar a empregar alguns recursos gráficos especiais, disponíveis no seu computador. Comandos como **MOVE**, **PLOT**, **DRAW**, **PAINT** e **CIRCLE** permitem que você solte as rédeas de sua imaginação, criando qualquer tipo de imagem — de um diagrama explicativo para ilustrar uma mensagem comercial ao cenário de um excitante jogo de aventuras.

No artigo da página 113 mostramos como utilizar comandos simples para criar ilustrações com linhas na tela e, em alguns casos, como acrescentar cor. Mas você pode sofisticar seu repertório, aprendendo outras maneiras de traçar pontos, desenhar linhas, triângulos, quadrados e círculos. Os comandos aqui ensinados — em combinação com cores ou não — constituem um eficiente recurso para formar imagens estáticas ou até mesmo móveis.

Embora os vários modelos de computador utilizem esses comandos de maneira diferente, todos são capazes de produzir alguns efeitos interessantes. As exceções são o Apple, o TK-2000, o TRS-80 e o ZX-81, cujo BASIC não inclui esses comandos. Entretanto, você pode obtê-los acrescentando um cartucho **ROM** adequado ou adquirindo programas em disco ou fita.

ARCOS E CÍRCULOS

Os arcos e círculos são um dos mais úteis "instrumentos" do Spectrum para fazer desenhos na tela.

Este programa para um campo de golfe mostra como empregá-los para desenhar árvores, cercas, água, terrenos acidentados e buracos.

Você pode testar seu progresso na medida em que for rodando cada grupo de linhas. Não dê o comando **NEW** toda hora; se você deixar as linhas intactas, obterá a cena mostrada na ilustração da página 237.

Antes de começar o trabalho com os círculos, porém, convém deixar pronta a sede do clube. Para isso, entre estas linhas no computador:

```
 90 BORDER 4: PAPER 4: CLS
200 LET w=10: LET s=50
210 FOR c=162 TO 174
220 PLOT  INK 2;w,c
230 DRAW  INK 2;s,0
240 LET w=w+2: LET s=s-4
250 NEXT c
260 FOR b=148 TO 162
270 PLOT  INK 2;10,b
280 DRAW  INK 2;50,0
290 NEXT b
295 DRAW  INK 2;10,-3: DRAW 0,
-11
300 PRINT  INK 0;AT 2,2;" ";AT
2,4;" ";AT 2,6;" "
```

As linhas numeradas de 200 a 250 desenham o telhado, utilizando técnicas semelhantes às ensinadas no artigo anterior (página 113). Na tela, em **10.162**, aparece primeiro uma linha de 50 pixels de largura. Depois uma largura de 4 pixels, para cada pixel desenhado.

Um laço parecido aos das linhas 260 a 290 desenha as paredes, com duas linhas extras (na linha 295) para traçar a varanda. Em seguida, a linha 300 preenche as janelas pelo método mais simples possível — imprimindo um quadrado negro, com caracteres gráficos da **ROM**, em três lugares da parede.

COMO DESENHAR ARCOS

No Spectrum, como já foi explicado anteriormente (página 115), o comando **CIRCLE** é o instrumento mais simples para se desenhar um círculo completo.

Mas, se quiser somente uma parte do círculo, será mais fácil acrescentar um número a uma declaração convencional **DRAW**. Pode-se obter uma variedade imensa de efeitos por meio dessa declaração; assim, vale a pena fazer alguns experimentos antes de prosseguir.

```
10 PLOT 130,30
20 DRAW 0,10,1
30 GOTO 20
```

(Não se preocupe com a mensagem de erro.)

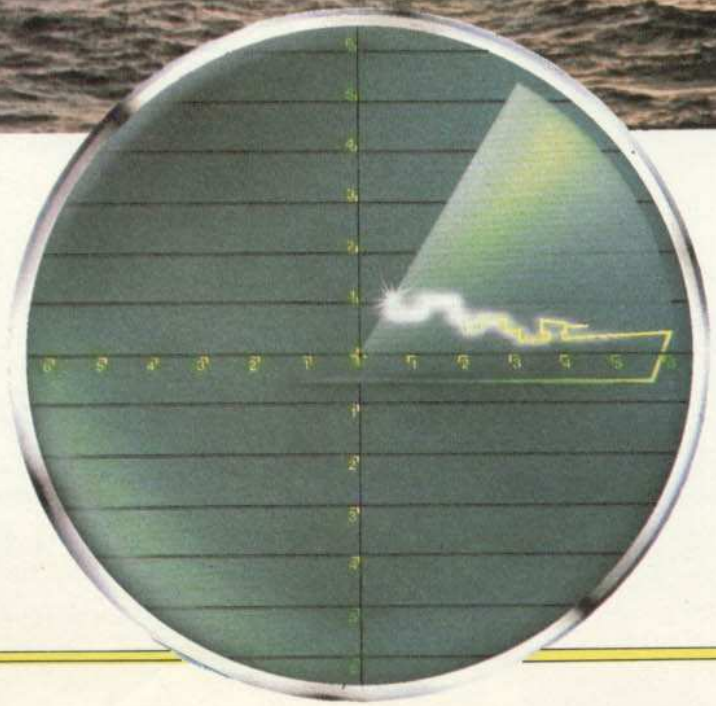
Como você deve se lembrar, os primeiros dois números da linha 20 dizem ao computador para desenhar uma linha do ponto traçado até um ponto 10 pixels acima dele. O último número, por sua vez, determina a curvatura da linha.

Informa-se assim, ao computador, qual fração do círculo será desenhada. Um círculo completo é representado por duas vezes Pi (3.1416); portanto, com o número 1, obtém-se o desenho de aproximadamente um sexto do círculo (ou, mais precisamente, 1 dividido por duas vezes Pi). Se você tentar modificar a linha 20 para:

```
20 DRAW 0,10,2
```

... obterá uma série de curvas mais pronunciadas. Do mesmo modo, 0,10,3 fornece um modelo denteado, e 0,10,4 produz parte de uma cerca (ou a metade de um tronco de palmeira, dependendo de

- RECURSOS GRÁFICOS ESPECIAIS
- ARCOS E CÍRCULOS
- DESENHE UM CAMPO DE GOLFE NO SPECTRUM
- O COMANDO **DRAW**
- UM NAVIO EM POUCAS LINHAS
- COMO DESENHAR LETRAS
- MENSAGENS LONGAS



sua imaginação), enquanto 0,10,6 produz uma espiral. Se, no entanto, você tentar:

```
20 DRAW 0,10,2*PI
```

... poderá ter uma surpresa! O Spectrum desenhará um círculo em que dois pontos estarão em linha reta. Mas, já que tal círculo pode se tornar muito maior do que sua sala — na verdade, bem maior do que o sistema solar —, o computador desenhará apenas a parte do círculo que é capaz.

Para o cenário, experimente isto:

```
10 PLOT 0,100
20 DRAW RND*5+5,0,2
30 GOTO 20
```

A forma obtida assemelha-se à das ondas em um mar agitado. Para desenhar águas calmas, tente **RND*10 + 10** ou mesmo **RND*15 + 10**.

Com relação aos círculos e arcos, vale ainda lembrar que um número negativo no final da linha **DRAW** produzirá a imagem-espelho do arco.

Agora, limpe as linhas de 10 a 30 e continue montando o cenário para a pista de golfe.

A CERCA E O LAGO

O programa da pista de golfe inclui dois exemplos de arcos utilizados para efeito gráfico — o primeiro compõe a pequena cerca em frente à sede do clube; o segundo, o lago.

Digite estas linhas:

```
310 FOR f=0 TO 84 STEP 3
320 PLOT f,142
330 DRAW 3,0,-3
340 NEXT f
350 DRAW 35,-42: DRAW -12,-6
```

O ponto de partida na tela é **0,142**. A linha 330 desenha uma série de arcos estreitos — ou seja, semicírculos com apenas 3 pixels de largura — para formar a cerca. O número obtido é regulado pelo laço **FOR...NEXT**.

Agora, entre e rode estas linhas:

```
100 LET x=130: LET y=125: LET
z=50
110 PLOT  INK 5;x,0
120 DRAW  INK 5;y,z,-1.25
130 LET x=x+1: LET y=y-1: LET
z=z-1
```

```
140 IF x>254 THEN  GOTO 170
150 IF z<1 THEN  LET z=0
160 GOTO 110
```

Aqui a ilustração é mais complicada. Em primeiro lugar, o computador traça um ponto de 130 pixels a partir da esquerda e 0 pixel a partir do canto inferior da tela. Em seguida, desenha uma linha no ponto 125 pixels à direita e a eleva a 50 pixels do canto inferior da tela, "arqueando" a linha em −1.25 radiano, à medida que ela vai se desenvolvendo.

A partir desse estágio as variáveis entram em cena. A variável **x** inicia cada linha 1 pixel mais à direita; a variável **y** torna a linha 1 pixel mais curta do que a anterior (caso contrário ela ultrapassaria os limites da tela), enquanto a variável **z** faz com que a linha termine 1 pixel abaixo da anterior.

Eventualmente, é claro, a variável **z** poderá se tornar um número negativo, fazendo com que o computador tente (mas falhe) imprimir o canto inferior da tela. Por isso a linha 150 é necessária: ela faz com que todas as pequenas linhas no final do programa terminem na linha inferior da tela.

AS ÁRVORES E OS ARBUSTOS

O programa da pista de golfe também utiliza círculos completos (como um substituto para **PAINT** ou uma declaração similar, disponível em alguns computadores) para produzir árvores e arbustos. Estas poucas linhas desenharão alguns arbustos ao acaso no "gramado":

```
400 FOR r=172 TO 168 STEP -1
410 LET x=RND*45+195
415 PLOT x,r-2: DRAW 0,-2
420 CIRCLE x,r,RND*2+1
440 CIRCLE x+10,r,RND*2+1
450 NEXT r
```

Estas outras linhas produzirão algo parecido, no lado direito:

```
460 FOR r=135 TO 172 STEP 6
470 LET y=252
480 CIRCLE y,r,RND+2
490 NEXT r
```

As árvores, no lado inferior esquerdo, são muito grandes e, por isso, não se pode desenhá-las aleatoriamente, usando **DRAW**. Assim, utilizamos uma técnica diferente: **READ...DATA** (veja páginas 128 a 133):

```
900 FOR w=1 TO 3
910 READ a,b
920 PLOT a,b
930 DRAW 0,-24
940 LET f=RND*5+5
950 CIRCLE a-10,b+f,f: CIRCLE
a,b+f,f: CIRCLE a+10,b+f,f
960 CIRCLE a-5,b+f*2,f:
CIRCLE a+5,b+f*2,f
970 CIRCLE a,b+f*3,f
980 NEXT w
3000 DATA 20,70,52,85,84,100
```

A solução, aqui, foi desenhar primeiro os troncos, de modo descendente, a partir dos pontos traçados originalmente. Desse modo, evita-se que eles apareçam através das "folhagens". Os troncos são traçados na tela a **20,70** e assim por diante, pelos dados fornecidos pela linha 3000. A linha 940 desenha aleatoriamente o tamanho das folhagens, enquanto que **b + f**, na linha 950, assegura que as séries de círculos ao fundo iniciem com uma distância razoável acima dos troncos.

TOQUES FINAIS

O **tee** propulsor, ou seja, o apoio onde se coloca a bola para a tacada inicial, é desenhado por estas linhas:

```
1000 LET t=30
1010 FOR y=0 TO 10
1020 PLOT t,y
1030 DRAW -30,30
1040 LET t=t+2
1050 NEXT y
```

E as linhas abaixo desenharão as bandeirinhas no gramado:

```
170 PLOT 220,140
180 DRAW 0,15: DRAW 8,-3
DRAW -8,-2
190 PLOT 22,120
195 DRAW 0,18: DRAW 9,-3:
DRAW -9,-2
```

Finalmente, você precisará de alguns buracos. Na ausência de uma declaração **PAINT**, a maneira de desenhá-los que oferece os melhores resultados (embora também seja a mais demorada) é começar com uma elipse bem pequena e ir aumentando-a, pixel por pixel, até atingir um tamanho razoável.

Como o traçado de elipses será detalhadamente explicado em um artigo futuro sobre as funções matemáticas do computador, entre e rode estas linhas:

```
1495 LET r=1
1500 FOR x=0 TO 2*PI STEP PI/18
0
1510 PLOT  INK 6;168+r*SIN x,14
7+r*COS x/2.5
1520 PLOT  INK 6;235+r*SIN x,10
6+r*COS x/2.75
1530 PLOT  INK 6;225+r*SIN x,97
+r*COS x/2.5
1540 NEXT x
1550 LET r=r+2
1560 IF r>20 THEN  GOTO 6000
1570 GOTO 1500
```

Alternativamente, você poderá fazer os arbustos de um modo mais simples, porém mais grosseiro, utilizando blocos gráficos.

O COMANDO DRAW

Os comandos **CIRCLE** e **LINE** são muito úteis para desenhos simples de formato regular, requerendo um mínimo esforço do programador. Empregá-los para desenhos mais complicados, porém, tornará seus programas longos e de difícil manuseio. Imagine o tamanho de um programa que desenhasse algo como o navio da página 232, usando um comando **LINE** para cada linha da ilustração...

Para contornar esse problema, devemos utilizar o comando **DRAW**, que permite direcionar a trajetória de uma linha enquanto ela está sendo traçada. Podemos dizer ao computador que leve a linha até uma certa distância à direita, até uma outra distância para cima e assim por diante. Essas instruções são fornecidas dentro de um cordão, o que torna o programa muito mais compacto.

Como exemplo, digite e rode este programa; veja como é possível desenhar um navio com algumas poucas linhas.

```
10 PMODE 4,1
20 PCLS 5
30 SCREEN 1,1
40 DRAW "BM23,96C0"
50 DRAW "R28E2U3L6UR6E2R5F2D3R3
U2R7D2R4U9E2R4F2D5R3U2R4U6E3R5D
8"
60 DRAW "R3U24L3UR8DL4D24RF2R5D
4R4U4R6U9E3R5D10R4U2R4U14RD10R3
U2"
70 DRAW "R4D11R7U3E2R4F2D3RD8R3
U5E2R8D2R6U2R7UE2R6F2D4R4U4E2R5
F2"
80 DRAW "R6DL6D3R24G12L195H4U5"
90 PAINT (127,100),0,0
100 GOTO 100
```

```
30 SCREEN 2
40 DRAW "BM23,96C1"
50 DRAW "R28E2U3L6UR6E2R5F2D3R3
U2R7D2R4U9E2R4F2D5R3U2R4U6E3R5D
8"
60 DRAW "R3U24L3UR8DL4D24RF2R5D
4R4U4R6U9E3R5D10R4U2R4U14RD10R3
U2"
70 DRAW "R4D11R7U3E2R4F2D3RD8R3
U5E2R8D2R6U2R7UE2R6F2D4R4U4E2R5
F2"
80 DRAW "R6DL6D3R24G12L195H4U5"
```

```
90 PAINT (127,100),1,1
100 GOTO 100
```

O programa funciona assim:

As linhas numeradas de 10 a 30 estabelecem as condições iniciais, selecionando uma tela de alta resolução.

Os comandos **DRAW** das linhas 40 a 80 formam o contorno do navio. Este comando opera somente no cordão de instruções contidas entre aspas.

A linha 40 é a mais simples, com um curto cordão de instruções que dizem à máquina onde começar o desempenho, e com que cor. A primeira instrução, **BM**, significa "mover em branco", colocando a "caneta" que fará o desenho na posição onde desejamos iniciá-lo. Se não houvesse a letra **B**, a "caneta" traçaria uma reta, enquanto se dirigisse ao ponto de partida, que no nosso caso é **23,96** (note que as coordenadas não estão entre parênteses, como em outros comandos gráficos). A instrução final do cordão seleciona a cor preta (**Ci** no MSX e **CO** no TRS-Color).

A linha 50 começa a traçar o contorno. Embora pareça confuso, o cordão da instrução **DRAW** é na realidade muito simples, consistindo de uma série de direções e distâncias. As letras controlam a direção da linha e os números, sua distância em pixels. Se não houver um número após uma direção, o movimento será de apenas um pixel.

Existem 8 direções: **U**, para cima (*up*, em inglês); **R**, direita (*right*); **D**, para baixo (*down*); **L**, esquerda (*left*); **E**, para cima à direita (45 graus); **F**, para baixo à direita (135 graus); **G**, para baixo à esquerda (225 graus) e **H**, para cima à esquerda (315 graus).

Se acompanharmos os comandos do cordão da linha 50, teremos: à direita 28 pixels, a 45 graus 2 pixels, para cima 3 pixels, e assim por diante. Para se exercitar, tente transformar o cordão em movimentos de lápis, em papel milimetrado.

As linhas 60 a 80 contêm cordões parecidos que completam o contorno do navio. Pode-se utilizar apenas um cordão longo em vez de todas essas linhas, mas um número grande de instruções tende a tornar mais difícil a manipulação e a correção dos dados.

Finalizando o desenho, a linha 90 pinta de preto a silhueta traçada na tela.

COMO DESENHAR LETRAS

Uma das limitações do TRS-Color em relação aos programas gráficos é sua incapacidade de exibir textos na tela de alta resolução (isto não vale para o MSX). Não se pode, por exemplo, imprimir uma contagem de pontos em um jogo que utiliza gráficos (por essa razão, o jogo da página 62 é todo exibido em tela de textos).

Para contornar o problema, você deve projetar seus próprios caracteres utilizando o comando **DRAW**. Digite e rode este programa e você verá a interjeição **ALÔ!** ser traçada:

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 OI$="D4R3U4L3BR5D4R3BR2U4R3D
2NL3D2BR2U1BU1U2"
50 DRAW "BM110,50;C3S8"+OI$
60 GOTO 60
```

```
30 SCREEN 2
40 OI$="D4R3U4L3BR5D4R3BR2U4R3D
2NL3D2BR2U1BU1U2"
50 DRAW "BM110,50;C3S8"+OI$
60 GOTO 60
```

No TRS, a mensagem é traçada em **PMODE3**, uma modalidade de quatro cores. No MSX, em **SCREEN 2**, a tela de alta resolução (lembre-se de que o MSX tem outras maneiras de escrever textos nesta tela).

A linha 40 mostra mais uma característica do comando **DRAW**. Como ele opera em um cordão de instruções, você poderá definir variáveis e chamá-las mais tarde com um comando **DRAW** no programa.

As instruções em **OI$** dizem ao computador como traçar as letras para formar **ALÔ!**. Tente transformá-las em movimentos de lápis, como fez anteriormente, lembrando-se de que **B** significa espaço em branco; assim, nenhuma linha aparecerá no desenho quando uma instrução for precedida de **B**.

Se você quiser exibir em telas de gráficos outras palavras ou expressões — **QUE AZAR!** ou **BEM FEITO!**, por exemplo — poderá estabelecer mais cordões, tais como **QA$** e **BF$**. Tente elaborar as instruções para uma dessas expressões em papel gráfico. Uma vez que tenha definido os cordões, você poderá chamá-los de volta sempre que forem necessários ao programa.

O **ALÔ!** é traçado pela linha 50. A posição inicial é **110,50** e a cor é de número 3; a palavra será impressa em tamanho 8 (escala dupla). O **S** pode ser seguido por um número de 1 a 62, no TRS, e de 1 a 255, no MSX. O 1 é um quarto do tamanho, o 4 é o tamanho normal (a escala que se obtém se não houver instrução) e o 8 é o tamanho duplo.

Como demonstra a linha 50, é possível unir cordões de instruções **DRAW** como quaisquer outros cordões. Se você quiser chamar cordões como **QA$** ou **BF$**, poderá fazê-lo do mesmo modo que na linha 50. Movimente para o ponto de partida utilizando **BM** e, então, estabeleça a cor por **C** e o tamanho por **S**.

MENSAGENS MAIS LONGAS

É muito trabalhoso definir cordões para cada mensagem quando se quer exibir na tela um grande número delas.

Para simplificar a tarefa, defina todos os caracteres que irá utilizar, digitando um programa como o que se segue:

```
10 PMODE 3,1
20 DIM LE$(26)
30 PCLS
40 FOR K=0 TO 26
45 READ LE$(K):NEXT
50 FOR K=0 TO 9
55 READ NU$(K):NEXT
60 DATA BR2,ND4R3D2NL3ND2BE2,ND
4R3DGNL2FDNL3BU4BR2,NR3D4R3BU4B
R2,ND4R2FD2GL2BE4BR,NR3D2NR2D2R
3BU4BR2
70 DATA NR3D2NR2D2BE4BR,NR3D4R3
U2LBE2BR,D4BR3U2NL3U2BR2,ND4BR2
,BD4REU3L2R3BR2,D2ND2NF2E2BR2
80 DATA D4R3BU4BR2,ND4FREND4BR2
,ND4F3DU4BR2,NR3D4R3U4BR2,ND4R3
D2NL3BE2,NR3D4R3NHU4BR2
90 DATA ND4R3D2L2F2BU4BR2,BD4R3
U2L3U2R3BR2,RND4RBR2,D4R2U4BR2,
D3FEU3BR2,D4EFU4BR2
100 DATA DF2DBL2UE2UBR2,DFND2EU
BR2,R3G3DR3BU4BR2
110 DATA NR2D4R2U4BR2,BDEND4BR2
,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R
2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2B
E4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D
4R2U2NL2U2BR2,NR2D2R2D2U4BR2
120 SCREEN 1,0
130 A$="TESTANDO0123456789"
140 DRAW "BM60,50C3S8"
150 GOSUB 9000
160 GOTO 160
9000 FOR K=1 TO LEN(A$)
9010 B$=MID$(A$,K,1)
9020 IF B$>="0" AND B$<="9" THE
N DRAW NU$(VAL(B$)):GOTO 9050
9030 IF B$=" " THEN N=0 ELSE N=
ASC(B$)-64
9040 DRAW LE$(N)
9050 NEXT
9060 RETURN
```

```
20 DIM LE$(26)
30 CLS
40 FOR K=0 TO 26
45 READ LE$(K):NEXT
50 FOR K=0 TC 9
55 READ NU$(K):NEXT
60 DATA BR2,ND4R3D2NL3ND2BE2,ND
4R3DGNL2FDNL3BU4BR2,NR3D4R3BU4B
R2,ND4R2FD2GL2BE4BR,NR3D2NR2D2R
3BU4BR2
70 DATA NR3D2NR2D2BE4BR,NR3D4R3
U2LBE2BR,D4BR3U2NL3U2BR2,ND4BR2
,BD4REU3L2R3BR2,D2ND2NF2E2BR2
80 DATA D4R3BU4BR2,ND4FREND4BR2
,ND4F3DU4BR2,NR3D4R3U4BR2,ND4R3
D2NL3BE2,NR3D4R3NHU4BR2
90 DATA ND4R3D2L2F2BU4BR2,BD4R3
U2L3U2R3BR2,RND4RBR2,D4R2U4BR2,
D3FEU3BR2,D4EFU4BR2
100 DATA DF2DBL2UE2UBR2,DFND2EU
BR2,R3G3DR3BU4BR2
110 DATA NR2D4R2U4BR2,BDEND4BR2
,R2D2L2D2R2BU4BR2,NR2BD2NR2BD2R
2U4BR2,D2R2D2U4BR2,NR2D2R2D2L2B
E4,D4R2U2L2BE2BR2,R2ND4BR2,NR2D
4R2U2NL2U2BR2,NR2D2R2D2U4BR2
120 SCREEN 2
130 A$="TESTANDO0123456789"
140 DRAW "BM60,50C1S8"
150 GOSUB 9000
160 GOTO 160
9000 FOR K=1 TO LEN(A$)
9010 B$=MID$(A$,K,1)
9020 IF B$>="0" AND B$<="9" THE
N DRAW NU$(VAL(B$)):GOTO 9050
9030 IF B$=" " THEN N=0 ELSE N=
ASC(B$)-64
9040 DRAW LE$(N)
9050 NEXT
9060 RETURN
```

O conjunto de caracteres — letras de A a Z e algarismos de 0 a 9 — está contido nas linhas **DATA** de 60 a 110. O conteúdo destas é transferido para duas variáveis alfanuméricas — **LE$** e **NU$** — pelas linhas 40 a 55.

Para utilizar esse conjunto é necessário definir a mensagem a ser escrita, assim como a posição inicial, a cor e o tamanho. A linha 130 coloca em **A$** uma mensagem de teste — **TESTANDO0123 456789** — que pode ser substituída por qualquer outra de sua criação. A linha 140 estabelece a posição inicial, a cor e o tamanho. Com a mensagem e suas características definidas, podemos dizer à máquina que desenhe a mensagem. A sub-rotina de impressão — que começa na linha 9000 — examina cada caractere em **A$**, um por vez, acha as instruções apropriadas nos conjuntos (veja programação de jogos, na página 101) e, em seguida, desenha o caractere na tela.

Modificando **A$** na linha 130 e, se necessário, a posição inicial na linha 140, podemos desenhar a mensagem que quisermos. Como a parte do programa que traça as letras é uma sub-rotina, teremos quantas mensagens desejarmos, mas devemos colocá-las sempre em **A$**.

O procedimento será muito útil, sobretudo se tivermos uma série de mensagens para exibir em diversos pontos da tela durante um jogo.

Comece seus programas na linha 120, ou seja, acima das linhas **DATA**; lembre-se, porém, de que as linhas **DIM**, **CLEAR** ou **PCLEAR** em geral precisam ficar no início. Quando quiser escrever uma mensagem, estabeleça cordões com as palavras escolhidas apenas no ponto em que elas forem necessárias ao programa. Em seguida, acrescente o **DRAW**, como na linha 140, e chame a sub-rotina de impressão com **GOSUB 9000**.

E FINALMENTE...

Uma vez elaboradas as instruções para desenhar um perfil, podemos posicioná-lo em qualquer direção, ou mesmo de cabeça para baixo. Basta, para isso, incluir uma nova instrução no cordão: **A**, seguida de um valor de 0 a 3.

A0 posiciona o desenho — no nosso exemplo, a casa — a 0 graus, ou seja, na vertical. **A1** desenha a casa deitada sobre seu lado direito — a 90 graus. **A2**, de cabeça para baixo — a 180 graus. **A3**, finalmente, traça a casa deitada sobre seu lado esquerdo — a 270 graus.

O controle da direção em que a imagem é traçada permite obter o mesmo desenho, de quatro maneiras diferentes, a partir de um único conjunto de instruções. Não será necessário, portanto, informar ao computador como desenhar cada versão.

Para ver como esse recurso funciona, digite e rode o programa a seguir.

```
10 PMODE 3,1
20 PCLS
30 SCREEN 1,0
40 S$="NR16E8F4U4R2D6F2D12L6U6L
4D6L6U12"
50 FOR K=1 TO 20
60 D=RND(200)+27:E=RND(140)+27:
C=RND(3)+1:A=RND(4)-1
70 DRAW"BM"+STR$(D)+","+STR$(E)
+"C"+STR$(C)+"A"+STR$(A)+"XS$;"
80 NEXT K
90 GOTO 90
```

```
30 SCREEN 2
40 S$="NR16E8F4U4R2D6F2D12L6U6L
4D6L6U12"
50 FOR K=1 TO 20
60 D=INT(RND(1)*200)+28:E=INT(R
ND(1)*140)+28:C=INT(RND(1)*15)+
1:A=INT(RND(1)*4)
70 DRAW "BM"+STR$(D)+","+STR$(E
)+"C"+STR$(C)+"A"+STR$(A)+"XS$;
"
80 NEXT K
90 GOTO 90
```

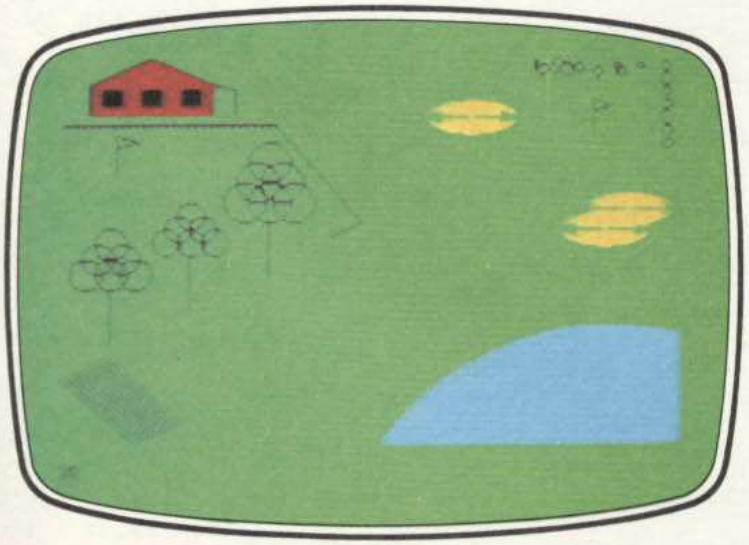
Serão desenhadas 20 casas, todas iguais, exceto na cor e na orientação.

O formato das casas é definido na linha 40. Na linha 60, quatro números aleatórios são escolhidos: **D** e **E** são as coordenadas da posição inicial, **C** é a cor, e **A**, a direção da imagem.

A linha 70 desenha a casa, acrescentando ao cordão todos os parâmetros gerados na linha 60. A função **STR$** converte as variáveis numéricas em cordões — na verdade, coloca o valor destas variáveis entre aspas. Por exemplo, se **D = 2**, então **STR$ = "2"**.

A linha 70 move, então, a "caneta" para uma posição qualquer e, usando uma cor aleatória, desenha, na direção escolhida ao acaso, as instruções contidas em **S$**. O **X** antes de **S$** significa "execute as instruções contidas em **S$**" e permite a adição de um cordão a outro durante o desenho, sem a necessidade de utilizar o sinal "+".

O emprego do **X** apresenta a vantagem de economizar a "memória de cordões" da máquina — que é limitada —, pois permite unir cordões sem criar novos cordões. O uso do "+", ao contrário, cria um novo cordão, que vai ocupar boa parte da memória que estamos querendo economizar.



O Spectrum usa arcos e círculos para desenhar um campo de golfe.

É possível colocar textos na tela gráfica do Apple II?

Infelizmente, os microcomputadores compatíveis com a linha Apple II não permitem misturar texto e gráficos na mesma tela. Essa limitação reduz bastante o campo de ação do iniciante que quer desenvolver jogos ou outros tipos de programas gráficos que precisam de alguma espécie de título ou identificação na tela.

Como os computadores da linha MSX e TRS-Color, os micros da linha Apple II também dispõem de um comando **DRAW**. Porém, se o objetivo deste recurso é semelhante em todos os computadores mencionados, a filosofia de sua utilização é bastante diferente, quando se trata do Apple.

O funcionamento do comando **DRAW** do Apple baseia-se nas *tabelas de forma* (*shape tables*), que residem em uma certa parte da memória e que determinam um verdadeiro "mapa" de movimentação do cursor gráfico, com base em números binários que identificam quais os bits que devem ficar acesos ou apagados na tela. A especificação de tabelas de forma será explicada em um artigo futuro. Basta saber, por enquanto, que letras e números de qualquer tamanho podem ser desenhados na tela, um a um, com o comando **DRAW**.

ASSEMBLER PARA O APPLE

Traduzir mnemônicos Assembly para código de máquina é uma tarefa extremamente cansativa. Deixe o Apple trabalhar por você: ele o fará com rapidez e facilidade.

O Assembler do Apple é bem mais curto do que os programas correspondentes para o Spectrum, o MSX e o TRS-Color. Isto ocorre porque seu microprocessador — o 6502 — tem bem menos instruções que o Z-80 e o 6809, os microprocessadores das outras máquinas.

Este programa não precisará trabalhar com a infinidade de mnemônicos do Z-80, nem terá de calcular os pós-bytes do 6809.

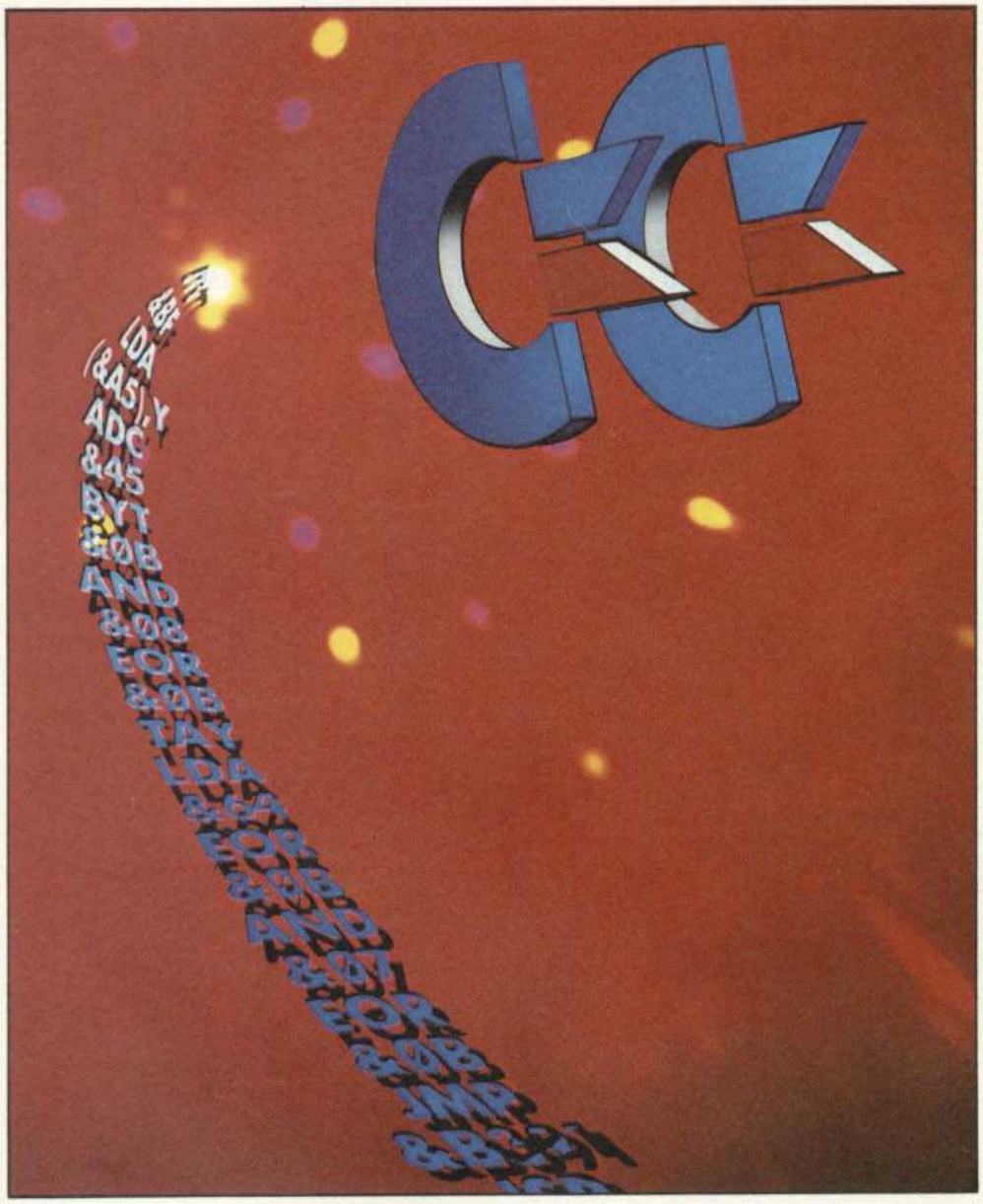
Devido ao seu tamanho, ele é mais rápido — o que não significa, porém, que sua velocidade seja comparável à dos programas escritos em código. Ele levará alguns minutos para traduzir um longo programa em Assembly.

É bom lembrar que o TK-2000 já vem com um míni-Assembler embutido.

O ASSEMBLER

```
10  HOME
20  DIM K$(77),K1(77),K2(77):H$
 = "0123456789ABCDEF":R$ =  CHR
$ (13)
30  DIM T$(200),RR(100),Z$(100)
40 P = 0: FOR II = 1 TO 76: REA
D K$(II),K1(II),K2(II): IF K$(I
I) <  > "*" THEN  NEXT
50  DATA    ADC,105,101,AND,41,
37,ASL,,6,BCC,,144,BCS,,176,BEQ
,,240,BIT,,36,BMI,,48
60  DATA  BRK,,,BYT,,-256
70  DATA   BNE,,208,BPL,,16,BVC
,,80,BVS,,112,CLC,24,,CLD,216,,
CLI,88,,CLV,184,
80  DATA  CMP,201,197,CPX,224,2
28,CPY,192,196,DEC,,198,DEX,202
,,DEY,136,,EOR,73,67
90  DATA  INC,,230,INX,232,,INY
,200,,JMP,,68,JSR,,24
100  DATA ,,,,,,,,,LDA,169,165,
LDX,162,166,LDY
110  DATA  160,164,LSR,,70,,,,,
,,,,,NOP,,234,,,,,,,,,ORA,9,5,
PHA,72,,PLA,104,,PLP
120  DATA  40,,PHP,8,,,,,,,,ROL
,,38,ROR,,102
130  DATA   RTI,64,,RTS,96,,SBC
,233,229,SEC,56,,SED,248,,SEI,1
20,,STA,,133,STX,,134
140  DATA  STY,,132,TAX,170,,TA
Y,168,,TSX,186,,TXA,138,,TXS,15
4,,TYA,152,
150  DATA  WOR,,-256,*,,
160  HOME : HTAB 16: VTAB 1: IN
VERSE : PRINT "MENU": NORMAL :
PRINT : PRINT : PRINT
170  PRINT  TAB( 10);"1 - LER O
 DISCO": PRINT : PRINT  TAB( 10
);"2 - GRAVAR NO DISCO": PRINT
: PRINT  TAB( 10);"3 - MONTAR"
180  PRINT : PRINT  TAB( 10);"4
 - EDITAR LIMHA": PRINT : PRINT
  TAB( 10);"5 - APAGAR LINHA":
PRINT : PRINT  TAB( 10);"6 - LI
STAR"
190  PRINT : PRINT  TAB( 10);"7
 - SAIDA": PRINT : PRINT : PRIN
T  TAB( 10);"SELECIONE A OPCAO
";
200  GET A$: IF  VAL (A$) < 1 O
R  VAL (A$) > 7 THEN 200
210 JJ =  VAL (A$): HOME
220  ON JJ GOSUB 1080,1100,250,
1120,1200,1230,1300: PRINT : PR
INT "QUALQUER TECLA PARA CONTIN
UAR"
230  GET A$: IF A$ = "" THEN 23
0
240  GOTO 160
250 K0 = 0:K9 = 0:PS = 0:P0 = 0
260 PS = PS + 1: IF PS <  = 3 T
HEN K = K0:P = P0: PRINT "ETAPA
 ";PS: GOTO 280
270 P0 = P: RETURN
280  GOSUB 980
290  GOSUB 910:OP$ = I$: IF  LE
FT$ (OP$,1) = "*" AND PS = 3 TH
EN  PRINT OP$
300  IF  LEFT$ (OP$,1) = "*" TH
EN 280
310  IF OP$ = "END" AND PS = 3
THEN  PRINT : PRINT " FIM. ENDE
RECO FINAL: ";P - 1
320  IF OP$ = "END" THEN 260
330  IF OP$ <  > "ORG" AND OP$
<  > "P%=" THEN 370
340  GOSUB 910:S = 0: IF  LEFT$
 (I$,1) = "*" THEN S = P:I$ =
RIGHT$ (I$, LEN (I$) - 1)
350 P =  VAL (I$) + S: IF PS =
3 THEN  PRINT "    ORG";P
360  GOTO 280
370  IF P = 0 THEN  PRINT "FALT
A ORG": RETURN
380  IF OP$ <  > "$P%=" AND OP$
 <  > "TXT" THEN 420
390  GOSUB 910: IF PS = 3 THEN
 PRINT "    TXT";I$; TAB( 20);
400  FOR J = 2 TO  LEN (I$):BY
=  ASC ( MID$ (I$ + I$,J,1)): I
F J >  =  LEN (I$) THEN BY = 13
410  GOSUB 870: NEXT J: GOTO 28
0
420  IF  LEN (OP$) <  > 3 THEN
 FOR I = 1 TO 1: GOTO 440
430  FOR I = 1 + 3 * ( ASC (OP$
) - 65) TO 76: IF OP$ = K$(I) T
HEN 490
440  NEXT : IF PS = 3 THEN  PRI
NT OP$
450  IF  LEFT$ (I$,1) = "." THE
N I$ =  RIGHT$ (I$, LEN (I$) -
1)
460  GOSUB 1010:RR(Q2) = P: IF
I$ = "" THEN 290
470  IF PS = 3 THEN  PRINT "(LI
NHA NAO RECONHECIDA)"
480  GOTO 280
490 AD$ = "$0":R = 0: IF K2(I)
<  > 0 THEN  GOSUB 910:AD$ = I$
:OP = K2(I)
500  IF PS = 3 THEN  PRINT " ";
OP$;: IF K2(I) <  > 0 THEN  PRI
NT " ";AD$;
510  IF  LEFT$ (AD$,1) = "$" TH
EN AD$ =  RIGHT$ (AD$, LEN (AD$
) - 1):OP = K1(I)
520  IF  RIGHT$ (AD$,1) <  > ")
" THEN 530
525 OP = OP - 20:AD$ =  MID$ (A
D$,2, LEN (AD$) - 2): IF OP = 4
8 THEN OP = 100
530  IF  RIGHT$ (AD$,2) = ";X"
THEN OP = OP + 16:AD$ =  LEFT$
(AD$, LEN (AD$) - 2)
540  IF AD$ = "A" THEN AD$"0":O
P = OP + 4
550  IF  RIGHT$ (AD$,2) <  > ";
Y" THEN 580
560 OP = OP + 16:AD$ =  LEFT$ (
AD$, LEN (AD$) - 2): IF  RIGHT$
 (OP$,1) <  > "X" THEN OP = OP
- 4:R = 65536
570  IF  RIGHT$ (AD$,1) = ")" T
HEN AD$ =  MID$ (AD$,2, LEN (AD
$) - 2):R = R - 65536
580 S = 1
590  IF AD$ = "" THEN 770
600 X$ =  LEFT$ (AD$,1): IF  LE
N (AD$) = 1 THEN BD$ = "": GOTO
 610
605 BD$ =  RIGHT$ (AD$, LEN (AD
$) - 1)
610  IF X$ = "*" THEN R = R + P
 * S:AD$ = BD$: GOTO 580
620  IF X$ = "+" THEN AD$ = BD$
: GOTO 590
630  IF X$ = "-" THEN AD$ = BD$
:S =  - S: GOTO 590
640 Q = 0
650  IF (X$ <  > "$" AND X$ <
> "&") OR BD$ < "0" OR BD$ >  =
 "G" THEN 690
660  FOR Q2 = 0 TO 15: IF  LEFT
$ (BD$,1) <  >  MID$ (H$,Q2 + 1
,1) THEN 680
670 Q = Q * 16 + Q2: IF  LEN (B
```

- TRADUÇÃO DE MNEMÔNICOS ASSEMBLY PARA CÓDIGO HEXADECIMAL
- CÁLCULO DE SALTOS E DESVIOS
- MANIPULAÇÃO DE RÓTULOS
- TRANSFERÊNCIA DO PROGRAMA EM CÓDIGO PARA A MEMÓRIA



```
D$) = 1 THEN BD$ = "": GOTO 660
675 BD$ = RIGHT$ (BD$, LEN (BD
$) - 1): GOTO 660
680 NEXT :R = R + Q * S:AD$ =
BD$: GOTO 580
690 IF X$ < "A" OR X$ > "Z" TH
EN 720
700 I$ = AD$: GOSUB 1010: IF I$
 < > "" THEN GOSUB 1050
710 R = R + RR(Q2) * S:AD$ = I$
: GOTO 580
720 IF X$ < "0" OR X$ > "9" TH
EN R = 0: GOTO 760
730 IF AD$ < "0" OR AD$ > "9"
THEN 750
740 Q = Q * 10 + ASC (AD$) - 4
8: IF LEN (AD$) = 1 THEN AD$ =
 "": GOTO 730
745 AD$ = RIGHT$ (AD$, LEN (AD
$) - 1): GOTO 730
750 R = R + Q * S: GOTO 580
760 IF PS = 3 THEN PRINT "(EN
DERECAMENTO INVALIDO)";
770 PP = (OP - ( ABS (OP) > =
8) * 8 * INT (OP / 8) > = 4)
* 4 + (OP > = - 4 AND OP < =
 - 1) * 4
772 RP = R - P - 2:RP = ABS (
ABS ((RP < 0) * INT (RP / 256)
) * 256 - ABS (RP - (RP > = 2
56) * 256 * INT (RP / 256)))
775 IF LEFT$ (OP$,1) = "B" AN
D (0 = PP AND OP > 0) THEN R =
RP
780 IF LEFT$ (OP$,1) = "J" OR
 LEFT$ (OP$,1) = "W" THEN R =
R + 65536
790 IF R > 255 THEN OP = OP +
8
800 K2 = K2(I):PA = ABS ( ABS
((OP < 0) * INT (OP / 16)) * 1
6 - ABS (OP - (OP > = 16) * 1
6 * INT (OP / 16)))
805 IF PA = 10 THEN K2 = 0
810 IF PS = 3 THEN PRINT TAB
( 16);:BY = P / 256: GOSUB 890:
BY = P - 32768: GOSUB 890: GOSU
B 850
820 IF OP > = 0 THEN BY = OP
/ 256: GOSUB 870:BY = OP: GOSUB
 880
830 IF K2 = 0 THEN 280
840 GOSUB 850:BY = R - 256 *
INT (R / 256): GOSUB 880:BY = R
 / 256: GOSUB 870: GOTO 280
850 IF PS = 3 THEN PRINT " ";
860 RETURN
870 IF INT (BY) < = 0 THEN
RETURN
880 P = P + 1:YY = ABS ( ABS (
(BY < 0) * INT (BY / 256)) * 2
56 - ABS (BY - (BY > = 256) *
 256 * INT (BY / 256))): IF PS
 = 3 THEN POKE P - 1,YY
890 BY = ABS ( ABS ((BY < 0) *
 INT (BY / 256)) * 256 - ABS
(BY - (BY > = 256) * 256 * IN
T (I / 256)))
895 YY = ABS ( ABS ((BY < 0) *
 INT (BY / 16)) * 16 - ABS (B
Y - (BY > = 16) * 16 * INT (B
Y / 16))): IF PS = 3 THEN PRIN
T MID$ (H$,BY / 16 + 1,1); MID
$ (H$,YY + 1,1);
900 RETURN
910 IF K > N THEN I$ = "END":
RETURN
920 K1 = K9 + 1: IF K9 > = LE
N (T$(K)) THEN I$ = "/FALTANDO/
": RETURN
930 K9 = K1: IF MID$ (T$(K),K1
,1) = " " THEN 920
940 IF K9 > LEN (T$(K)) THEN
I$ = MID$ (T$(K),K1,K9 - K1):
RETURN
950 IF MID$ (T$(K),K9,1) < >
 " " THEN K9 = K9 + 1: GOTO 940
960 I$ = MID$ (T$(K),K1,K9 - K
```

```
1)
970  RETURN
980  IF K9 <  = LEN (T$(9)) AN
D PS = 3 THEN  IF  LEN (T$(K))
> K9 - 1 THEN  PRINT  RIGHT$ (T
$(K), LEN (T$(K)) - K9 + 1);
990 K = K + 1:K9 = 0: IF PS = 3
 THEN  PRINT
1000  RETURN
1010 X$ = ""
1020  IF I$ < "A" OR I$ >  = "[
" THEN 1040
1030 X$ = X$ +  LEFT$ (I$,1): I
F  LEN (I$) = 1 THEN I$ = "": G
OTO 1020
1035 I$ =  RIGHT$ (I$, LEN (I$)
 - 1): GOTO 1020
1040  IF I$ <  > "" THEN RETURN
1050  FOR Q2 = 1 TO VV: IF X$ =
 Z$(Q2) THEN 1070
1060  NEXT :VV = VV + 1:Z$(VV)
= X$:Q2 = VV:RR(VV) = 32768
1070  RETURN
1080  INPUT "NOME DO ARQUIVO ";
AR$:D$ = "": REM   CTRL-D
1090  PRINT D$;"OPEN";AR$: PRIN
T D$;"READ";AR$: INPUT N: FOR J
 = 1 TO N: INPUT T$(J): NEXT :
PRINT D$;"CLOSE";AR$: RETURN
1100  INPUT "NOME DO ARQUIVO ";
AR$:D$ = "": REM   CTRL-D
1110  PRINT D$;"OPEN";AR$: PRIN
T D$;"WRITE";AR$: PRINT N: FOR
J = 1 TO N: PRINT T$(J): NEXT :
 PRINT D$;"CLOSE";AR$: RETURN
1120 K = 0: INPUT "QUAL O NUMER
O DA LINHA ";K: HOME
1140 IP$ = "": PRINT K;: INPUT
IP$: IF IP$ = "" THEN  RETURN
1150 K2 = K / 10: IF K2 > N THE
N K2 = N + 1:N = N + 1
1160 IF K2 < .1 THEN K2 = .1
1170 IF K2 =  INT (K2) THEN 11
 90
1180 K2 =  INT (K2) + 1: FOR K3
 = N TO K2 STEP  - 1:T$(K3 + 1)
 = T$(K3): NEXT :N = N + 1
1190 T$(K2) = IP$:K = K + 10: G
OTO 1140
1200 K = 0: INPUT "QUAL O NUMER
O DA LINHA ";K:K2 = K / 10
1210  IF K2 > N OR K2 < 1 OR K2
 >  INT (K2) THEN  RETURN
1220  FOR K3 = K2 TO N:T$(K3) =
 T$(K3 + 1): NEXT :N = N - 1: R
ETURN
1230  IF N = 0 THEN  RETURN
1240 K = 0:K2 = 0: HOME : INPUT
 "QUAL A PRIMEIRA E A ULTIMA LI
NHA ";K,K2:K1 = K / 10:K2 = K2
/ 10
1250  IF K2 > N THEN K2 = N
1260  IF K1 < 1 THEN K1 = 1
1270  HOME : FOR K3 = K1 TO K2:
 PRINT " ";K3 * 10;" ";T$(K3):
NEXT : RETURN
1300  END
```

Nas linhas 1080 e 1100, a variável **D$** deve conter o caractere produzido pelas teclas <**CTRL**> e <**D**> pressionadas ao mesmo tempo. **D$**, portanto, não está vazia. **CTRL + D** é um caractere de controle para uso do disco.

COMO FUNCIONA O PROGRAMA

Uma vez digitado o programa, rode-o. Normalmente, ele é colocado na área livre da memória que vai de 300 até 3EF, em hexa, ou de 768 até 1007, em decimal. Se quisermos colocá-lo em código em outro local, devemos proteger esta área da memória (veja página 88).

Depois que o programa for rodado, surgirá um menu na tela. Para entrar um programa em Assembly, deve-se escolher a opção 4, que diz "EDITAR LINHA". O Assembler usa linhas BASIC com números múltiplos de 10.

A primeira linha — número 10 — deve conter o endereço inicial do programa em código. Ela ficará mais ou menos como esta:

10 org 800

... determinando que o programa em código seja colocado na memória a partir deste endereço. Se tivermos protegido outra área da memória para colocar nosso programa em código, a origem deve ser o endereço inicial daquela porção da memória.

Depois da primeira linha, os números de linha múltiplos de dez aparecem automaticamente, toda vez que se pressiona a tecla **RETURN**. Cada linha deve conter apenas um mnemônico e seus operandos. Caso se pretenda inserir uma linha entre duas já existentes, utiliza-se um número de linha intermediário — e o programa se encarrega da inserção. Os rótulos, ou *labels* — que indicam os locais para onde o programa será desviado —, devem ficar no início da linha, precedidos de um ponto. Os rótulos precisam ter mais de uma letra e não podem ser iguais a nenhum mnemônico. São empregados mnemônicos padrão do 6502, com uma exceção: o uso do ponto e vírgula no lugar da vírgula. Isto é necessário porque o programa utiliza a instrução **INPUT** para receber as linhas. Uma vírgula provocaria a mensagem de "EXTRA IGNORADO".

Caso deseje evitar a tradução de uma linha, coloque um asterisco antes dela. Ao pressionar **RETURN**, surgirá a próxima linha, mas se a mesma tecla for pressionada novamente, sem que qualquer coisa tenha sido escrita na linha, o programa retornará ao menu.

O menu fornece a opção de listar o programa. Se outra tecla for pressionada, volta-se ao menu. Podemos também optar por apagar uma linha, ou simplesmente escrever a nova, com o mesmo número no modo de edição.

Quando estivermos satisfeitos com o programa em Assembly que digitamos, podemos acionar a opção de "montagem" do programa. O Apple mostrará, então, os mnemônicos acompanhados dos códigos hexa correspondentes, ao mesmo tempo em que coloca estes códigos na memória.

Pode-se gravar o Assembler em fita ou disco, como qualquer outro programa (veja página 53). Os mnemônicos — ou programa fonte — devem ser gravados e lidos em disco usando as opções correspondentes.

O programa em código, por sua vez, pode ser gravado em fita ou disco utilizando-se o monitor do Apple, ou por meio do comando **BSAVE**.

Para rodá-lo, deve-se sair do Assembler, por meio da opção "SAÍDA", e digitar:

CALL 800

O que fazer quando um programa longo — como este Assembler — não funciona depois de digitado?

Mesmo o programador mais experiente tem problemas ao digitar programas longos. Sempre se comete algum erro, e é muito difícil localizá-lo.

A maioria dos erros provoca uma mensagem que orienta o programador na direção correta (veja página 141). Mas programas longos seguem caminhos tortuosos, entram em laços e fazem saltos, de modo que o número de linha indicado na mensagem de erro algumas vezes não ajuda em nada.

Por tudo isso, é necessário contar com uma função de rastreamento, como a função **TRACE**, disponível no Apple. Com ela podemos ver o número da linha que está sendo executada, o que torna mais fácil diagnosticar os erros. Para utilizá-la, basta digitar **TRACE** no modo imediato, isto é, sem número de linha na frente. Depois, rode o programa e veja o rastreamento em ação. A função é desativada por **NONTRACE**.

Assim, não descarte o programa se ele não funcionar na primeira tentativa. Procure pelos erros mais comuns. Se ocorrer um "**OUT OF DATA**", por exemplo, verifique com atenção suas linhas **DATA**. Se você tiver esquecido um número, ou mesmo uma vírgula, o programa não funcionará.

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR LINHA

LINHA	FABRICANTE	MODELO
Apple II +	Appletronica	Thor 2010
Apple II +	CCE	MC-4000 Exato
Apple II +	CPA	Absolutus
Apple II +	CPA	Polaris
Apple II +	Digitus	DGT-AP
Apple II +	Dismac	D-8100
Apple II +	ENIAC	ENIAC II
Apple II +	Franklin	Franklin
Apple II +	Houston	Houston AP
Apple II +	Magnex	DM II
Apple II +	Maxitronica	MX-2001
Apple II +	Maxitronica	MX-48
Apple II +	Maxitronica	MX-64
Apple II +	Maxitronica	Maxitronic I
Apple II +	Microcraft	Craf II Plus
Apple II +	Milmar	Apple II Plus
Apple II +	Milmar	Apple Master
Apple II +	Milmar	Apple Senior
Apple II +	Omega	MC-400
Apple II +	Polymax	Maxxi
Apple II +	Polymax	Poly Plus
Apple II +	Spectrum	Microengenho I
Apple II +	Spectrum	Spectrum ed
Apple II +	Suporte	Venus II
Apple II +	Sycomig	SIC I
Apple II +	Unitron	AP II
Apple II +	Victor do Brasil	Elppa II Plus
Apple II +	Victor do Brasil	Elppa Jr.
Apple IIe	Microcraft	Craft IIe
Apple IIe	Microdigital	TK-3000 IIe
Apple IIe	Spectrum	Microengenho II
MSX	Gradiente	Expert GPC-1
MSX	Sharp	Hotbit HB-8000
Sinclair Spectrum	Microdigital	TK-90X
Sinclair Spectrum	Timex	Timex 2000
Sinclair ZX-81	Apply	Apply 300
Sinclair ZX-81	Engebras	AS-1000
Sinclair ZX-81	Filcres	NEZ-8000
Sinclair ZX-81	Microdigital	TK-82C
Sinclair ZX-81	Microdigital	TK-83
Sinclair ZX-81	Microdigital	TK-85
Sinclair ZX-81	Prologica	CP-200
Sinclair ZX-81	Ritas	Ringo R-470
Sinclair ZX-81	Timex	Timex 1000
Sinclair ZX-81	Timex	Timex 1500
TRS-80 Mod. I	Dismac	D-8000
TRS-80 Mod. I	Dismac	D-8001/2
TRS-80 Mod. I	LNW	LNW-80
TRS-80 Mod. I	Video Genie	Video Genie I
TRS-80 Mod.III	Digitus	DGT-100
TRS-80 Mod.III	Digitus	DGT-1000
TRS-80 Mod.III	Kemitron	Naja 800
TRS-80 Mod.III	Prologica	CP-300
TRS-80 Mod.III	Prologica	CP-500
TRS-80 Mod.III	Sysdata	Sysdata III
TRS-80 Mod.III	Sysdata	Sysdata Jr.
TRS-80 Mod.IV	Multix	MX-Compacto
TRS-80 Mod.IV	Sysdata	Sysdata IV
TRS-Color	Codimex	CS-6508
TRS-Color	Dynacom	MX-1600
TRS-Color	LZ	Color 64
TRS-Color	Microdigital	TKS-800
TRS-Color	Prologica	CP-400

CLASSIFICAÇÃO DOS MICROCOMPUTADORES POR NOME

FABRICANTE	MODELO	PAÍS	LINHA
Appletronica	Thor 2010	Brasil	Apple II +
Apply	Apply 300	Brasil	Sinclair ZX-81
CCE	MC-4000 Exato	Brasil	Apple II +
CPA	Absolutus	Brasil	Apple II +
CPA	Polaris	Brasil	Apple II +
Codimex	CS-6508	Brasil	TRS-Color
Digitus	DGT-100	Brasil	TRS-80 Mod.III
Digitus	DGT-1000	Brasil	TRS-80 Mod.III
Digitus	DGT-AP	Brasil	Apple II +
Dismac	D-8000	Brasil	TRS-80 Mod. I
Dismac	D-8001/2	Brasil	TRS-80 Mod. I
Dismac	D-8100	Brasil	Apple II +
Dynacom	MX-1600	Brasil	TRS-Color
ENIAC	ENIAC II	Brasil	Apple II +
Engebras	AS-1000	Brasil	Sinclair ZX-81
Filcres	NEZ-8000	Brasil	Sinclair ZX-81
Franklin	Franklin	USA	Apple II +
Gradiente	Expert GPC1	Brasil	MSX
Houston	Houston AP	Brasil	Apple II +
Kemitron	Naja 800	Brasil	TRS-80 Mod.III
LNW	LNW-80	USA	TRS-80 Mod. I
LZ	Color 64	Brasil	TRS-Color
Magnex	DM II	Brasil	Apple II +
Maxitronica	MX-2001	Brasil	Apple II +
Maxitronica	MX-48	Brasil	Apple II +
Maxitronica	MX-64	Brasil	Apple II +
Maxitronica	Maxitronic I	Brasil	Apple II +
Microcraft	Craft II Plus	Brasil	Apple II +
Microcraft	Caft IIe	Brasil	Apple IIe
Microdigital	TK-3000 IIe	Brasil	Apple IIe
Microdigital	TK-82C	Brasil	Sinclair ZX-81
Microdigital	TK-83	Brasil	Sinclair ZX-81
Microdigital	TK-85	Brasil	Sinclair ZX-81
Microdigital	TK-90X	Brasil	Sinclair Spectrum
Microdigital	TKS-800	Brasil	TRS-Color
Milmar	Apple II Plus	Brasil	Apple II +
Milmar	Apple Master	Brasil	Apple II +
Milmar	Apple Senior	Brasil	Apple II +
Multix	MX-Compacto	Brasil	TRS-80 Mod.IV
Omega	MC-400	Brasil	Apple II +
Polymax	Maxxi	Brasil	Apple II +
Polymax	Poly Plus	Brasil	Apple II +
Prologica	CP-200	Brasil	Sinclair ZX-81
Prologica	CP-300	Brasil	TRS-80 Mod.III
Prologica	CP-400	Brasil	TRS-Color
Prologica	CP-500	Brasil	TRS-80 Mod.III
Ritas	Ringo R-470	Brasil	Sinclair ZX-81
Sharp	Hotbit HB-8000	Brasil	MSX
Spectrum	Microengenho I	Brasil	Apple II +
Spectrum	Microengenho II	Brasil	Apple IIe
Spectrum	Spectrum ed	Brasil	Apple II +
Suporte	Venus II	Brasil	Apple II +
Sycomig	SIC I	Brasil	Apple II +
Sysdata	Sysdata III	Brasil	TRS-80 Mod.III
Sysdata	Sysdata IV	Brasil	TRS-80 Mod.IV
Sysdata	Sysdata Jr.	Brasil	TRS-80 Mod.III
Timex	Timex 1000	USA	Sinclair ZX-81
Timex	Timex 1500	USA	Sinclair ZX-81
Timex	Timex 2000	USA	Sinclair Spectrum
Unitron	AP II	Brasil	Apple II +
Victor do Brasil	Elppa II Plus	Brasil	Apple II +
Victor do Brasil	Elppa Jr.	Brasil	Apple II +
Video Genie	Video Genie I	USA	TRS-80 Mod. I

UM LOGOTIPO PARA CADA MODELO DE COMPUTADOR

INPUT foi especialmente projetado para microcomputadores compatíveis com as sete principais linhas existentes no mercado.
Os blocos de textos e listagens de programas aplicados apenas a determinadas linhas de micros podem ser identificados por meio dos seguintes símbolos:


Sinclair ZX-81


TRS-80


TK-2000


MSX


Spectrum


TRS-Color


Apple II

Quando o emblema for seguido de uma faixa, então tanto o texto como os programas que se seguem passam a ser específicos para a linha indicada.

NO PRÓXIMO NÚMERO

PROGRAMAÇÃO BASIC

Aprenda a empregar as cadeias de caracteres. Elas serão úteis em quase todo tipo de programa

CÓDIGO DE MÁQUINA

Um programa que dimensiona os saltos e faz a tradução dos mnemônicos Assembly: o que mais poderíamos desejar?

APLICAÇÕES

Precisão e velocidade são fundamentais para a digitação e o processamento de textos. Domine o teclado.

PROGRAMAÇÃO BASIC

Códigos de controle podem substituir comandos em BASIC. Veja as vantagens.