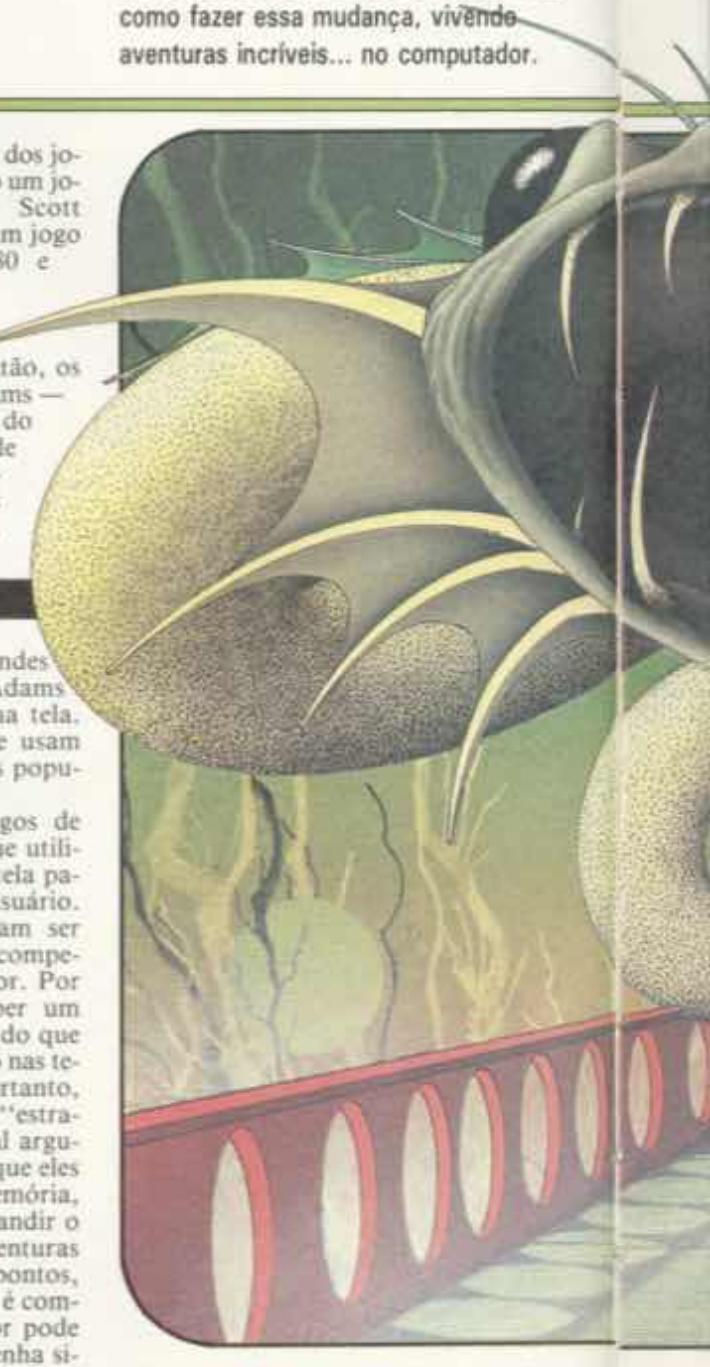
COMO PLANEJAR UMA AVENTURA

Sair da rotina e fazer da vida uma aventura errante: quantos de nós já não sonhamos com isso? Veja como fazer essa mudança, vivendo aventuras incríveis... no computador.



Os jogos de aventura são uma boa alternativa para quem está cansado dos arcaicos videogames e fliperamas. Neles, o jogador é totalmente envolvido por um mundo fantástico, criado unicamente pela imaginação do programador.

A ORIGEM DOS JOGOS DE AVENTURAS

A idéia de criar jogos de aventura surgiu da popularidade, nos EUA e na Grã-Bretanha, dos jogos de ação não-computadorizados, tais como *Masmorras e Dragões* e, ao mesmo tempo, do desejo de se utilizar os computadores para fazer coisas mais interessantes do que processar dados.

Por outro lado, os jogos não-computadorizados não atendiam plenamente à demanda do público, que exigia emoções cada vez mais fortes. Essa demanda só foi satisfeita pelos jogos de aventuras programados para o computador.

Em *Masmorras e Dragões* os jogadores assumem o papel de certos personagens, que lutam encarniçadamente (na imaginação) num lugar conhecido como *Masmorra*. Esse lugar, por sua vez, é criado por outro jogador, que faz o papel do *Senhor da Masmorra*. Nos jogos de aventura, o programador vive um personagem parecido com este, podendo inventar, assim, seu próprio mundo.

Ao contrário de outros jogos, em *Masmorras e Dragões* os jogadores não podem escolher as características de seus personagens; isso depende inteiramente do jogo. Em algumas versões mais sofisticadas, os jogadores podem selecionar o equipamento e os mantimentos que os personagens vão usar, desde que isso seja feito no início do jogo, antes de começar a aventura. O primeiro jogo de aventura destinado a computadores foi escrito para máquinas de grande porte, em linguagem FORTRAN, e não em BASIC. O programa ocupou aproximadamente 300k de memória, ou seja, bem mais do que a memória RAM total da maioria dos micros pessoais. Apesar do seu tamanho e complexidade, esse jogo conquistou milhares de usuários de computadores nos Estados Unidos e no Canadá; eles passaram a dedicar suas noites a tentar decifrar os mistérios do jogo.

Na realidade, a popularização dos jogos de aventura começou quando um jovem programador chamado Scott Adams desenvolveu, em 1978, um jogo para microcomputador TRS-80 e provou que era possível escrever uma aventura razoável em um espaço de memória bem menor. Desde então, os temas de jogos adotados por Adams — A Terra da Aventura, O Refúgio do Pirata, O Mistério do Parque de Diversões etc. — têm sido usados e recusados inúmeras vezes pelos escritores de outros jogos.

TIPOS DE AVENTURAS

Como as aventuras para grandes computadores, os jogos de Adams apresentavam somente textos na tela. Na verdade, os programas que usam apenas textos são ainda os mais populares.

Existem ainda inúmeros jogos de aventura para computadores que utilizam ilustrações desenhadas na tela para situar, orientar e motivar o usuário. Entretanto, os gráficos precisam ser muito sofisticados, para poder competir com a imaginação do jogador. Por exemplo, você poderia conceber um monstro bem mais horripilante do que seria capaz de reproduzir, mesmo nas telas gráficas mais avançadas. Portanto, é bem possível que os gráficos "estraguem" sua diversão: o principal argumento contra a sua utilização é que eles desperdiçam grande parte da memória, que poderia ser usada para expandir o campo da aventura. Certas aventuras apresentam uma contagem de pontos, sempre que uma etapa da busca é completada. Desse modo, o jogador pode avaliar seu desempenho, caso tenha sido derrotado em algum estágio. Alguns programas chegam ao ponto de classificar esse desempenho com conceitos, que podem ir de "palhaço" a "perito".

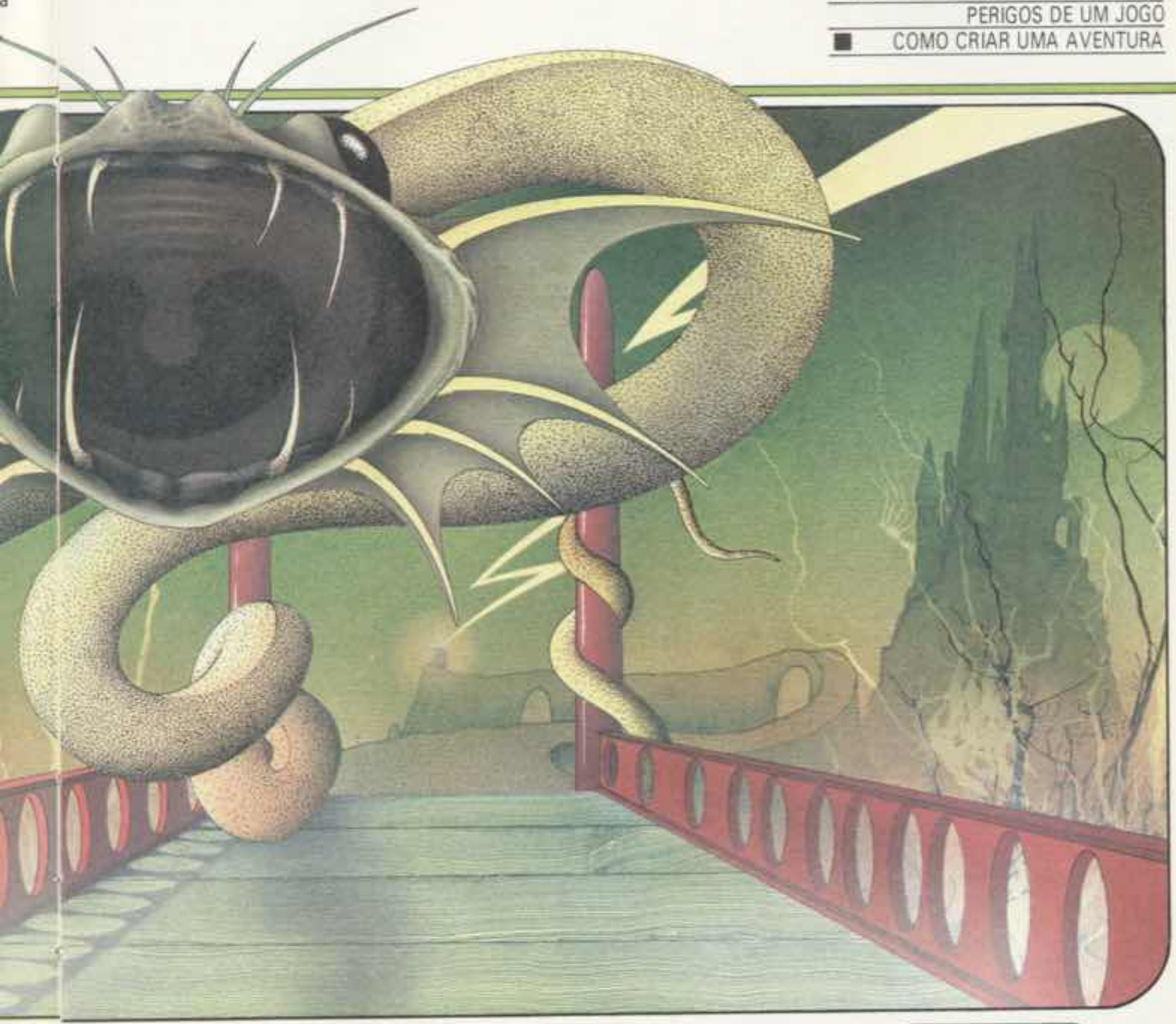
Outros jogos não fornecem qualquer dica de como o jogador está se saindo ou da proximidade do desfecho.

Seja qual for o caso, porém, a diversão maior está na solução de uma série aparentemente infinita de charadas, depois da qual a aventura chega ao fim. Alguns costumam durar muitas horas, ou dias, podendo ser interrompidos e retomados a qualquer momento.

UMA HISTÓRIA SEM FIM

Normalmente, quando se inicia uma aventura através do comando **RUN**, são

- JOGOS DE AVENTURA
- COMO JOGAR
- DICAS PARA VENCER OS PERIGOS DE UM JOGO
- COMO CRIAR UMA AVENTURA



dadas informações sobre o mundo em que se acaba de entrar: este pode ser um lugar exótico na Terra, um planeta distante, ou um local imaginário. O jogo pode acontecer em épocas diversas. Geralmente, são fornecidas informações como: quem governa o lugar; quem é você e como fazer para atingir o objetivo e ganhar o jogo. Depois disso, aparecerá uma primeira descrição do local, provavelmente dizendo algo como:

```
 VOCÊ SE ENCONTRA PRÓXIMO A UM
ENORME CALDEIRÃO CHEIO DE UM
LÍQUIDO VERDE BORBULHANTE. HÁ
UM CHEIRO DEMONÍACO NO AR. UMA
COLHER ESTÁ NO CHÃO.
VOCÊ PODE IR PARA O LESTE
                    OESTE
                    NORTE

E AGORA?
```





Agora, você tem que decidir o que fazer. Será conveniente usar a colher para misturar o líquido ou mesmo experimentá-lo? Não seria melhor deixar isso de lado? Ou talvez fosse preferível procurar um recipiente para levar um pouco do líquido verde com você?

Se decidir usar a colher, digite algo assim: **PEGUE COLHER**; a resposta do computador será: OK ou VOCÊ AINDA NÃO PODE PEGAR A COLHER!, ou alguma outra mensagem.

Em cada etapa do jogo você deve informar ao computador exatamente o que pretende fazer. A forma de fazer isso dependerá apenas do tipo de jogo. A maioria deles espera você digitar seus comandos no computador através de um verbo seguido de substantivo; por exemplo: **PEGUE COLHER**.

Jogos mais sofisticados aceitam sentenças completas, mas isso é uma exceção à regra. Esses tipos de jogos permitem-lhe dizer algo como: **MATE O INSETO PISANDO NELE ENQUANTO CANTA "O TICO-TICO NO FUBÁ"**.

A maioria dos jogos de aventura consegue entender abreviaturas das palavras válidas. Por exemplo, é muito comum, numa aventura, digitar **N** em vez de **NORTE**. As direções podem ser pontos cardeais — **N, S, L** e **O** — ou colaterais — **SE, SO, NE** e **NO** —, ou instruções como **PARA CIMA, PARA BAIXO**.

Um jogo típico de aventura é baseado numa grade de locais possíveis, que assume geralmente a forma de um quadrado. Dependendo da imaginação do programador, esses lugares podem representar ambientes tão variados como os quartos de um castelo, os subterrâneos de uma mina etc.

DICAS PARA VENCER

Habitualmente, existe apenas uma solução para a aventura: por exemplo, encontrar um pote de ouro e levá-lo para o fim do arco-íris, ou matar um guerreiro medieval e escapar ileso etc. Até chegar a esse objetivo, o jogador precisa resolver toda uma série de problemas. O mais provável é que ele tenha que fazer muitas tentativas diferentes até concluir a aventura.

Existem algumas regras básicas e dicas que o ajudarão a resolver mais rapidamente a maioria dos jogos. Quase todos os objetos que você encontrar nas aventuras serão de alguma utilidade. A existência de muitas pistas completamente falsas representa um desperdício de memória, embora seja necessário es-

tar sempre atento para alguns objetos que podem ser "facas de dois gumes". Por exemplo, o jogador poderia estar carregando uma sacola com moedas de ouro para passar pelo pedágio de uma ponte; caso ele se decidisse a nadar no rio, porém, o peso delas o faria afundar. Como regra geral, é aconselhável tentar carregar o maior número possível de objetos, mas algumas vezes apenas uma parte deles lhe será realmente útil.

Quase sempre, a maioria dos objetos é usada somente uma vez durante uma aventura. Mas há exceções: uma espada, por exemplo, poderia ser usada muitas vezes para derrubar gigantes, dragões ou bandidos. Dessa forma, se você for obrigado a limitar o número de objetos que leva consigo, deve ter em mente que é mais seguro descartar os que já foram usados uma vez.

Outra regra geral: desenhe *sempre* um mapa, marcando nele os nomes dos aposentos, a posição dos sentinelas e dos objetos distribuídos pelos quartos, *todas* as entradas e saídas com suas direções e outros pontos de referência que lhe possam ser úteis.

O mapa o fará economizar tempo e esforço quando você estiver retornando para algum ponto anterior — manobra que você repetirá várias vezes durante o jogo. Se tiver que abandonar algum objeto, por não poder carregar tudo, não se esqueça de marcar a posição dele no mapa. A importância do mapa está em que ele lhe permitirá ter certeza de que explorou todas as possibilidades da aventura — o que, em muitos casos, salvará sua "vida".

Certos jogos permitem que você peça um inventário do que está carregando. Assim, quando estiver diante de uma charada, procure saber exatamente o que tem em mãos, digitando **INVENTÁRIO**, **LISTA**, ou algo semelhante, dependendo da aventura.

Há jogos, ainda, que dão ao jogador a possibilidade de pedir ajuda. A forma de fazê-lo varia, mais uma vez, conforme o conteúdo da aventura. Freqüentemente, porém, você obterá algo assim, em resposta ao seu pedido:
NENHUMA AJUDA POSSÍVEL.

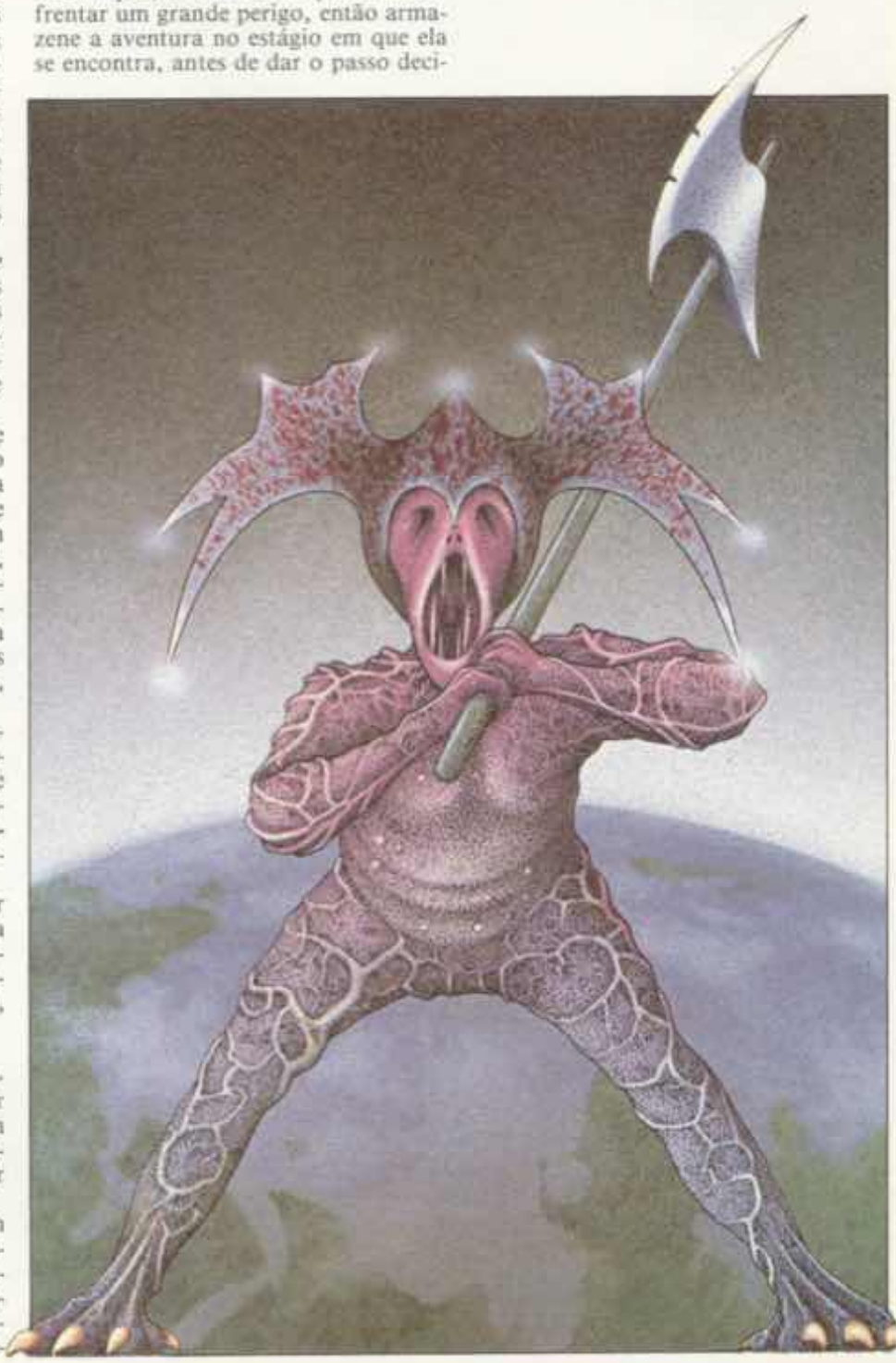
Alguns jogos seguem cuidadosamente o enredo de um livro específico (por exemplo, *A Ilha do Tesouro*, escrito em 1833 por Robert Louis Stevenson). Nestes casos, é altamente recomendável ler o livro em questão.

Outros jogos inspiram-se apenas em alguns capítulos ou idéias de um romance famoso. Assim, se você encontrar numa aventura algum elemento que o faça recordar um texto literário e tiver dificuldades para resolver um determinado problema, tente encontrar a resposta consultando o livro. Um dicionário de sinônimos pode ser de grande utilidade, permitindo que o jogador explore todas as variações possíveis de uma determinada frase. Por exemplo: o programador pode ter usado no jogo a palavra POLIR, em vez de LUSTRAR.

Uma última dica: se o jogo permitir o uso do comando **GRAVAR** em algumas etapas, e você estiver prestes a enfrentar um grande perigo, então armazene a aventura no estágio em que ela se encontra, antes de dar o passo decisivo, pois, se você for "morto", poderá voltar atrás e continuar a partir do ponto em que parou.

CRIE AS SUAS PRÓPRIAS AVENTURAS

A criação de aventuras é uma boa maneira de dominar a linguagem BASIC. Quase todos os aspectos importan-



tes dessa linguagem são utilizados nesse tipo de jogo: variáveis, matrizes, formatos de telas, cadeias alfanuméricas, comandos **IF**, e assim por diante.

Muitos jogos de aventuras à venda ainda são escritos em BASIC, pois não há muita necessidade de velocidade de execução do programa.

Antes de programar um jogo desse gênero, você precisa ter uma idéia bem clara do que pretende fazer e do enredo da aventura. A seguir, planeje cuidadosamente a trama com suas charadas, perigos, objetos etc.

Para começar, pegue uma folha de papel e faça um rascunho de algumas idéias. Não se preocupe se ainda não tem uma visão completa da ordem do jogo: o que você precisa inicialmente é de uma noção do roteiro, do local da aventura e de algumas charadas para o jogador resolver.



Quanto espaço de memória necessito para escrever um jogo de aventura?

Nas próximas lições de INPUT desenvolveremos uma aventura simples, ocupando cerca de 5K de BASIC. Como todo primeiro passo, esse jogo não trará episódios de grande emoção. Seu objetivo é apenas introduzi-lo ao mundo fascinante da aventura.

Os melhores jogos têm um grande número de aposentos e charadas e são muito difíceis, devido ao efeito crescente de problemas cada vez mais complicados. Com uma quantidade limitada de memória, você será forçado a determinar um número menor de problemas, que por sua vez serão mais difíceis do que o normal. Isso, contudo, não é muito satisfatório.

Um jogo como esse exige um computador com, no mínimo, 16K de memória; mas sugerimos uma máquina com 48 ou 64K para jogos realmente desafiantes.

O que devo fazer se cair numa armadilha durante a aventura?

Não desista; tente novamente quando tiver uma nova idéia ou estratégia.

Se você realmente empacar, os jogos de aventuras comercializados oferecem um folheto com dicas e respostas. Muitas vezes, é preciso solicitar pelo correio um mapa da solução.

Outro recurso para ajudá-lo são as revistas do gênero, que costumam apontar a solução completa de aventuras complexas.

Além de livros, você pode inspirar-se em filmes, programas de TV e peças teatrais, ou mesmo inventar novas histórias. Pode também buscar idéias em outras aventuras. Mas o melhor recurso para sua inspiração é fazer trabalhar a imaginação.

Ao mesmo tempo, procure alcançar um certo equilíbrio entre o desafio e as condições para vencê-lo. Não é conveniente criar uma aventura que qualquer um poderia resolver no espaço de meia hora. Em contrapartida, seu jogo não deve apresentar problemas insolúveis. A regra básica é: "dê aos jogadores uma chance"... mas não exagere.

Tente não deixar muitos aposentos ou locais vazios. Além de ocupar muito espaço de memória, estes não contribuem para a resolução dos problemas e tornam o enredo cansativo.

Por outro lado, procure evitar que as suas primeiras aventuras sejam excessivamente complexas, para que os problemas que aparecem ao longo da história possam ser resolvidos por quem ainda não tem prática; familiarize-se com o tema antes de aventurar-se em qualquer enredo extravagante; mantenha-se alerta quanto ao espaço restante de memória, pois programas como esses tendem a se tornar gigantescos.

No jogo de aventura que será elaborado nas próximas partes do curso de *Programação de jogos*, encontraremos muitas declarações **REM** (comentários, para entender o fluxo do programa). Entretanto, para economizar memória em jogos com longas aventuras, é preferível trabalhar sem tais declarações, mesmo que no começo elas facilitem a criação da trama.

Você poderá também poupar espaço de memória na descrição dos aposentos. Descrições muito curtas, porém, tendem a estragar todo o prazer suscitado pela aventura.

O ESPAÇO RESTANTE DE MEMÓRIA

Ao escrever um longo jogo de aventura, é fácil descobrir quando se está chegando ao limite de memória do computador. Obviamente, os problemas de desperdício de memória tornam-se mais graves no caso de máquinas com memórias menores (na verdade, seria perda de tempo tentar escrever um programa de aventura em um computador que tenha menos do que 16K de RAM).

Digite a seguinte linha, em modo direto:

```
PRINT (PEEK 23730 + 256* PEEK
23731) - (PEEK 23653 + 256*
PEEK 23654)
```

O que foi digitado leva em consideração tanto o espaço ocupado pelo programa como o usado para armazenar as variáveis. A melhor contagem para descobrir a quantidade de memória restante só será possível, portanto, depois que o programa tiver sido rodado.

Para saber quanto espaço está sobrando, subtraia esse valor da quantidade total de memória RAM existente no computador.

Nos microcomputadores das linhas TRS-80 e TRS-Color, o que resta da memória pode ser descoberto digitando-se:

```
PRINT MEM
```

Essa instrução levará em conta o espaço ocupado pelo programa e pelas variáveis. Assim, é melhor rodar antes o programa, durante seu desenvolvimento (se isso for possível), a fim de conseguir uma indicação real do total de memória usado por ele.

Para saber quanto espaço está sobrando, subtraia esse valor da quantidade total de memória RAM existente no computador. Há uma maneira bem fácil de se aumentar o espaço livre de memória no TRS-Color para seus programas em BASIC. Antes de começar a programar, digite esses dois comandos **POKE**:

```
POKE 25,6
POKE 26,1
NEW
```

Isso economizará uns 6K extras para seu programa, pois enganará o computador, colocando o programa em BASIC dentro de um espaço normalmente reservado para gráficos de alta resolução.

Neste caso os comandos **POKE** excluem os comandos para gráficos na sua aventura. Se eles existirem, o seu programa será alterado.

Quando você armazenar o programa em fita ou disco e quiser carregá-lo mais tarde, não se esqueça de usar antes os comandos **POKE** e **NEW**.

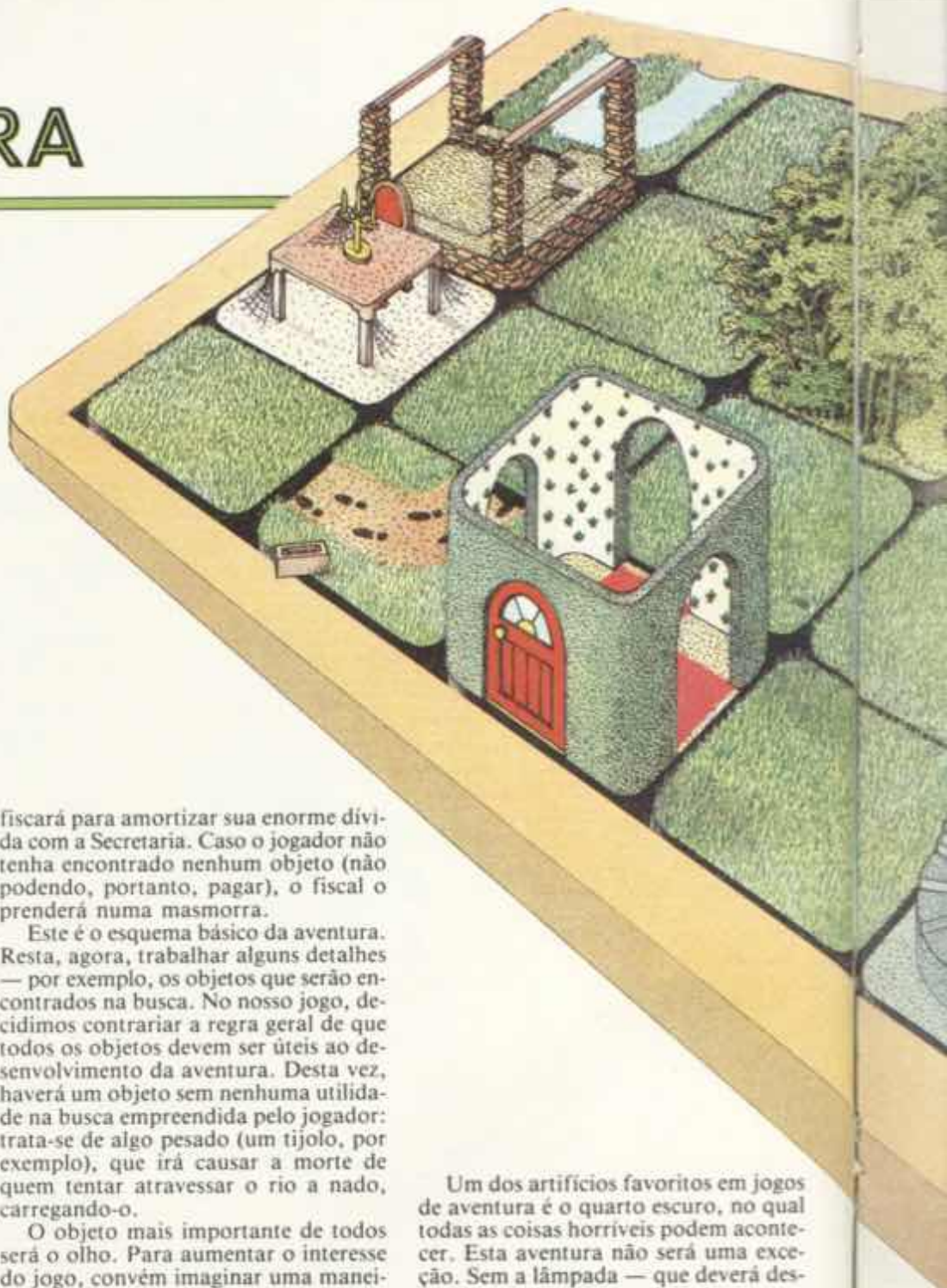
Nos microcomputadores das linhas Apple e MSX, o que resta da memória pode ser descoberto digitando-se:

```
PRINT FRE(0)
```

A função **FRE** (abreviatura de **FREe**, ou livre, em inglês) retorna o espaço livre na memória RAM, ou seja, aquele não ocupado pelo programa e pelas variáveis.

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.

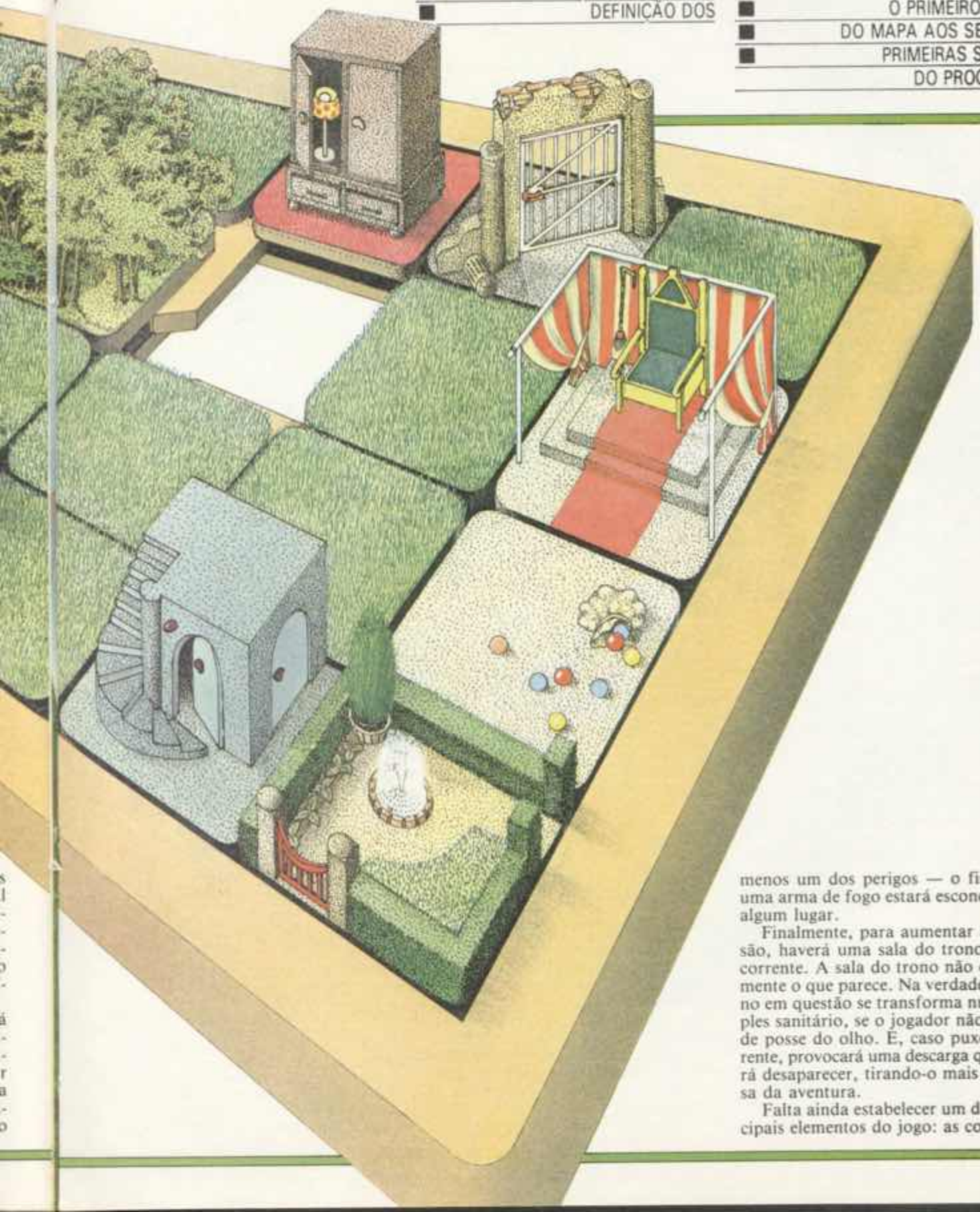
Este é o esquema básico da aventura. Resta, agora, trabalhar alguns detalhes — por exemplo, os objetos que serão encontrados na busca. No nosso jogo, decidimos contrariar a regra geral de que todos os objetos devem ser úteis ao desenvolvimento da aventura. Desta vez, haverá um objeto sem nenhuma utilidade 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

■ CRIAÇÃO DO ROTEIRO
■ DEFINIÇÃO DOS PERSONAGENS E OBJETOS
■ O PRIMEIRO MAPA
■ DO MAPA AOS SETORES
■ PRIMEIRAS SEÇÕES DO PROGRAMA



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

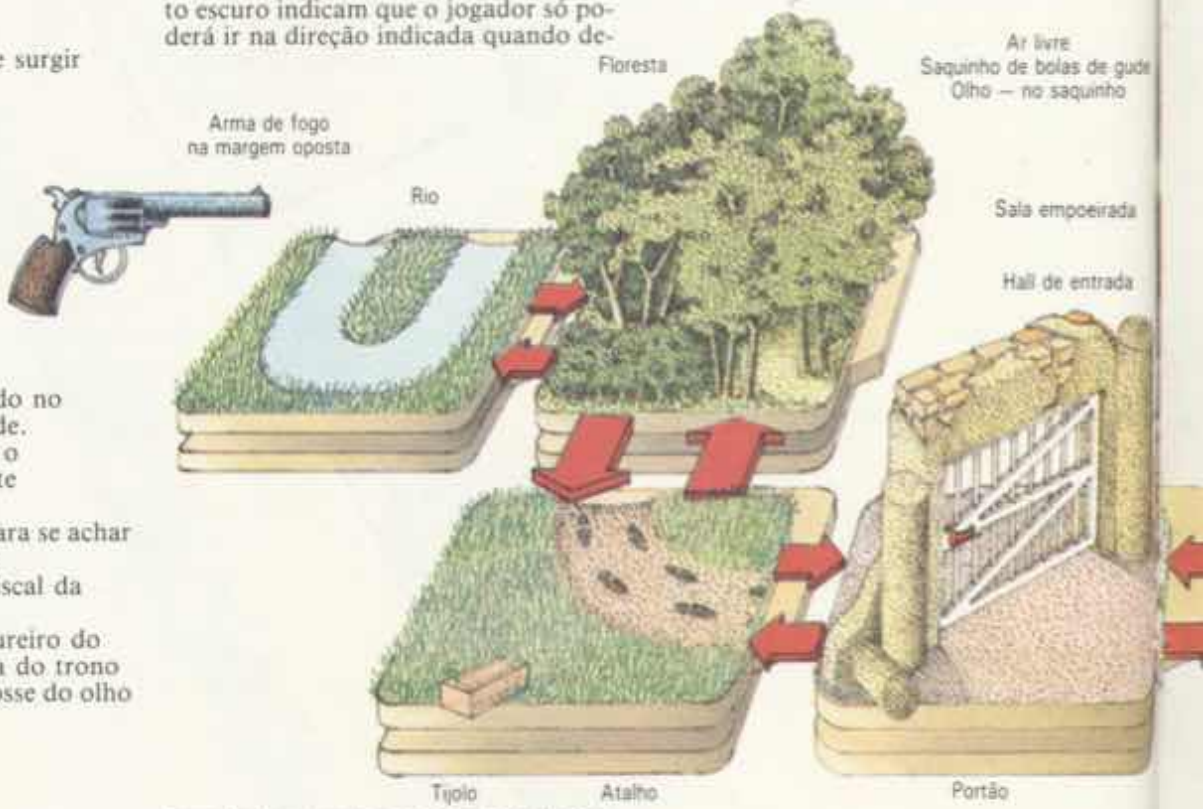
As faixas listradas que saem do quarto escuro indicam que o jogador só poderá ir na direção indicada quando determinadas condições forem satisfeitas. No nosso caso, a condição é estar de posse da lâmpada acesa, para enxergar as saídas.

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.

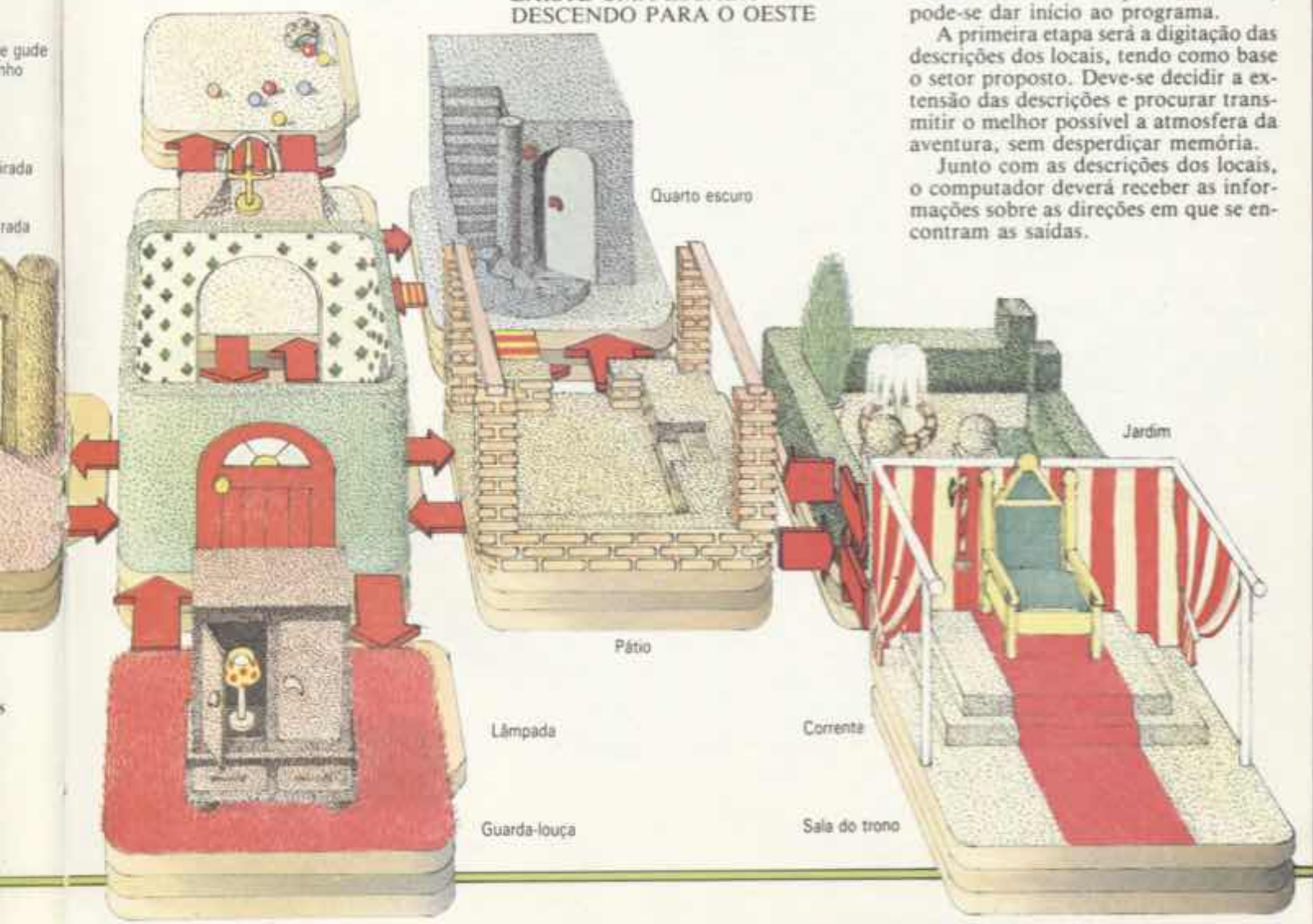
Uma vez numerados os quadrados e transferidos os detalhes, o conjunto dos setores terá uma aparência semelhante à da ilustração da página 231.

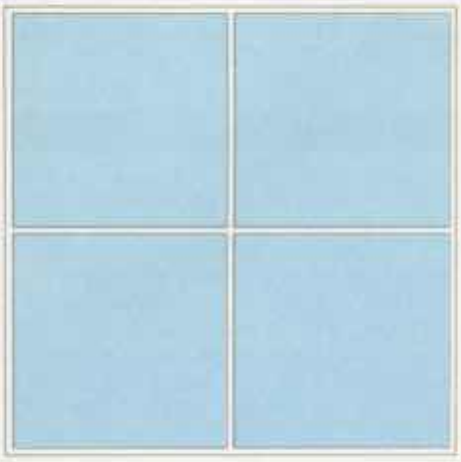
COMO COMEÇAR O PROGRAMA

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

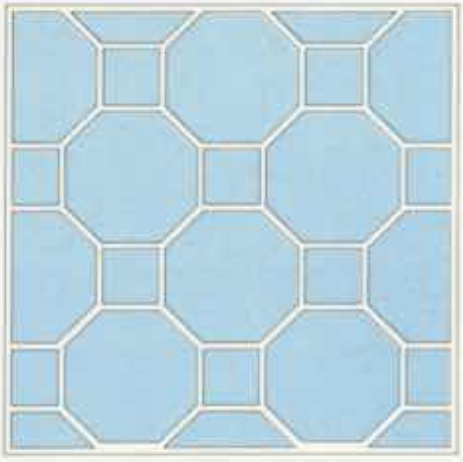
A primeira etapa será a digitação das descrições dos locais, tendo como base o setor proposto. Deve-se decidir a extensão das descrições e procurar transmitir o melhor possível a atmosfera da aventura, sem desperdiçar memória.

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
```

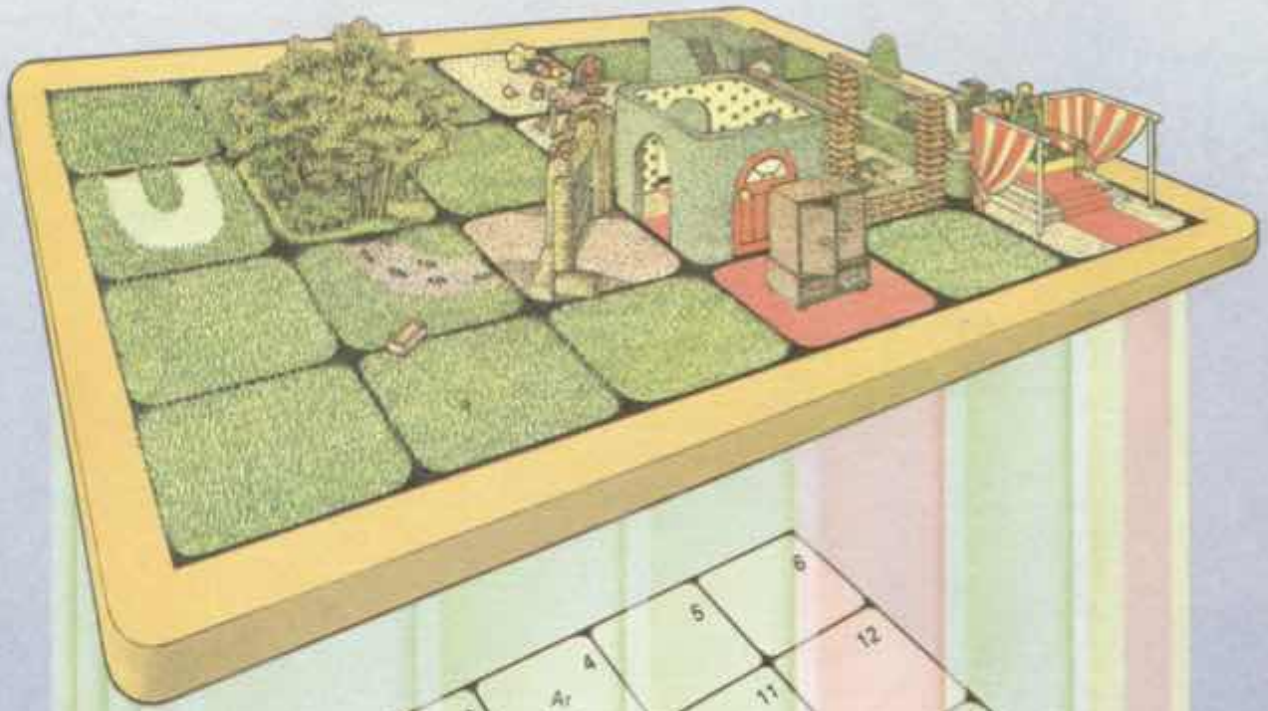
Não se preocupe com o uso repetido de comentários REM incluídos na memória. Nesta etapa inicial do desenvolvimento do programa, o importante é saber o que cada trecho do programa faz, ou qual o número do local a que determinada descrição se refere. Os comentários sempre podem ser eliminados posteriormente. 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.

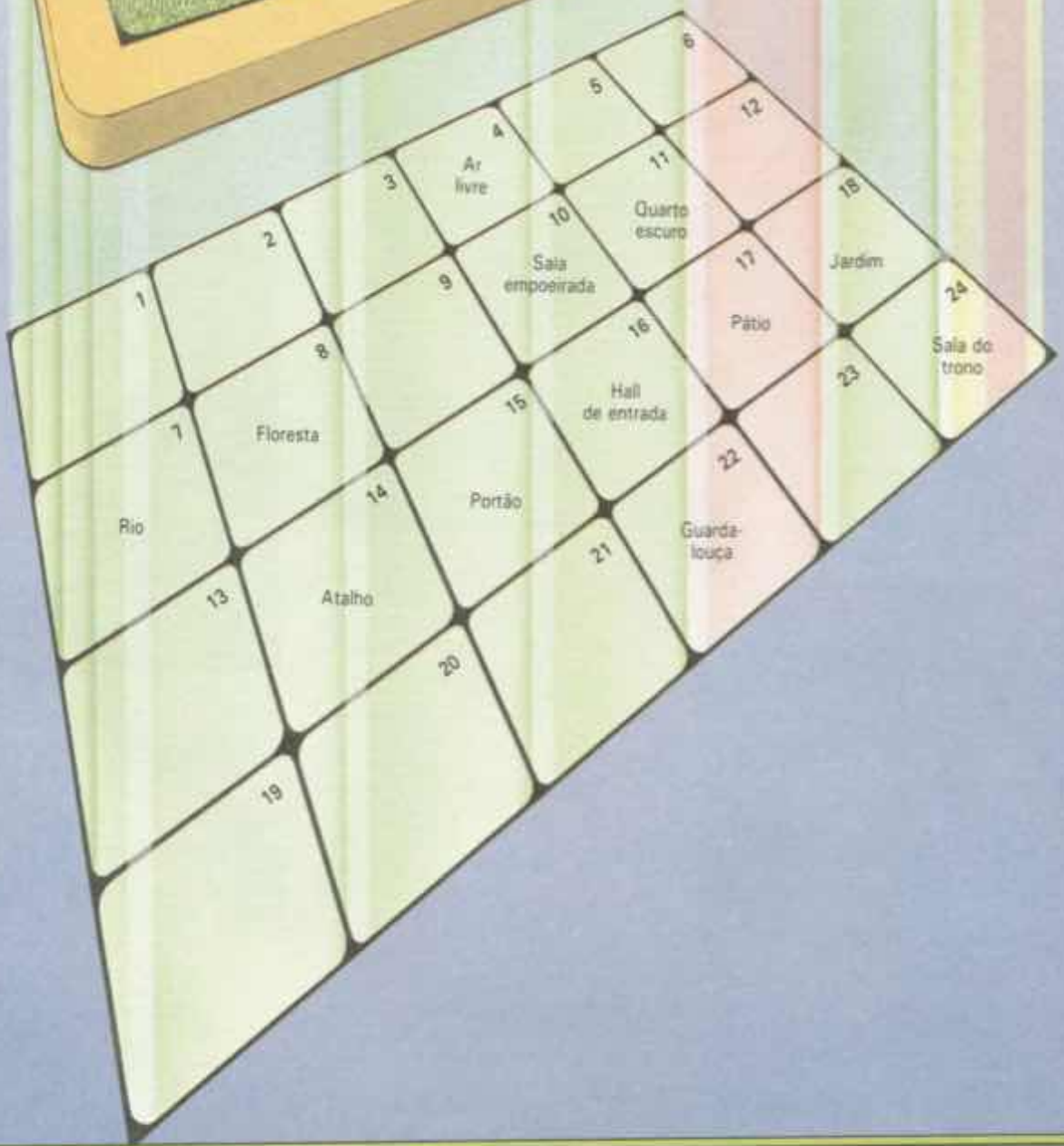
Finalmente, existe um **RETURN** após as seções do programa, pois cada descrição de local será chamada por uma instrução **GOSUB**.

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

COMO MOVIMENTAR O AVENTUREIRO

Para descobrir os mais estranhos lugares sem sair de casa, basta entrar no mundo da aventura. Veja aqui como proceder para dar movimento ao aventureiro e iniciar as explorações.

Você já digitou o programa contendo as descrições dos locais que fazem parte desta aventura? Então deve estar ansioso para fazer com que o jogador comece a explorá-los, possibilitando-lhe deslocar-se de um local para outro. Para que isso aconteça, você precisará definir os movimentos possíveis a partir de um determinado local; deverá estabelecer critérios que permitam julgar as respostas dadas pelo jogador e, finalmente, determinar os caminhos que podem ser percorridos.

Apresentaremos, adiante, algumas rotinas necessárias ao desenvolvimento do programa de aventuras. Estas rotinas serão responsáveis pela correta descrição da localização do jogador e pela apresentação das saídas que lhe serão permitidas. Você digitará cada opção e aprenderá a escrever a seção de programa que movimenta o jogador no mundo da aventura, de acordo com a opção que escolheu.

INÍCIO

É fundamental conhecer a localização exata do jogador, a qualquer momento, dentro do jogo. Para isso, o programa usa uma variável **L**, que indica Local. O valor dessa variável muda de acordo com a localização do jogador, a cada movimento realizado.

Para começar, você deve informar ao computador qual a posição do jogador no início da aventura. A primeira seção de um programa para fazer isso é dada em seguida. Carregue a parte do programa que você já possui, por meio do comando **LOAD**, e adicione estas novas linhas:

```
100 CLS : LET DA=0: LET TA=0:
LET LA=0
270 REM **POSICAO INICIAL**
280 LET L=15
290 GOTO 330
```

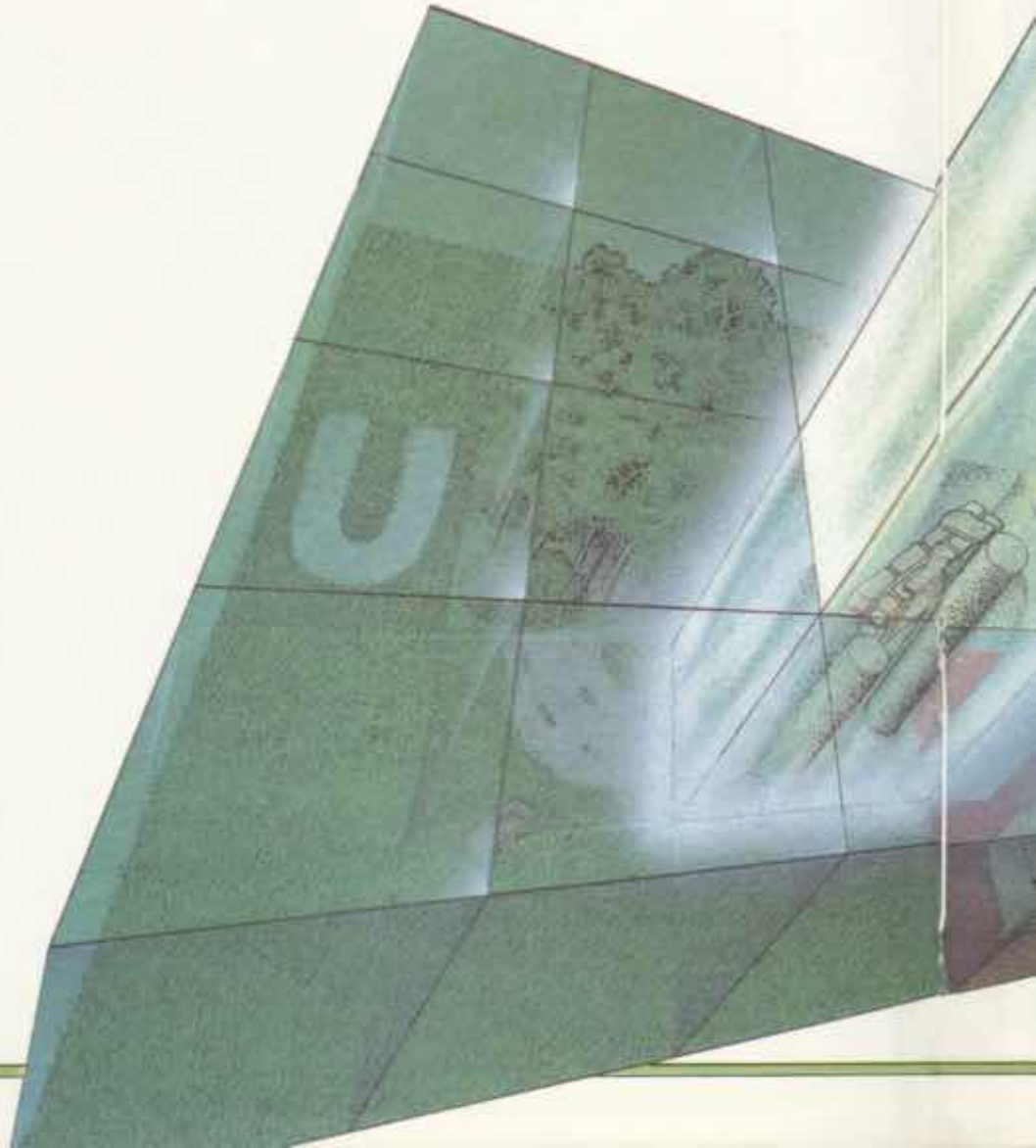
```
270 REM **POSICAO INICIAL**
280 L=15
290 GOTO 330
```

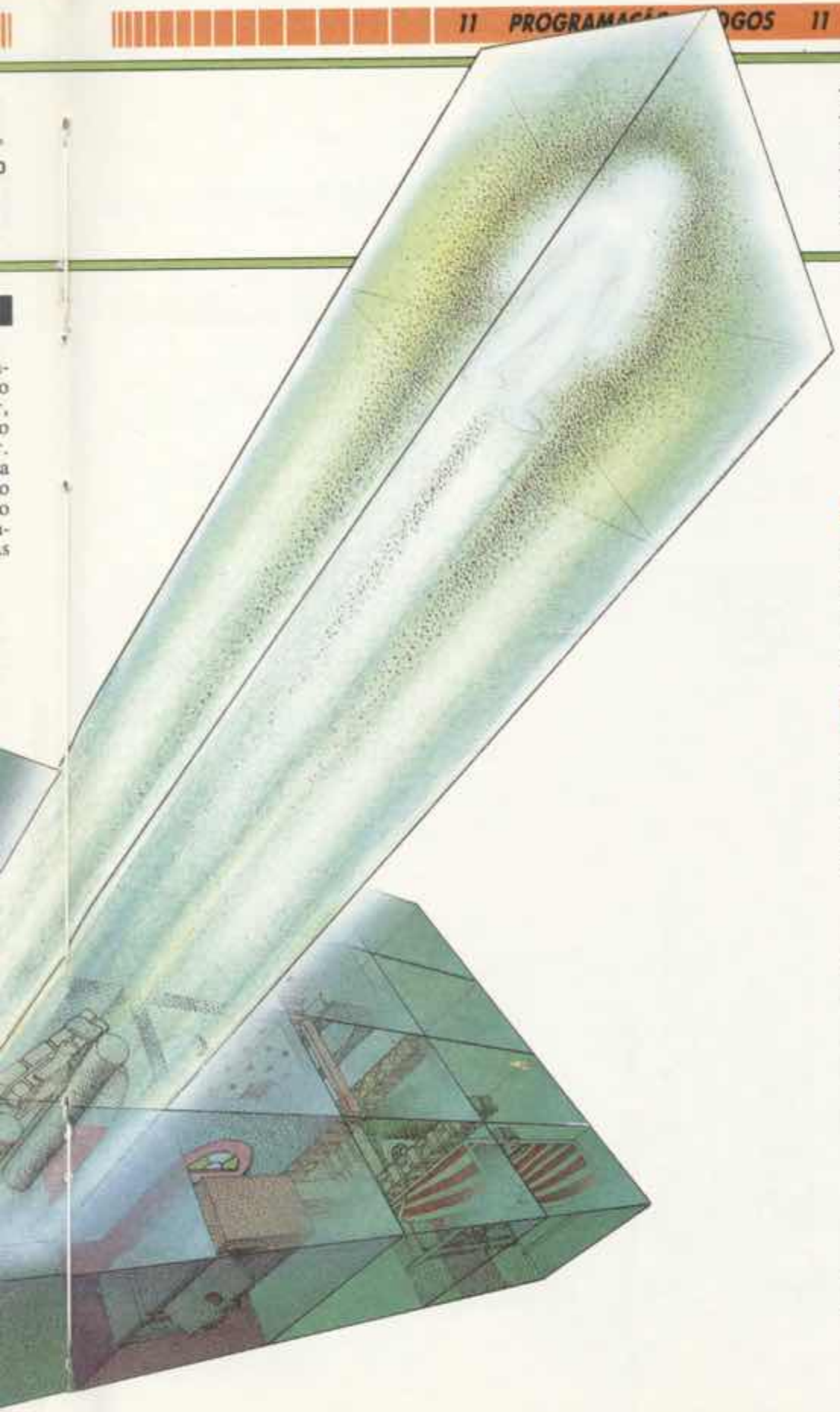
O número 15 indica a localização da porta de entrada da Cidade Oculta. Se você desejar que o jogador inicie a aventura em outro local, bastará mudar o valor da variável **L**. Adiante, você verá como ajustar o valor de **L** durante o jogo, de modo que passe a indicar localização diferente. Por enquanto, devemos preparar o programa para receber do jogador informações indicativas do caminho que pretende seguir.

RESPOSTAS

Para que o computador possa compreender e agir corretamente de acordo com as respostas dadas pelo jogador, você deve dar à máquina um conjunto de palavras que ela passe a identificar.

Neste estágio, o computador precisa conhecer apenas as quatro direções, o que pode ser feito por meio do cordão **RS**. Esse cordão é carregado com as palavras indicativas de direção, colocadas em uma instrução **DATA**.





- APRESENTAÇÃO E DESCRIÇÃO DOS LOCAIS
- DIREÇÕES POSSÍVEIS
- A MOVIMENTAÇÃO DO AVENTUREIRO

S

```
110 REM **PREPARACAO DAS MATRI
ZES DE RESPOSTAS**
120 DIM R$(19,40): DIM R(19)
130 FOR K=1 TO 4: READ R$(K),R
(K): NEXT K
150 DATA "NORTE",1,"SUL",1,"LE
STE",1,"OESTE",1
```

TK

```
110 REM**PREPARAR MATRIZES DAS
RESPOSTAS**
120 DIM R$(19),R(19)
130 FOR K=1 TO 4:READ R$(K),R(K
):NEXT
150 DATA NORTE,1,SUL,1,LESTE,1,
OESTE,1
```

Observe que as matrizes foram dimensionadas na linha 120, podendo agora armazenar todas as respostas que o jogo requer. Como precisamos, inicialmente, de quatro direções, apenas os quatro primeiros elementos das matrizes **R$** e **R** serão utilizados. A instrução **FOR...NEXT,** na linha 130, que lê **R$** e **R** por meio do comando **READ**, varia, assim, de um a quatro. As direções e os valores a ela associados estão colocados na instrução **DATA,** na linha 150.

Essas informações, entretanto, ainda não podem ser usadas pelo jogador. Antes, será necessário dotar o programa de uma rotina que indique onde ele está localizado.

COMO ENCONTRAR O LOCAL

Para que o jogador saiba onde se encontra, após efetuar cada movimento, é necessário fornecer-lhe uma descrição de sua posição. Você já digitou esta des-

crição. Ela está nas linhas de comentários, que se iniciam com a palavra **REM**. Agora, só nos falta uma rotina que associe o número contido na variável **L**, que indica a posição, à descrição correspondente. Neste ponto, os usuários do Spectrum devem digitar uma rotina extra:

```
 20 DIM G(11,4): POKE 23658,8
 30 FOR N=1 TO 4: FOR M=1 TO
 11: READ G(M,N): NEXT M: NEXT
N
40 DATA 0,0,0,5020,0,0,5050,
5080,0,5110,0
50 DATA 5140,0,0,5180,5210,
5240,5270,5300,0,0,0
60 DATA 0,5330,0,5360,0,0,0,0
,0,0,0
70 DATA 1010,1150,1240,1310,
1410,1460,1500,1360,1080,1550
,3110
300 REM **ACHAR O LOCAL**
310 CLS
330 IF L<11 THEN  GOSUB G(L,1)
: GOTO 400
340 IF L<21 THEN  GOSUB G(L-10
,2): GOTO 400
350 IF L<26 THEN  GOSUB G(L-20
,3)
```

```
300 REM **ACHAR LOCAL**
310 CLS
330 IF L<11 THEN ON L GOSUB 0,0
,0,5020,0,0,5050,5080,0,5110:GO
TO 400
340 IF L<21 THEN ON L-10 GOSUB
5140,0,0,5180,5210,5240,5270,53
00,0,0:GOTO 400
350 IF L<26 THEN ON L-20 GOSUB
0,5330,0,5360
```

Antes de escrever esse tipo de rotina, verifique se o número correspondente a cada descrição de localização está correto. Começando com a localização 1, faça uma lista dos números de linhas correspondentes a cada descrição. Se não houver descrição para uma determinada localização, escreva o número 0. Nesta aventura, já dispomos de descrição para a localização de número 4, mas não para as de número 1, 2 e 3.

De posse do conjunto contendo os números de linhas correspondentes à descrição de cada localização, você pode começar a escrever a rotina. Em todos os computadores, exceto o Spectrum, esta rotina é constituída de uma seqüência de operações que verifica o valor da variável **L** e usa o comando **ON...GOSUB**. No caso do Spectrum, que não possui este comando, o número das linhas é obtido de uma matriz.

Nos outros programas, o conjunto contendo os números de linhas está nas instruções colocadas nas linhas 330 a 350. Inicia com a localização 1 no começo da linha 330 e termina com a localização 24 no final da linha 350.

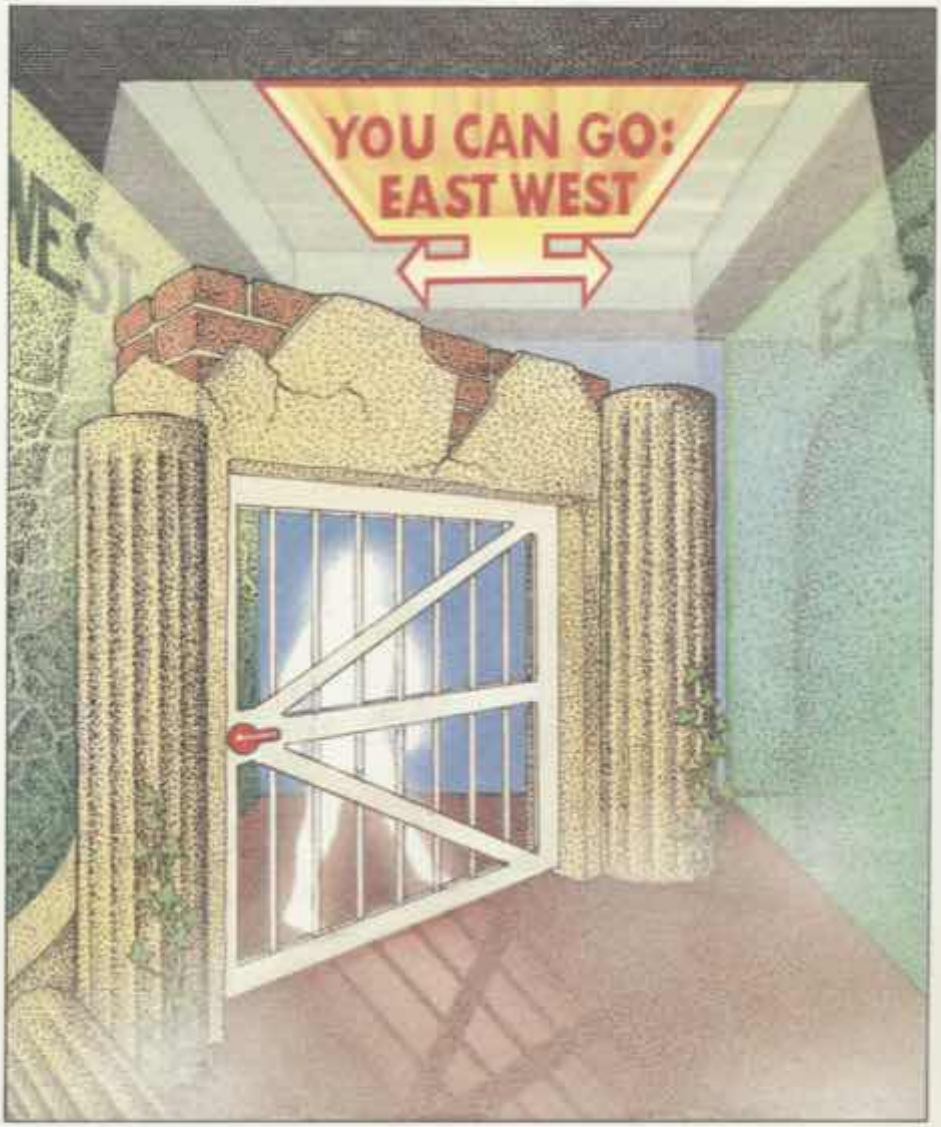
No Spectrum, use o valor de **L** para ter acesso a um elemento da matriz construído pelas instruções apresentadas nas linhas 20 e 30. Note que a matriz está superdimensionada em relação à quantidade de locais existentes na aventura. Os números excedentes são todos iguais a zero e não têm nenhuma influência no desenrolar da aventura. O dimensionamento extra foi feito porque a matriz será utilizada em partes do programa que mais tarde apresentaremos.

Uma observação válida somente para os usuários do Spectrum: o comando **POKE**, na linha 20, faz com que o computador aceite apenas letras maiúsculas. Isto evitará problemas no momento de se comparar as respostas dadas pelo jogador com as armazenadas na variável **RS**.

APRESENTAÇÃO DAS DIREÇÕES

Além da descrição do local onde se encontra, o jogador também precisa conhecer as saídas disponíveis. O programa deve verificar, pelas variáveis **N**, **E**, **S**, e **W**, quais são estas saídas. A informação é necessária porque nem sempre se pode sair em todas as direções, estando em um determinado local. A próxima seção do programa indica as saídas.

```
390 REM **APONTAR DIRECOES**
```



```
400 IF DA<>1 THEN  PRINT '"POD
E SEGUIR":
410 IF N>0 THEN  PRINT TAB 11;
"NORTE"
420 IF E>0 THEN  PRINT TAB 11;
"LESTE"
430 IF S>0 THEN  PRINT TAB 11;
"SUL"
440 IF W>0 THEN  PRINT TAB 11;
"OESTE"
```

```
390 REM **INDICAR DIRECOES**
400 IF L<>11 OR (LA=1 AND OB(6)
=-1) THEN PRINT:PRINT"PODE SEGU
IR ";ELSE 460
410 IF N>0 THEN PRINT TAB(13);"
NORTE"
420 IF E>0 THEN PRINT TAB(13);"
LESTE"
430 IF S>0 THEN PRINT TAB(13);"
SUL"
440 IF W>0 THEN PRINT TAB(13);"
OESTE"
```

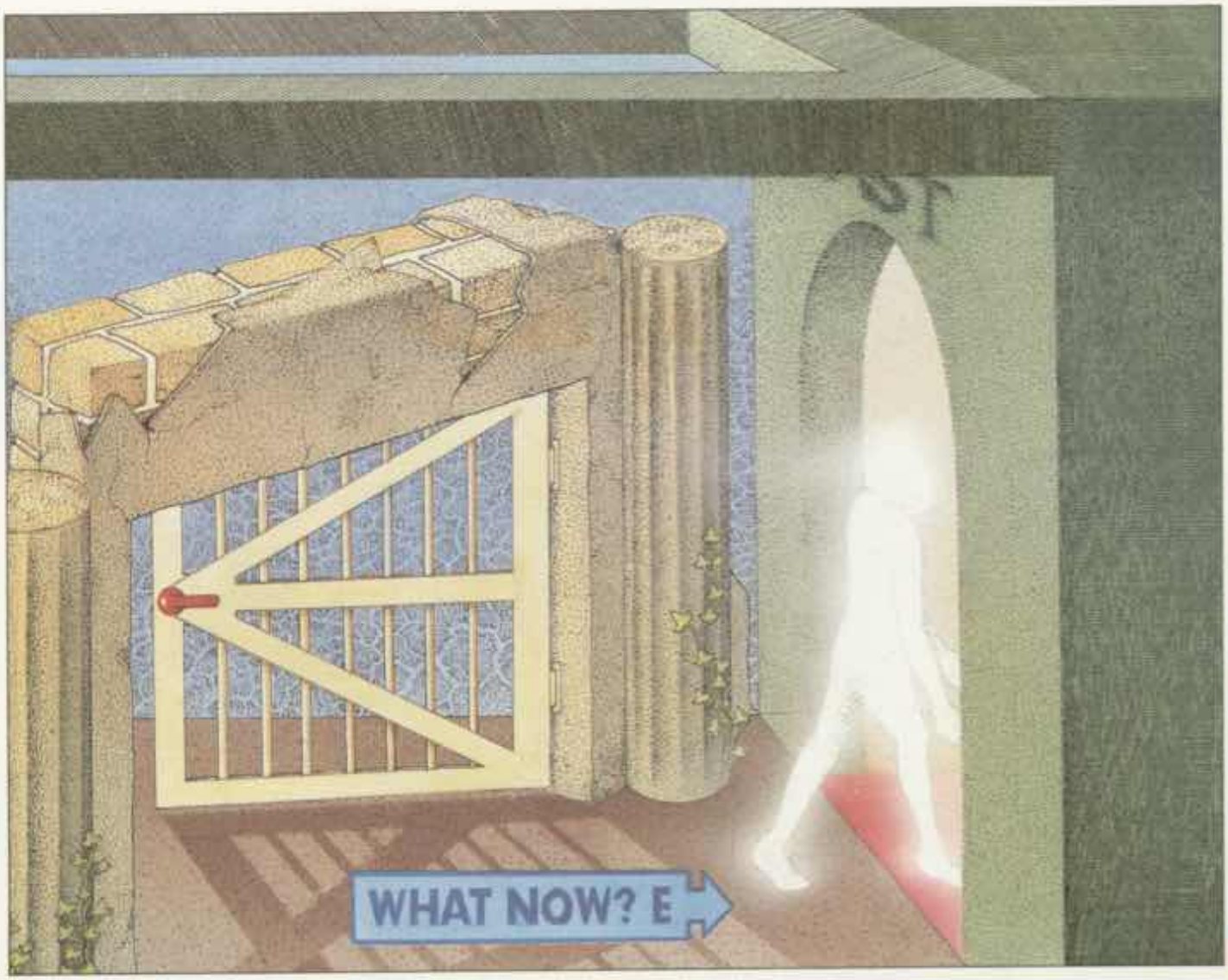
```
390  REM  **APONTAR DIRECOES**
400  IF L <  > 11 OR (LA = 1 AN
D OB(6) =  - 1 THEN  PRINT : PR
INT "PODE SEGUIR ":: GOTO 410
405  GOTO 460
410  IF N > 0 THEN  PRINT  TAB(
 11);"NORTE"
420  IF E > 0 THEN  PRINT  TAB(
 11);"LESTE"
430  IF S > 0 THEN  PRINT  TAB(
 11);"SUL"
440  IF W > 0 THEN  PRINT  TAB(
 11);"OESTE"
```

A rotina apresentada simplesmente verifica os valores das variáveis **N, S, E,** e **W** que você definiu, baseado em seu mapa de localidades. Se o valor da variável for maior do que zero, a direção escolhida é uma direção possível — ou seja, é uma saída. Essa rotina pode ser utilizada, sem modificações, em qualquer aventura que empregue um setoramento desse tipo.

INSTRUÇÕES

Agora que o jogador conhece as direções possíveis, podemos perguntar-lhe: "**PARA ONDE?**". A rotina responsável por esta pergunta é apresentada abaixo.

```
450 REM **INSTRUCOES**
460 INPUT  INVERSE 1;"PARA OND
E "; LINE I$
470 GOSUB 3010
515 GOTO G(I,4)
```



```
450 REM **INSTRUCOES**
460 PRINT:INPUT"PARA ONDE";IS
470 GOSUB 3010
```

A resposta dada pelo jogador é armazenada na variável **IS**. O programa verificará o tipo de resposta e atuará de acordo.

A próxima linha — linha 470 — desvia o fluxo do programa para uma sub-rotina localizada na linha 3010. Esta sub-rotina é responsável pela validação da resposta do jogador.

```
 600 REM **ROTINA INSTR.**
 610 LET IN=0: IF LEN YS>LEN XS
  THEN  RETURN
 620 FOR K=1 TO (LEN XS-LEN YS+
 1)
 630 IF YS=XS(K TO K+LEN YS-1)
 THEN  LET IN=K: GOTO 650
 640 NEXT K
 650 RETURN
3000 REM **ROTINA VERIFICACAO**
3010 LET NS="": LET XS=IS: LET
YS=" ": GOSUB 600: LET I=IN
3020 IF I=0 THEN  LET VS=IS: GO
TO 3050
3030 LET VS=IS( TO I-1)
3040 LET NS=IS(I+1 TO )
3050 LET I=0
3060 FOR K=1 TO 19
3070 IF VS=RS(K, TO LEN VS) THE
N  LET I=R(K): LET IS=VS( TO 1)
3080 NEXT K
3090 RETURN
```

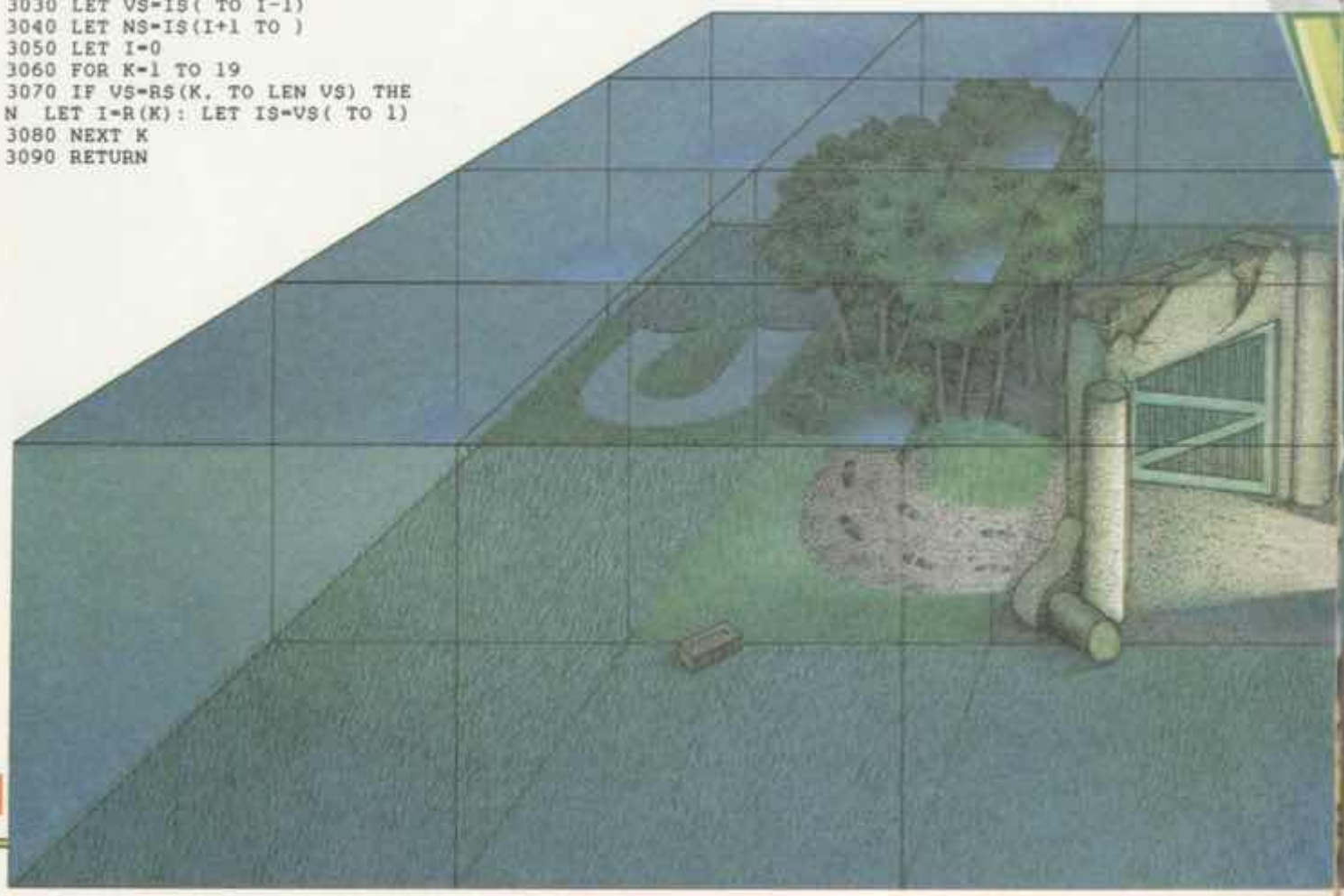
```
3000 REM **INSTRUCAO DE CHECAGE
M**
3010 NS="":I=INSTR(IS," ")
3020 IF I=0 THEN VS=IS:GOTO 305
0
3030 VS=LEFTS(IS,I-1)
3040 NS=MIDS(IS,I+1)
3050 I=0
3060 FOR K=1 TO 19
3070 IF INSTR(RS(K),VS)=1 THEN
I=R(K):IS=LEFTS(VS,1)
3080 NEXT
3090 RETURN
```

```
3000  REM  **ROTINA DE VERIFICA
CAO**
3010 NS = "": FOR Z = 1 TO  LEN
 (IS): IF  MIDS (IS,Z,1) = " "
THEN I = Z: GOTO 3020
3015  NEXT :I = 0
3020  IF I = 0 THEN VS = IS: GO
TO 3050
3030 VS =  LEFTS (IS,I - 1)
3040 NS =  MIDS (IS,I + 1)
3050 I = 0
3060  FOR K = 1 TO 19
3070  IF VS =  LEFTS (RS(K), LE
N (VS)) THEN I = R(K):IS =  LEF
TS (VS,1)
3080  NEXT
3090  RETURN
```

A rotina verifica se a variável **IS** compõe-se de duas palavras. Em caso afirmativo, a primeira palavra é chamada de **VS** e a segunda de **NS**. A variável **VS** contém um verbo — como **PEGAR, MATAR, LEVAR** — e todas as palavras indicativas de direção: **NORTE, SUL, LESTE** e **OESTE**. A variável **NS** armazena os nomes dos objetos que fazem parte da aventura.

Os computadores TRS-Color e MSX usam o comando **INSTR**, na linha 3010, para verificar se há algum espaço na resposta que foi armazenada em **IS**. Este espaço será indicativo da existência de duas palavras: a que pertence a **VS** e a que pertence a **NS**. No Spectrum e no Apple não existe o comando **INSTR**. Para substituí-lo, usamos uma pequena sub-rotina, colocada nas linhas 600 a 650.

Se um espaço for encontrado, a instrução na linha 3030 separa **IS** em suas componentes **VS** e **NS**. Se não houver espaço, a linha 3020 considera **VS** igual a **IS**.



A parte final da sub-rotina, composta pelas linhas 3060 a 3080, compara as respostas dadas com o conteúdo da matriz **RS**. Como sabemos, a matriz **RS** contém as palavras indicativas das direções que podem ser seguidas. Depois você verá como fazer para expandir o conteúdo da matriz **RS**. Pela instrução da linha 3070, o programa verifica se há igualdade entre os conteúdos **RS** e **VS**. Se houver, a variável **I** assume o valor da variável **R**. O programa reconhecerá a igualdade dos conteúdos verificando se o valor de **I** é maior que zero. A última parte da linha 3070 retira a primeira letra de **VS** e armazena-a na variável **IS**. A variável **IS** será utilizada depois, para fazer com que o jogador se movimente.

As sub-rotinas apresentadas adaptam-se a qualquer aventura, sem necessidade de grandes modificações. Apenas um detalhe pode precisar de alguns ajustes: a duração do comando **FOR...NEXT**, na linha 3060.

MOVIMENTOS

O passo seguinte consiste em adicionar uma rotina destinada a manipular a variável **L**, indicativa de localização, de acordo com o valor assumido pela variável **IS**. Esta rotina é apresentada em seguida:

```
1000 REM **ROTINA MOVIMENTO**
1010 IF IS="N" AND N>0 THEN  LE
T L=L-6: GOTO 310
1020 IF IS="L" AND E>0 THEN  LE
T L=L+1: GOTO 310
1030 IF IS="S" AND S>0 THEN  LE
T L=L+6: GOTO 310
1040 IF IS="O" AND W>0 THEN  LE
T L=L-1: GOTO 310
1050 REM **SE NAO HA LOCAL POSS
IVEL NESSA DIRECAO**
1060 PRINT '"DESCULPE, VOCE NAO
 PODE SEGUIR  NESTA DIRECAO.":
GOTO 330
```

```
1000 REM **ROTINA DE MOVIMENTO*
*
1010 IF IS="N" AND N>0 THEN L=L
-6:GOTO 310
1020 IF IS="L" AND E>0 THEN L=L
+1:GOTO 310
1030 IF IS="S" AND S>0 THEN L=L
+6:GOTO 310
1040 IF IS="O" AND W>0 THEN L=L
-1:GOTO 310
1050 REM **SE NAO HOUVER LOCAL
POSSIVEL NESSA DIRECAO**
1060 PRINT:PRINT"DESCULPE - VOC
E NAO PODE SEGUIR POR ESTE CAMI
NHO.":GOTO 330
```

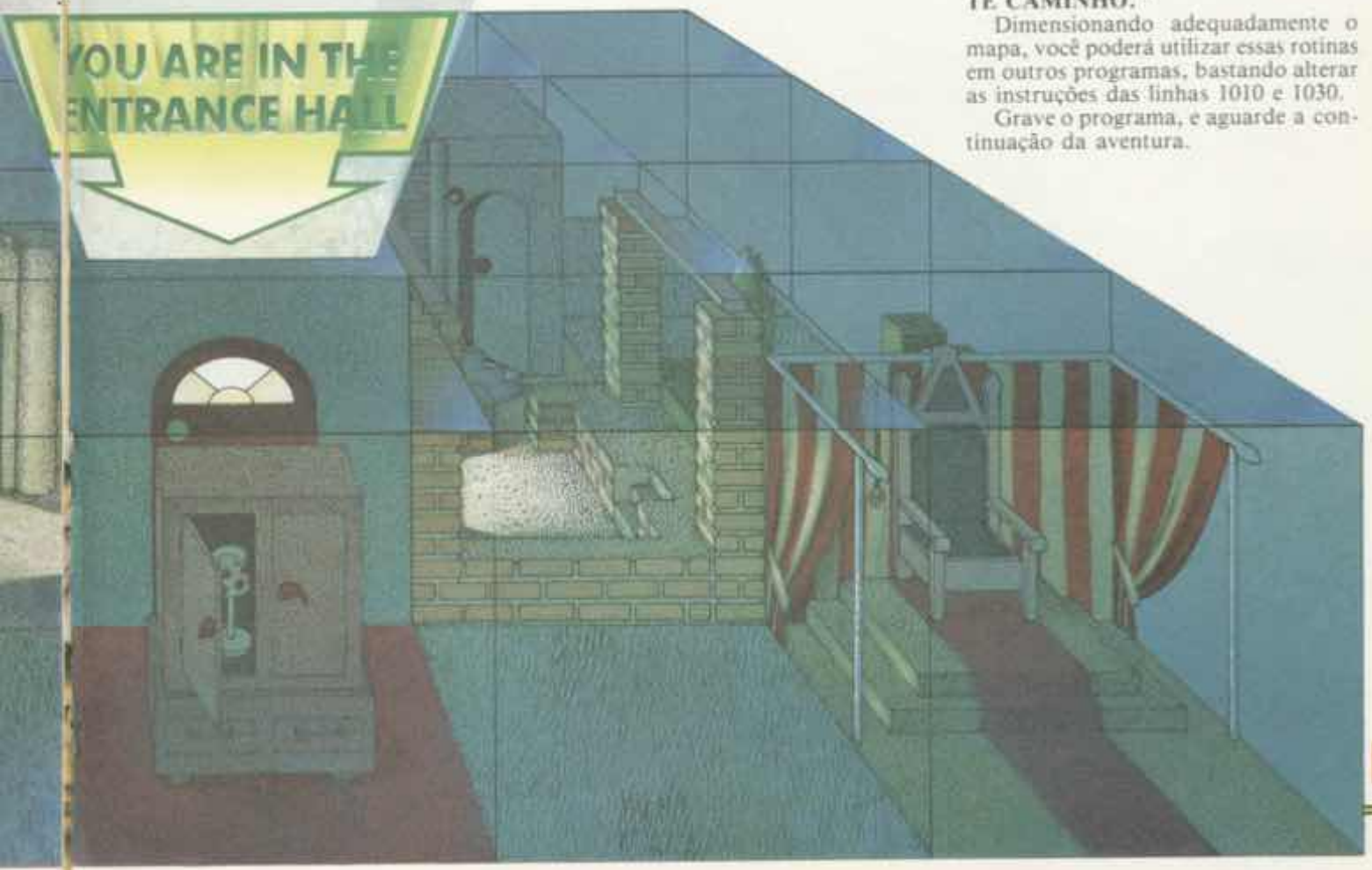
Como você está lembrado, o ponto de partida da aventura foi um mapa com uma largura de seis posições. Mover o jogador por essas posições significa alterar o valor da variável **L** por um fator baseado no tamanho do mapa. Por exemplo, para fazer com que o jogador se movimente nas direções Norte ou Sul basta adicionar ou subtrair seis da variável **L**. Isso fará com que o jogador suba ou desça uma linha completa no mapa. De modo semelhante, mover o jogador nas direções Leste ou Oeste significa adicionar ou subtrair 1 de **L**.

As instruções das linhas 1010 a 1040 verificam o conteúdo da variável **IS** e ajustam o valor de **L**. As saídas possíveis são definidas nas linhas que seguem as descrições dos locais.

Se não há uma saída na direção que o jogador escolheu, a linha 1060 apresenta a mensagem: "**DESCULPE — VOCÊ NÃO PODE SEGUIR POR ESTE CAMINHO.**"

Dimensionando adequadamente o mapa, você poderá utilizar essas rotinas em outros programas, bastando alterar as instruções das linhas 1010 e 1030.

Grave o programa, e aguarde a continuação da aventura.



OS OBJETOS DA AVENTURA

Chegou a hora de preencher o mundo ainda vazio de nossa aventura. Você verá aqui como acrescentar ao programa uma lista de objetos e, também, como manipulá-los.



Na última seção de Programação de Jogos, dispúnhamos de um conjunto completo de posições para nossa aventura e o jogador já estava capacitado a explorar todas elas. As movimentações do aventureiro, porém, até agora não têm propósito, uma vez que o mundo da aventura está vazio. Convém, assim, retomar o planejamento inicial e examinar o que se pretendia incluir em cada ponto.

Veremos, em seguida, como adicionar ao programa as rotinas que colocarão os objetos envolvidos na aventura em seus devidos lugares. Outras rotinas permitirão ao jogador carregar objetos consigo ou deixá-los de lado. Escreveremos, ainda, uma rotina para listar um inventário de todos os itens utilizados — ela será importante para o jogador, quando ele se deparar com um problema e precisar verificar exatamente a situação de cada objeto.

Use o **LOAD** e carregue o programa usado recentemente, deixando-o pronto para receber as novas rotinas.

OBJETOS

A máquina precisa saber três coisas sobre cada objeto da aventura: o número de posição onde foi inicialmente colocado, seu nome (ou descrição curta) e, finalmente, a descrição detalhada do local onde se encontra. As três informações são necessárias já que, primeiro, o computador deverá escolher um objeto adequado a cada posição. Depois, precisará contar ao aventureiro que objetos estão nesta posição — mesmo que use uma longa descrição. E, por último, terá que dispor de um nome para usar em instruções e na lista.

Os números de posição serão colocados em uma matriz, o nome do objeto em outra, e a descrição longa em uma terceira. As três matrizes serão manipuladas em paralelo pelo programa — cada elemento da matriz guardará um espaço equivalente de informação sobre o objeto: o primeiro conterá o número da posição, o segundo o nome do objeto e assim por diante. Adicione estas linhas ao seu programa:

- LINHAS **DATA** PARA A LISTA DE OBJETOS
- DESCRIÇÕES CURTAS E LONGAS
- COLOCAÇÃO DOS OBJETOS NAS POSIÇÕES CORRETAS
- MAIS VERBOS
- COMO PEGAR E LARGAR OBJETOS
- COMO MOSTRAR AO JOGADOR A LISTA DOS OBJETOS QUE CARREGA



5

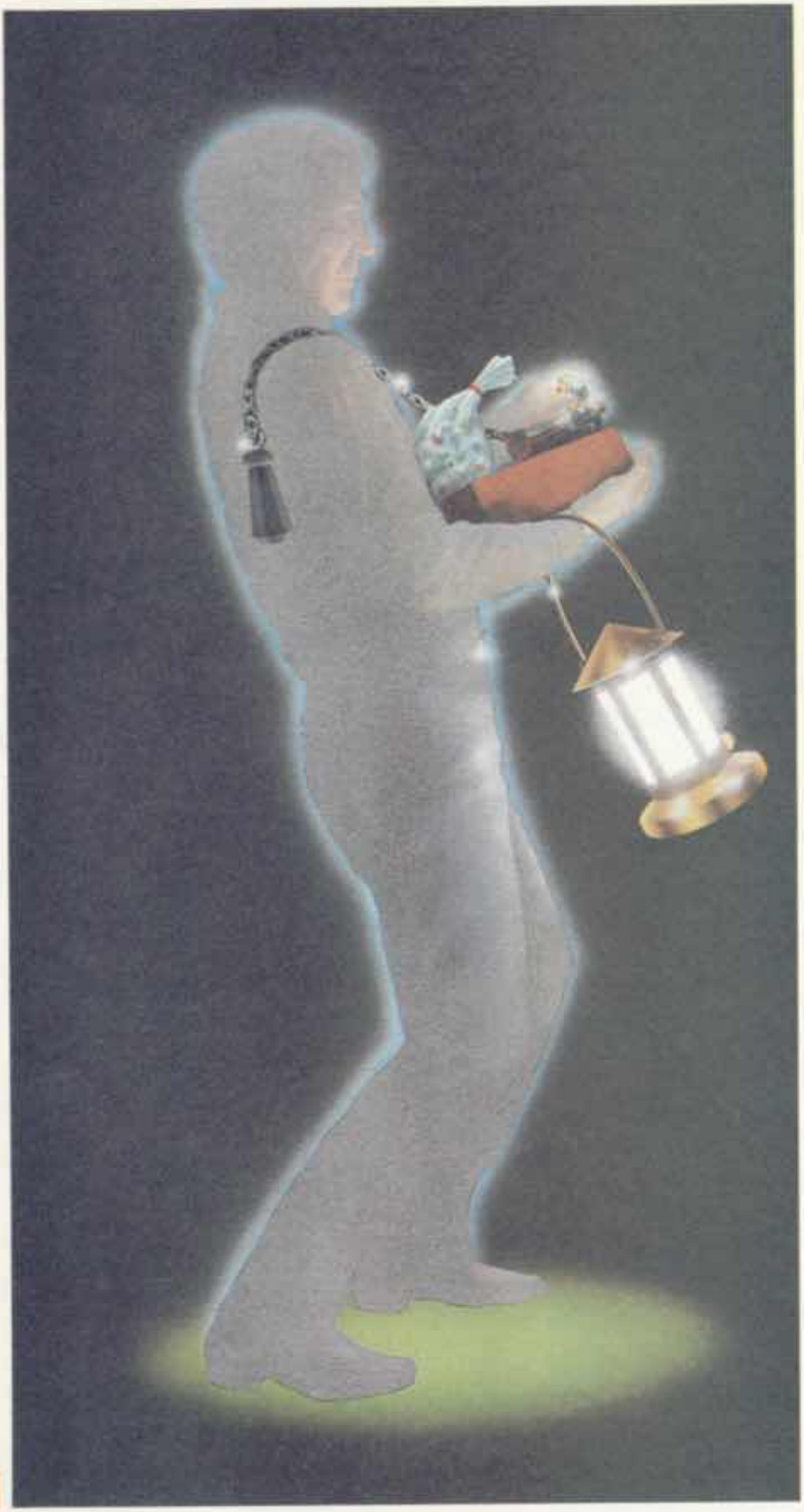
```
160 REM **PREPARA MATRIZES DE
OBJETOS**
170 READ NB
180 DIM B(NB): DIM BS(NB,14):
DIM SS(NB,40)
190 FOR I=1 TO NB: READ B(I),B
S(I),SS(I): NEXT I
200 DATA 7,4,"SACO","HA UM SAC
O DE BOLAS DE GUDE AQUI"
210 DATA 14,"TIJOLO","TEM UM T
IJOLO NO CHAO"
220 DATA 24,"CORRENTE","HA UMA
 CORRENTE PENDURADA SOBRE O TR
ONO"
230 DATA 0,"REVOLVER","TEM UM
REVOLVER NO CHAO"
240 DATA 0,"OLHO","UM OLHO CRA
VEJADO DE BRILHANTES ESTA NO C
HAO"
250 DATA 22,"LAMPADA","VOCE ES
TA DIANTE DE UMA LAMPADA"
260 DATA 0,"COLETOR","DE REPEN
TE SURGE UM COLETOR DE IMPOSTO
S"
```

TK

```
160 REM **PREPARA MATRIZES DE O
BJETOS**
170 READ NB
180 DIM OB(NB),OBS(NB),SIS(NB)
190 FOR I=1 TO NB:READ OB(I),OB
S(I),SIS(I):NEXT
200 DATA 7,4,SACO,HA UM SACO DE
 BOLAS DE GUDE AQUI
210 DATA 14,TIJOLO,TEM UM TIJOL
O NO CHAO
220 DATA 24,CORRENTE,HA UMA COR
RENTE PENDURADA SOBRE O TRONO
230 DATA 0,REVOLVER,TEM UM REVO
LVER NO CHAO
240 DATA 0,OLHO,UM OLHO CRAVEJA
DO DE BRILHANTES ESTA NO CHAO
250 DATA 22,LAMPADA,VOCE ESTA D
IANTE DE UMA LAMPADA
260 DATA 0,COLETOR,DE REPENTE S
URGE UM COLETOR DE IMPOSTOS
```

Cada linha entre 200 e 260 contém três partes dos dados referentes ao mesmo objeto. A linha 200 apresenta um dado a mais em sua lista. O número 7 — o primeiro do comando **DATA** — diz à máquina quantos conjuntos de dados existem.

Uma vez que o número 7 tenha sido



lido pela linha 170, três matrizes serão dimensionadas para este tamanho, pela linha 180. **OB** conterá a posição de cada objeto — um número da posição, ou 0, se o elemento ainda não existe (é como se fosse a tampa de um baú, que precisasse ser aberto a cada aventura), e −1, se está sendo levado pelo aventureiro. **OBS** conterá a descrição curta e **SIS** a descrição longa.

A linha 190 completará a matriz com os dados das linhas 200 a 260. Os comandos **DATA** estão arrumados em conjuntos de três elementos: um número de posição, a descrição curta do objeto e a descrição longa do objeto.

Para usar a mesma rotina em outras aventuras, não é necessário fazer muitas alterações nessa estrutura, pois, ajustando a primeira parte dos dados, automaticamente os comandos **FOR... NEXT** e as dimensões das matrizes estarão ajustados.

COMO POSICIONAR OS OBJETOS

O programa contém agora todas as informações sobre os objetos e as posições onde deverão ser colocados. A rotina seguinte exibe a descrição longa do objeto quando o jogador está no local adequado.

```
360 REM **COLOCA CADA OBJETO E
M SEU LUGAR**
370 FOR I=1 TO NB: IF B(I)=L
THEN  PRINT S$(I)
380 NEXT I
```

```
360 REM**COLOCA CADA OBJETO EM
SEU LUGAR**
370 FOR  I=1 TO NB:IF OB(I)=L T
HEN PRINT SIS(I)
380 NEXT
```

Neste estágio, faça uma pequena alteração nas linhas 330 e 340: troque **GOTO 400** para **GOTO 370**. As linhas 370 e 380 checam a matriz que guarda a posição do objeto. Se algum número de posição combinar com a posição corrente — **L** — uma descrição curta será exibida após a descrição da posição. Essa rotina pode ser usada em outras aventuras, sem alterações.

MAIS VERBOS

A aventura inclui objetos em diferentes posições mas, como a máquina até aqui não entende qualquer palavra ex-

ceto NORTE, SUL, LESTE e OESTE, o pobre aventureiro não pode fazer nada com eles. Imagine sua frustração ao se ver incapaz de pegar o tão desejado saquinho de bolas de gude, ou sem condições de se defender do fiscal da Receita! Precisamos fornecer ao computador um vocabulário de palavras que ele possa reconhecer, informando o que fazer com os objetos. Mais tarde, veremos como proceder se o jogador entrar uma palavra que não está no vocabulário programado.

Como o programa trata todas as palavras de direção como verbos, o melhor lugar para os verbos, que descrevem o que fazer com os objetos, é a matriz **R$** e, para os números correspondentes, **R**.

Precisaremos, no entanto, fazer algumas alterações na linha 130. Os limites do **FOR...NEXT** deverão ser mudados. Reescreva toda a linha ou use o editor da máquina para mudá-la. Seja qual for sua escolha, a linha 130 será agora:

```
130 FOR K=1 TO 19: READ R$(K),
R(K): NEXT K
```

```
130 FOR K=1 TO 19:READ R$(K),R(
K):NEXT
```

Agora, adicione as linhas 140 e 145:

```
140 DATA "NADAR","5","ESVAZIAR
","6","ACENDER","7","DESISTIR"
,"8","LISTAR","9","MATAR","10"
,"ATIRAR","10","AJUDAR","11"
145 DATA "PEGAR","2","APANHAR"
,"2","CARREGAR","2","COLOCAR",
"3","DEIXAR","3","LARGAR","3",
"PUXAR","4"
```

```
140 DATA NADAR,5,ESVAZIAR,6,ACE
NDER,7,DESISTIR,8,LISTAR,9,MATA
R,10,ATIRAR,10,AJUDAR,11
145 DATA PEGAR,2,APANHAR,2,CARR
EGAR,2,COLOCAR,3,DEIXAR,3,LARGA
R,3,PUXAR,4
```

A cada verbo corresponde um número. Verbos com o mesmo número possuem o mesmo significado e, portanto, o mesmo efeito. Planejamos o programa de modo que o computador reconheça, por exemplo, **PEGAR**, **APANHAR** e **CARREGAR**, evitando que o aventureiro gaste seu precioso tempo tentando descobrir qual desses verbos usar. Você pode facilmente adicionar suas próprias palavras às linhas de comando **DATA**: basta trocar o laço **FOR...NEXT** na linha 130 e colocar outro comando **DATA** após a linha 145. Você deverá fazer algumas alterações em outros locais do programa mas, nas próximas seções de Programação de Jogos, informaremos exatamente como proceder.

SELEÇÃO DAS ROTINAS ADEQUADAS

Depois de entrar todos os verbos na última rotina, o computador precisará de outras rotinas que o tornem capaz de agir conforme as instruções e de fazer com que o aventureiro carregue alguns objetos.

A sub-rotina que começa na linha 3010, por exemplo, define **V$**, **N$** e **I** (um número da matriz **R** que você já deve ter escrito).

Esta pequena rotina fará com que a máquina seja capaz de selecionar a rotina correta, de acordo com o valor de **I** — que é o significado da entrada do aventureiro.

O Spectrum não tem o comando **ON...GOTO**, usado em programas para outros computadores. As linhas do programa, portanto, têm que ser um pouco diferentes.

Já temos uma matriz, **G**, que contém números de linha para as descrições de posição. Os números de linha necessários às novas rotinas podem ser adicionados a esta matriz.

Por isso, a linha 30 ficou assim:

```
30 FOR N=1 TO 4: FOR M=1 TO
11: READ G(M,N): NEXT M: NEXT
 N
```

E é também pelo motivo acima que essa linha contém todos os números de linha que precisaremos (veja a explicação completa no artigo da página 270).

```
70 DATA 1010,1150,1240,1310,
1410,1460,1500,1360,1080,1550
,3110
```

Agora, adicione a rotina que solucionará a rotina correta, de acordo com o valor de **I**:

```
500 REM **SELECIONA OPCAO**
510 IF I=0 THEN  GOTO 520
520 PRINT '"EU NAO SEI COMO ";
V$: GOTO 370
```

Se a sub-rotina de verificação de instrução que começa na linha 3010 não encontrar uma combinação para **V$** em **R$**, **I** torna-se zero e a linha 510 faz com que a mensagem "EU NÃO SEI COMO" surja na tela. Se **I** tiver qualquer outro valor, a linha 515 encontra o número de linha correto na matriz **G** e executa um **GOTO**.

```
500 REM**SELECIONA OPCAO**
505 IF I=0 THEN GOTO 520
510 ON I GOTO 1010,1150,1240,13
10,1410,1460,1500,1360,1080,155
0,3110
520 PRINT:PRINT "EU NAO SEI COM
O ";V$:GOTO 370
```

Os números após o **ON...GOTO** na linha 510 iniciam as diversas rotinas. Cada valor de **I** é um verbo diferente ou um grupo de verbos. Se **I = 10**, por exemplo, a rotina "MATAR" será selecionada — e, sendo o décimo número na linha, ela começará na linha 1550.

Se a sub-rotina de verificação de instrução que começa na linha 3010 não encontrar uma combinação de **V$** em **R$**, **I** torna-se 0. Nesse caso, o **ON...GOTO** na linha 510 não terá nenhum efeito. A mensagem na linha 520 será exibida na tela.

COMO PEGAR OS OBJETOS

Já temos uma rotina para quando **I** for igual a 1, o que ocorre quando o aventureiro dá uma ordem. Ela está entre as linhas 1010 e 1060.

Quando **I = 2**, o aventureiro digitou uma rotina "PEGAR" — ou seja, as palavras **PEGAR**, **APANHAR** ou **CARREGAR**. Esta rotina, apresentada a seguir, permitirá ao aventureiro pegar e conservar consigo qualquer objeto que esteja na posição.

```
1140 REM **PEGAR**
1150 FOR G=1 TO NB
1160 IF N$=B$(G, TO LEN N$) THE
N  GOTO 1190
1170 NEXT G
1180 PRINT N$;"???": GOTO 330
1190 IF B(G)=-1 THEN  PRINT "VO
CE PEGOU": GOTO 330
1200 IF B(G)<>L THEN  PRINT "NA
O ESTA AQUI": GOTO 330
1210 PRINT "OK": LET B(G)=-1
1220 GOTO 330
```

```
1140 REM**PEGAR**
1150 FOR G=1 TO NB
1160 IF INSTR(OBS(G),NS)=1 THEN
 GOTO 1190
```

```
1170 NEXT
1180 PRINT NS;"???":GOTO 330
1190 IF OB(G)=-1 THEN PRINT "VO
CE PEGOU":GOTO 330
1200 IF OB(G)<>L THEN PRINT "NA
O ESTA AQUI":GOTO 330
1210 PRINT "OK":OB(G)=-1
1220 GOTO 330
```



```
1140  REM ** PEGAR **
1150  FOR G = 1 TO NB
1160  IF NS =  LEFT$ (OBS(G), L
EN (NS)) THEN 1190
1170  NEXT
1180  PRINT NS;" ??"; GOTO 330
1190  IF OB(G) =  - 1 THEN  PRI
NT "VOCE PEGOU": GOTO 330
1200  IF OB(G) <  > L THEN  PRI
NT "NAO ESTA AQUI": GOTO 330
1210  PRINT "OK":OB(G) =  - 1
1220  GOTO 330
```

As linhas 1150 a 1170 procuram a matriz contendo a descrição curta — **BS**, no caso do Spectrum, e **OBS** nos demais computadores — do objeto que o aventureiro chamou. Se este é encontrado, o programa pula para a linha 1190. Caso contrário, a linha 1180 exibe o nome do objeto que o aventureiro digitou, seguido de pontos de interrogação.

Depois de encontrar o nome do objeto, duas verificações são feitas. A linha 1190 checa o elemento da matriz de posição do objeto — **B** ou **OB** —, para saber se ele já foi levado. Em caso afirmativo (o valor do elemento da matriz é −1), a mensagem "VOCÊ PEGOU" será exibida.

A linha 1200 verifica se o objeto está presente, examinando novamente a posição da matriz. Se ele não estiver presente, o programa exibirá a mensagem "NÃO ESTA AQUI". Você poderá trocar essa mensagem, se ela não servir para a sua aventura.

Caso o objeto não tenha sido levado e se encontre na mesma posição que o aventureiro, a linha 1210 exibirá "OK" e o elemento na matriz de posição de objetos será mudado para −1.

COMO DEIXAR OBJETOS DE LADO

A rotina "DEIXAR" faz o contrário. Ela permite que o jogador abandone qualquer dos objetos que pegou.



```
1230 REM **DEIXAR**
1240 FOR G=1 TO NB
1250 IF NS=BS(G, TO LEN NS) THE
N  GOTO 1270
```

```
1260 NEXT G: PRINT NS;"???": GO
TO 330
1270 IF B(G)<>-1 THEN  PRINT "V
OCE NAO PODE LARGAR O QUE NAO T
EM": GOTO 330
1280 PRINT "OK": LET B(G)=L
1290 GOTO 330
```



```
1230 REM**DEIXAR**
1240 FOR G=1 TO NB
1250 IF INSTR(OBS(G),NS)=1 THEN
 1270
1260 NEXT:PRINT NS;"???":GOTO 3
30
1270 IF OB(G)<>-1 THEN PRINT "V
OCE NAO PODE LARGAR O QUE NAO T
EM":GOTO 330
1280 PRINT "OK":OB(G)=L
1290 GOTO 330
```



```
1230  REM ** DEIXAR **
1240  FOR G = 1 TO NB
1250  IF NS =  LEFT$ (OBS(G), L
EN (NS)) THEN 1270
1260  NEXT : PRINT NS;" ??": GO
TO 330
1270  IF OB(G) <  >  - 1 THEN
PRINT "VOCE NAO PODE LARGAR O Q
UE NAO TEM": GOTO 330
1280  PRINT "OK":OB(G) = L
1290  GOTO 330
```

Seu funcionamento se assemelha muito ao da rotina "PEGAR", vista anteriormente. A matriz de descrição curta é mais uma vez procurada — agora nas linhas 1240 a 1260. Se o objeto pedido estiver na matriz, a linha 1270 verifica se está sendo levado pelo aventureiro. Se não estiver, o computador exibirá a mensagem "VOCÊ NÃO PODE LARGAR O QUE NÃO TEM".

Se o aventureiro estiver carregando o objeto, a linha 1280 exibe "OK" e o elemento apropriado na matriz de posição de objetos — **OB** ou **B** — é ajustado. Ele toma, então, o número da posição corrente — L —, já que −1 significa que o objeto estava sendo carregado.

A LISTAGEM DO SAQUE

O aventureiro distraído ficará muito grato ao obter uma lista de todos os objetos que carrega.

Aqui está uma rotina que faz exatamente isso:



```
1070 REM **LISTAR**
1080 PRINT "VOCE TEM ";: LET IN
=0
```

```
1090 FOR G=1 TO NB
1100 IF B(G)=-1 THEN  PRINT TAB
 10;BS(G): LET IN=IN+1
1110 NEXT G
1120 IF IN=0 THEN  PRINT "NADA"
1130 GOTO 330
```



```
1070 REM**LISTAR**
1080 PRINT "VOCE TEM ";:IN=0
1090 FOR G=1 TO NB
1100 IF OB(G)=-1 THEN PRINT TAB
(10)OBS(G):IN=IN+1
1110 NEXT
1120 IF IN=0 THEN PRINT "NADA"
1130 GOTO 330
```



```
1070  REM ** LISTAR **
1080  PRINT "VOCE TEM ";:IN = 0
1090  FOR G = 1 TO NB
1100  IF OB(G) =  - 1 THEN  PRI
NT  TAB( 10);OBS(G):IN = IN + 1
1110  NEXT
1120  IF IN = 0 THEN  PRINT "NA
DA"
1130  GOTO 330
```

"VOCÊ TEM" será exibido pela linha 1080, antes que se inicie a listagem dos objetos. O laço **FOR...NEXT** checa, um a um, os elementos da matriz de posições de objetos. Dessa vez, os elementos importantes são os que contêm −1, indicando que o objeto está sendo carregado. Se o valor de qualquer um dos elementos for −1, a descrição curta do objeto será, então, exibida. O contador do inventário I será incrementado de 1.

Se nenhum objeto estiver sendo carregado, o contador **IN** continua em zero e a linha 1120 exibe "NADA", em vez da lista de objetos.

As rotinas "PEGAR", "DEIXAR" e "LISTAR" podem ser usadas como estão, se **NB** for definido em uma rotina anterior.

Agora, o programa está pronto para receber as rotinas finais, que serão apresentadas num próximo artigo. Elas se referem ao fiscal da Receita, ao tijolo, à lâmpada, à procura do globo ocular, ao término da aventura e, finalmente, à instrução descrevendo o objeto da busca.

Se você fizer uma experiência, executando o programa neste estágio, verá que, enquanto parte dele funciona, alguma coisa estranha acontece. A razão para isso é que o programa requer um número de rotinas que ainda não existe. Se você entrar certas palavras, o programa tentará desviá-las para linhas inexistentes.

A CONCLUSÃO DA AVENTURA

- COMO SOCORRER O HERÓI
- ELIMINE O COLETOR DE IMPOSTOS
- COMO ACENDER A LÂMPADA
- INSTRUÇÕES



Na busca incessante do olho perdido do totem inca, o herói chega à sala do trono de um reino remoto. Sairá ele com vida ou perecerá nas garras de um implacável coletor de impostos?

Depois de muitas peripécias e lances de emoção, aproxima-se do fim nosso jogo de aventura. Antes de concluí-lo, porém, devemos acrescentar-lhe certos detalhes, como novos perigos e avisos, além de uma saída pela qual o aventureiro possa escapar com vida. Algumas instruções também precisam ser incorporadas ao jogo.

Como essas rotinas contêm detalhes específicos a esta aventura, o programa que se segue não deve ser usado em outras aventuras sem modificações. Ele apenas completará o jogo em andamento e mostrará, em termos gerais, o que é necessário para fazer um programa desse tipo. No próximo artigo, mostraremos como os princípios apresentados aqui podem ser adaptados de modo a se encaixar às suas próprias idéias.

VOCÊ PRECISA DE AJUDA

Sempre que se encontrar diante de uma situação particularmente perigosa, o jogador (e, portanto, o personagem da aventura) pode recorrer ao computador, solicitando sugestões. Estas aparecerão na forma de mensagens impressas pela máquina em resposta ao pedido ("**AJUDAR**") do jogador. O significado de tais mensagens, assim como o momento em que elas serão impressas, depende apenas do programador. Podemos escolher, se desejarmos, não apresentar qualquer mensagem, ou fazê-las propositalmente ilusórias, ou mesmo prestar ajuda somente em alguns locais isolados.

Em nossa aventura existem vários pontos onde talvez valha a pena digitar

curtas mensagens para o jogador, advertindo-o, por exemplo, sobre o quarto escuro quando ele se encontrar em suas proximidades. Uma forma interessante de mensagem consiste em responder ao pedido de "**AJUDAR**" com um "OLHE PARA ONDE VAI", ou algo até mais enigmático.

Outra mensagem pode ser emitida às margens do rio, onde o aventureiro corre o risco de se afogar (principalmente se estiver carregando um tijolo).

Evidentemente, o número dessas mensagens pode ser muito grande. Tudo depende da vontade do programador. Suponhamos, contudo, que este queira emitir apenas um aviso, por exemplo, no rio. O primeiro passo para isso consistirá em relacionar o aviso com o número do local — número 7 — e com a variável que registra a presença do tijolo, **OB (7)**.

```
3100 REM **AJUDAR**
3110 IF L<>7 OR B(2)<>-1 THEN
PRINT "DESCULPE, NAO POSSO AJUD
AR AGORA": GOTO 330
3120 PRINT "TIJOLOS SAO MUITO P
ESADOS. SEU  BRACO DEVE ESTAR D
OENDO.": GOTO 330
```

```
3100 REM **AJUDAR**
3110 IF L<>7 OR OB(2)<>-1 THEN
PRINT "DESCULPE, NAO POSSO AJUD
AR AGORA":GOTO 330
3120 PRINT "TIJOLOS SAO MUITO P
ESADOS. SEU BRACO DEVE ESTAR DO
ENDO.":GOTO 330
```

Se o aventureiro não se encontrar no rio (**L < > 7**) ou não estiver carregando o tijolo — (**OB (2) < > −1** ou **B (2) < > −1**) —, a linha 3110 imprimirá a mensagem "DESCULPE, NÃO POSSO AJUDAR AQUI". Se, pelo contrário, ele estiver transportando o tijolo e solicitar ajuda ao chegar ao rio, então aparecerá o aviso "TIJOLOS SÃO MUITO PESADOS. SEU BRAÇO DEVE ESTAR DOENDO", impresso pela linha 3120.

Nada nos impede de acrescentar uma lista completa de mensagens de ajuda nas mais diversas condições e locais de aventura.

O COLETOR DE IMPOSTOS

Se o jogador é o "mocinho" da aventura, quem faz o papel de "bandido" é o coletor de impostos. O objetivo deste último é confiscar alguns objetos para saldar a dívida do nosso herói com o fisco. Assim, ele escolhe ao acaso um dos objetos transportados pelo jogador e o confisca, podendo levar até mesmo um tijolo como pagamento.

Se o aventureiro não estiver carregando alguma coisa quando o coletor de impostos aparecer, ele será preso e seu destino será apodrecer em uma masmorra. E assim o jogo terminará.

O papel do coletor de impostos é dar um toque de suspense ao jogo, já que ele surge de forma inesperada, independentemente do local ou de outras condições. Como em outros exemplos de acaso na programação de jogos, devemos fazer uso da função **RND**. O coletor é tratado como qualquer um dos objetos, com a diferença que sua localização é determinada ao acaso.

Aqui estão as linhas extras, para fazer o coletor de impostos aparecer.

```
320 IF  RND<(1/15)  AND  TA=0
THEN  LET B(7)=L: LET TA=1
480 IF B(7)=L AND I<>10 THEN
GOTO 1590
```

```
320 IF RND(15)=1 AND TA=0 THEN
OB(7)=L:TA=1
480 IF OB(7)=L AND I<>10 THEN 1
590
```

```
320 RN=RND(-TIME):IF INT(RND(1)
*15+1)=1 AND TA=0 THEN OB(7)=L:
TA=1
480 IF OB(7)=L AND I<>10 THEN 1
590
```

```
320  IF  INT ( RND (1) * 15 + 1
) = 1 AND TA = 0 THEN OB(7) = L
:TA = 1
480  IF OB(7) = L AND I <  > 10
 THEN 1590
```

Inicialmente, o Spectrum iguala todas essas variáveis a 0.

A linha 320 de todos os programas dá ao aventureiro uma chance em quinze de encontrar o coletor — este é o propósito do 15 no **RND**. O coletor só pode aparecer uma vez durante o jogo; portanto, você precisa da variável **TA** para saber se isso aconteceu ou não.

Se o número randômico for 1 (ou menor de que 1/15 no Spectrum) e o inspetor ainda não tiver aparecido, a linha 320 ajustará o valor do elemento 7 da matriz de localização de objetos — correspondente ao coletor —, de acordo com o lugar em que se encontra o jogador — **L**. O aviso de que o coletor chegou será dado como se ele fosse um objeto que tivesse sido colocado naquele local. Esse aviso nada mais é do que a descrição do objeto 7: o coletor.

A linha 480 tem a ver com o ato de eliminar o coletor, verificando se você tentou matá-lo. Se você não o fez, o programa pulará para a linha 1590.



COMO SE LIVRAR DO COLETOR

Quando o coletor de impostos mostrar sua cara feia, o aventureiro deverá



atirar nele com o revólver que pode ser encontrado no outro lado do rio:

```
1540 REM **MATAR**
1550 IF B(4)<>-1 THEN  PRINT "C
OM O QUE?": GOTO 320
1560 IF B(7)<>L THEN  PRINT VS;
" QUEM?": GOTO 320
1570 PRINT "VOCE MATOU O ";BS(7
): LET B(7)=0: GOTO 330
```

```
1540 REM **MATAR**
1550 IF OB(4)<>-1 THEN PRINT"CO
M O QUE ?":GOTO 320
1560 IF OB(7)<>L THEN PRINT VS;
" QUEM ?":GOTO 320
1570 PRINT " VOCE MATOU O ";OBS
(7):OB(7)=0:GOTO 330
```

Essa rotina será usada quando o jogador digitar "MATAR" ou "ATIRAR". Se ele estiver sem a arma (**OB (4) <> –1**) ou (**B (4) <> –1**), a linha 1550 perguntará "COM QUÊ?". Se o coletor não estiver e o jogador tentar matá-lo, a linha 1560 imprimirá "QUEM?". A linha 1570 comunica: "VOCÊ MATOU O COLETOR" e ajusta o elemento 7 da matriz de localização de objetos para o inspetor não mais existir.

A VINGANÇA DO COLETOR

Aqui, o aventureiro é quem sofre nas mãos do coletor de impostos:

```
1580 REM **COLETOR DE IMPOSTOS*
1590 LET IN=0: LET B(7)=0
1600 FOR K=1 TO NB
1610 IF B(K)=-1 THEN  LET IN=IN
+1
1620 NEXT K
1630 IF IN=0 THEN  PRINT "COMO
VOCE NAO TEM NADA QUE POSSASER
CONFISCADO, ELE O PRENDE EM UMA
 MASMORRA IMUNDA": GOTO 1360
1640 LET K=INT (RND*NB)+1: IF B
(K)<>-1 THEN  GOTO 1640
1650 PRINT "ELE TOMA O ";BS(K),
"DE VOCE": LET B(K)=0: GOTO 400
```

```
1580 REM **COLETOR DE IMPOSTOS*
1590 IN=0:OB(7)=0
1600 FOR K=1 TO NB
1610 IF OB(K)=-1 THEN IN=IN+1
1620 NEXT
1630 IF IN=0 THEN PRINT "COMO V
OCE NAO TEM NADA QUE POSSASER C
ONFISCADO, ELE O PRENDE  EMUMA
MASMORRA IMUNDA":GOTO 1360
1640 K=RND(NB):IF OB(K)<>-1 THE
N 1640
1650 PRINT "ELE TOMA O ";OBS(K)
," DE VOCE":OB(K)=0:GOTO 330
```

```
1580  REM  ** COLETOR DE IMPOST
OS **
1590 IN = 0:OB(7) = 0
1600  FOR K = 1 TO NB
1610  IF OB(K) =  - 1 THEN IN =
 IN + 1
1620  NEXT
1630  IF IN = 0 THEN  PRINT "CO
MO VOCE NAO TEM NADA QUE POSSA
SER    CONFISCADO, ELE O PRENDE
 EM UMA          MASMORRA IMUNDA
": GOTO 1360
1640 K =  INT ( RND (1) * NB +
1): IF OB(K) <  >  - 1 THEN 164
0
1650  PRINT "ELE TOMA O ";OBS(K
);" DE VOCE":OB(K) = 0: GOTO 33
0
```

Como ao inspetor de taxas é permitido aparecer apenas uma vez durante a aventura, a linha 1590 ajusta o elemento 7 da matriz de localização de objetos de forma que ele não mais exista enquanto durar o programa. Esse procedimento não afeta em nada a rotina, mas evita o aparecimento do coletor no futuro. **IN** é um marcador usado para verificar se objetos estão sendo carregados.

As linhas 1600 e 1620 passam pela matriz de localização de objetos, verificando se cada objeto está sendo transportado. Qualquer objeto que estiver

sendo carregado aumenta 1 no marcador **IN**.

Se nada estiver sendo transportado, então o valor de **IN** permanecerá 0 e o aventureiro será comunicado: "COMO VOCÊ NÃO TEM NADA QUE POSSA SER CONFISCADO, ELE O PRENDERÁ EM UMA MASMORRA IMUNDA". O jogo termina e pergunta-se ao aventureiro se ele deseja uma outra jogada; isto é feito pela linha 1360.

Caso haja objetos sendo transportados, a linha 1640 pegará um deles ao acaso. Se esse objeto estiver entre os que estão sendo carregados, ele será apreendido. No entanto, se não estiver, outro objeto será escolhido a esmo e assim por diante, até que a escolha coincida com um que esteja sendo transportado.

Se um objeto adequado tiver sido selecionado, a linha 1650 comunicará ao aventureiro que objeto foi confiscado pelo coletor. O elemento da matriz de localização de objetos correspondentes será então alterado; assim, o objeto não mais existirá.

UM POUCO DE NATAÇÃO

Esta rotina é usada quando o jogador decide atravessar o rio:

```
1400 REM **NADAR**
1410 IF L<>7 THEN  PRINT "NADAR
 ONDE ?!!": GOTO 400
1420 IF B(2)=-1 THEN  PRINT "QU
E VERGONHA, VOCE SE AFOGOU !":
GOTO 1360
1430 IF B(4)>-1 THEN  PRINT "VO
CE ACHOU UM REVOLVER": LET B(4)
=-1: GOTO 400
1440 PRINT "VOCE SE MOLHOU TODO
": GOTO 400
```

```
1400 REM **NADAR**
1410 IF L<>7 THEN PRINT " ONDE?
!!":GOTO 330
1420 IF OB(2)=-1 THEN PRINT " Q
UE VERGONHA, VOCE SE AFOGOU!":G
OTO 1360
1430 IF OB(4)>-1 THEN PRINT " V
OCE ACHOU UM REVOLVER":OB(4)=-1
:GOTO 330
1440 PRINT " VOCE SE MOLHOU TOD
O":GOTO 330
```

A linha 1410 verifica se o aventureiro está próximo ao rio. Se não estiver, o computador perguntará: "NADAR ONDE?!!" Como não existem piscinas ou praias na aventura, não há razão para escrever uma rotina de entrada para lidar com qualquer outra resposta. Nenhuma sugestão aparecerá e o jogo prosseguirá, mostrando as direções disponíveis. Caso tente atravessar o rio carregando o tijolo, o jogador morrerá — "QUE VERGONHA; VOCÊ SE AFOGOU!", comentará friamente o computador. Mas essa "morte" tem suas particularidades: depois de afundar, o jogador poderá ressuscitar, se escolher jogar novamente.

A linha 1430 verifica se o jogador carrega o revólver. Em caso negativo, o elemento da matriz de localização de objetos correspondente ao revólver será ajustado e surgirá a mensagem: "VOCÊ ENCONTROU UM REVÓLVER".

Se o aventureiro já encontrou a arma e estiver tentando atravessar o rio novamente, a linha 1440 dirá: "VOCÊ SE MOLHOU TODO".

FINALMENTE, O OLHO PERDIDO

O jogador só poderá recuperar o tão cobiçado olho perdido do totem inca se o saco de bolas de gude tiver sido encontrado. O passo a seguir será, portanto, esvaziar o saco para o olho aparecer. Aqui está a rotina.

```
1450 REM **ESVAZIAR**
1460 IF N$<>"SACO"( TO LEN N$)
THEN  PRINT "ISTO NAO PODE SER
ESVAZIADO": GOTO 400
1470 IF B(1)<>-1 THEN  LET G=1:
 GOTO 1270
1480 PRINT "AS BOLINHAS SE ESPA
LHAM PELO CHAO": LET B(5)=L: GO
TO 370
```

```
1450 REM **ESVAZIAR**
1460 IN=INSTR("SACO",N$):IF IN<
>1 THEN PRINT " ISTO NAO PODE S
ER ESVAZIADO":GOTO 330
1470 IF OB(1)<>-1 THEN G=1:GOTO
 1270
1480 PRINT " AS BOLINHAS SE ESP
ALHAM PELO     CHAO":OB(5)=L:GOT
O 370
```

```
1450  REM  ** ESVAZIAR **
1460 IN = 0: IF N$ =  LEFT$ ("S
ACO", LEN (N$)) THEN IN = 1
1465  IF IN <  > 1 THEN  PRINT
" ISTO NAO PODE SER ESVAZIADO":
 GOTO 330
1470  IF OB(1) <  >  - 1 THEN G
 = 1: GOTO 1270
1480  PRINT " AS BOLINHAS SE ES
PALHAM PELO CHAO":OB(5) = L: GO
TO 370
```

A rotina é chamada quando o aventureiro aciona o comando "**ESVAZIAR**" alguma coisa. A linha 1460 verifica se essa coisa é o saco. Se não for (**N$ < > "SACO"**), aparecerá a mensagem "ISTO NÃO PODE SER ESVAZIADO". A linha 1470 verifica se o saco está sendo carregado (**OB (1) < > -1** ou **B (1) < > -1**). Se não estiver, em vez de emitir outra mensagem, o programa pulará para a linha 1270, que exibirá a mensagem: "VOCÊ NÃO PODE LARGAR O QUE NÃO TEM". Como o jogador não quererá "LARGAR" nada, essa linha deve ser modificada, de forma a dar uma resposta adequada. Mude-a para:

```
1270 IF B(G) < > -1 THEN PRINT
"VOCE NAO PODE ";V$;" O QUE NAO
 TEM":GOTO 330
```

```
1270 IF OB(G)<>-1 THEN PRINT "V
OCE NAO PODE ";V$;" O QUE NAO T
EM":GOTO 330
```

A variável **V$** imprimirá o verbo adequado — "LARGAR" ou "ESVAZIAR" —, conforme a situação.

Se o saco estiver sendo carregado, o programa irá para a linha 1480. A mensagem "AS BOLINHAS SE ESPALHAM PELO CHÃO" será mostrada e o elemento 5 da matriz de localização de objetos será acertado.

Não há necessidade de imprimir uma mensagem para informar o resultado porque, ao pular para a linha 370, a impressão usual de uma descrição longa poderá substituí-la. Aparecerá então no vídeo a descrição usada na matriz de descrições longas (ver linha 240).

ACENDA A LÂMPADA

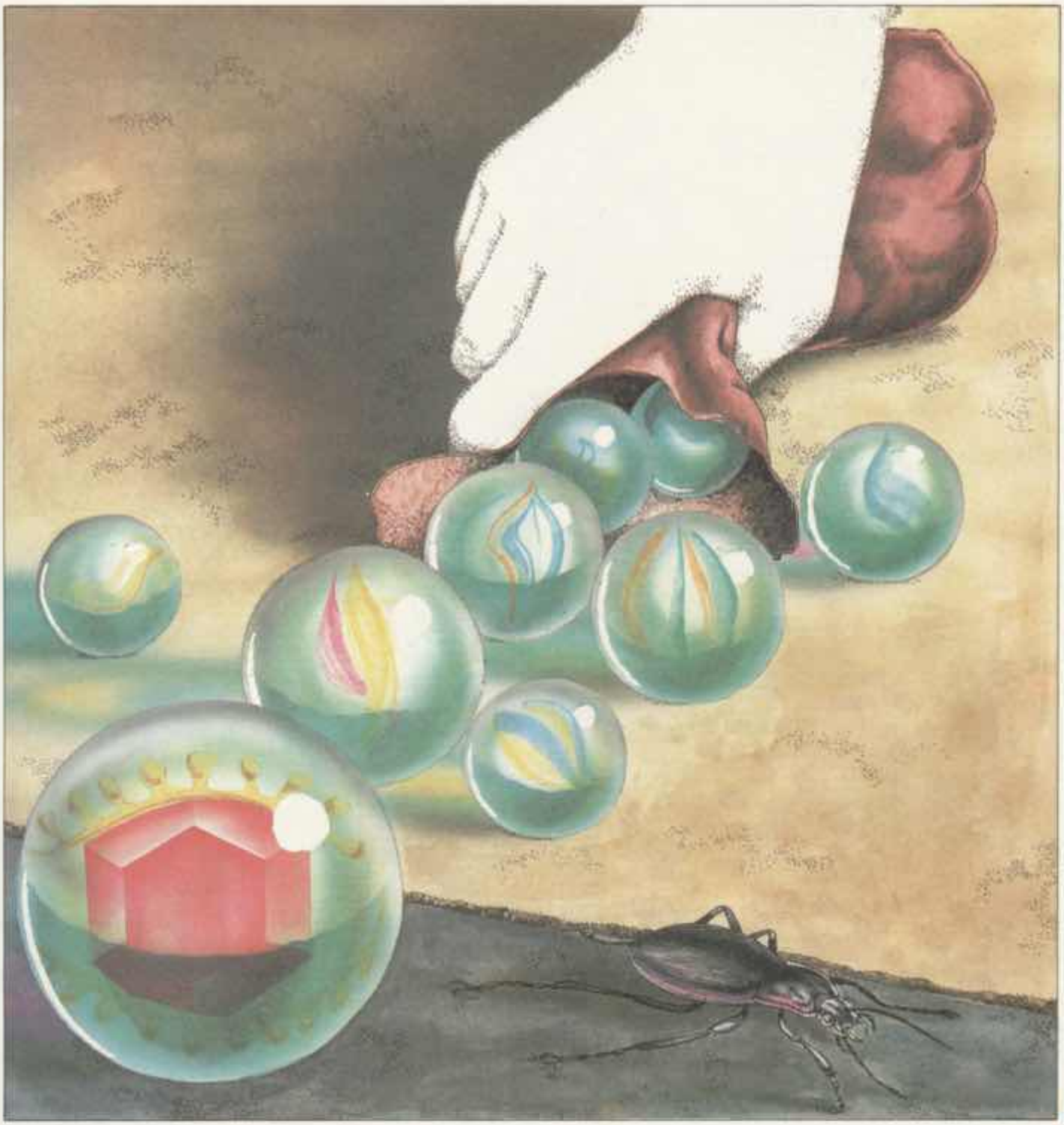
A lâmpada precisa ser acesa para que o jogador veja as saídas do quarto escuro. Se o aventureiro não estiver carregando a lanterna, a escuridão não será vencida e ele ficará preso. Esta é a rotina para acender a lâmpada.

```
1490 REM **ACENDER**
1500 IF N$<>"LAMPADA"( TO LEN N
$) THEN  PRINT "NAO PODE SER FE
ITO": GOTO 400
1510 IF B(6)<>-1 THEN  LET G=6:
 GOTO 1270
1520 IF LA=1 THEN  PRINT "JA ES
TA ACESA": GOTO 400
1530 LET LA=1: LET DA=0: PRINT
"OK": GOTO 330
```

```
1490 REM **ACENDER**
1500 IN=INSTR("LAMPADA",N$):IF
IN<>1 THEN PRINT " NAO PODE SER
 FEITO":GOTO 330
1510 IF OB(6)<>-1 THEN G=6:GOTO
 1270
1520 IF LA=1 THEN PRINT" JA EST
A ACESA":GOTO 330
1530 LA=1:PRINT "OK":GOTO 330
```

```
1490  REM  ** ACENDER **
1500 IN = 0: IF N$ =  LEFT$ ("L
AMPADA", LEN (N$)) THEN IN = 1
1505  IF IN <  > 1 THEN  PRINT
" NAO PODE SER FEITO": GOTO 330
1510  IF OB(6) <  >  - 1 THEN G
 = 6: GOTO 1270
1520  IF LA = 1 THEN  PRINT "JA
 ESTA ACESA": GOTO 330
1530 LA = 1: PRINT "OK": GOTO 3
30
```

Para que essa rotina seja requisitada, é preciso que o jogador emita a instru-



ção "**ACENDER**". A linha 1500 (1505 no Apple) é bem parecida com a linha equivalente na rotina "**ESVAZIAR**", verificando se o aventureiro escreveu a palavra "**LÂMPADA**". Caso a lanterna não esteja sendo carregada, surgirá a mensagem "VOCÊ NÃO PODE ACENDER O QUE NÃO TEM". A linha 1520 verifica se o "indicador de lâmpada acesa" — LA — está "ligado", comunicando o fato ao aventureiro. O indicador de lâmpada acesa é fixado em 1 pela linha 1530, que também imprime um "OK".

A AVENTURA CHEGA AO FIM

Quando nosso herói entra em cena, encontra uma corrente pendurada, próxima ao trono.

O que deve ele fazer? Que tal puxar a corrente? Aqui está uma rotina que cuidará das conseqüências:

```
1300 REM **PUXAR**
1310 IF N$="CORRENTE"( TO LEN N
$) THEN  LET IN=1: IF IN=1 AND
L<>24 THEN  PRINT "NADA ACONTEC
E": GOTO 400
1320 IF IN<>1 THEN  PRINT "VOCE
 NAO PODE PUXAR ISTO !": GOTO 4
00
1330 IF B(5)<>-1 THEN  PRINT "V
OCE CAI DENTRO DO VASO E VAI EM
BORA COM A DESCARGA": GOTO 1360
1335 REM **FIM DA AVENTURA**
1340 PRINT "PARABENS ! VOCE COM
PLETOU A TAREFA."
1360 PRINT '"QUER JOGAR NOVAMEN
TE (S/N)?"
1370 LET A$=INKEY$: IF A$<>"S"
AND A$<>"N" THEN  GOTO 1370
1380 IF A$="S" THEN  RUN
1390 STOP
```

```
1300 REM **PUXAR**
1310 IN=INSTR("CORRENTE",N$):IF
 IN=1 AND L<>24 THEN PRINT " NA
DA ACONTECE":GOTO 330
1320 IF IN<>1 THEN PRINT " VOCE
 NAO PODE PUXAR ISSO!":GOTO 330
1330 IF OB(5)<>-1 THEN PRINT" V
OCE CAI DENTRO DO VASO E VAI
EMBORA COM A DESCARGA":GOTO 136
0
1340 REM **FIM DA AVENTURA**
1350 PRINT "PARABENS! VOCE COMP
LETOU A TAREFA"
1360 PRINT:PRINT" QUER JOGAR NO
VAMENTE (S/N)?"
1370 A$=INKEY$:IF A$<>"S" AND A
$<>"N" THEN 1370
1380 IF A$="S" THEN RUN
1390 END
```

```
1300  REM  ** PUXAR **
1310 IN = 0: IF N$ =  LEFT$ ("C
ORRENTE", LEN (N$)) THEN IN = 1
1315  IF IN = 1 AND L <  > 24 T
HEN  PRINT "NADA ACONTECE": GOT
O 330
1320  IF IN <  > 1 THEN  PRINT
" VOCE NAO PODE PUXAR ISSO": GO
TO 330
1330  IF OB(5) =  - 1 THEN 1340
1335  PRINT " VOCE CAI DENTRO D
O VASO E VAI EMBORA COM A DESCA
RGA": GOTO 1360
1340  REM  ** FIM DA AVENTURA *
*
1350  PRINT "PARABENS ! VOCE CO
MPLETOU A TAREFA.:? "FIMDAAVENT
URA"
```



```
1360  PRINT : PRINT " QUER JOGA
R NOVAMENTE (S/N) ?"
1370  GET A$: IF A$ < > "S" AN
D A$ < > "N" THEN 1370
1380  IF A$ = "S" THEN  RUN
1390  END
```

A linha 1310 considera a possibilidade de o jogador ter trazido a corrente consigo antes mesmo de puxá-la. Neste caso, ela dirá ao aventureiro que "NADA ACONTECE".



Se o herói tentar puxar qualquer outro objeto, a linha 1320 lhe dirá: "VOCÊ NÃO PODE PUXAR ISSO".

Depois disso, acontece o inesperado. Se o jogador estiver na sala do trono, não tendo porém encontrado o olho, aparecerá a mensagem "VOCÊ CAI DENTRO DO VASO E VAI EMBORA COM A DESCARGA". E o jogo termina.

Se o aventureiro der um jeito de encontrar o olho e puxou a corrente no aposento real, nenhuma dessas linhas terá efeito e ele poderá suspirar aliviado ao receber a mensagem: "PARABÉNS! VOCÊ COMPLETOU A TAREFA. FIM DA AVENTURA".

Finalmente, nas linhas 1360 e 1380, há uma opção para jogar novamente. Esta, porém, só será útil se o aventureiro tiver sido enclausurado na masmorra ou seguido com a descarga cano adentro.

AS INSTRUÇÕES

Devemos incorporar, agora, algumas informações à rotina do jogo. Antes disso, porém, precisamos verificar o espaço restante da memória (ver página 212). Se esse espaço for pequeno, será necessário retirar todas as linhas REM — teremos então que remunerar os comandos **COSUB** que direcionam o programa para essas linhas, evitando assim mensagens de erro.

A decisão de quantas instruções serão fornecidas deve ser tomada de acordo com o espaço disponível de memória. Nelas podemos incluir até considerações quanto ao formato do vídeo do computador, o que influenciará na quantidade de detalhes a serem inseridos antes de passarmos para outra tela.

Como nossa aventura é muito simples, a rotina de instruções reduz-se a algumas linhas, contendo poucas informações. Aqui está:

```
  80 CLS : PRINT "QUER INSTRUCO
ES (S/N)?"
  90 LET A$=INKEY$: IF A$="" TH
EN  GOTO 20
  95 IF A$="S" THEN  GOSUB 6000
6000 REM **INSTRUCOES**
6010 CLS : PRINT : PRINT "  DEV
IDO A UM COLAPSO FINANCEIRO  VO
CE DEIXOU O PAIS."
6020 PRINT : PRINT "  SEUS PROB
LEMAS VAO TERMINAR      QUANDO V
OCE  ENCONTRAR  O       LEGENDA
RIO OLHO CRAVEJADO DE     BRILHA
NTES DE UM TOTEM INCA.     DEPOI
S DE FAZE-LO,VOCE TERA     QUE
ENCONTRAR A SAIDA."
6030 PRINT : PRINT "     CUIDADO
 COM O COLETOR                DE
IMPOSTOS !"
6040 PRINT AT 20,4;"PRESSIONE Q
UALQUER TECLA          PARA CONTI
NUAR"
6050 LET A$=INKEY$: IF A$="" TH
EN  GOTO 6050
6060 RETURN
```

```
10 CLS:PRINT" QUER INSTRUCOES
S/N)?"
20 A$=INKEY$:IF A$="" THEN 20
30 IF A$="S" THEN GOSUB 6000
6000 REM **INSTRUCOES**
6010 CLS:PRINT:PRINT " DEVIDO A
 UM COLAPSO FINANCEIRO  VOCE DE
IXOU O PAIS."
6020 PRINT:PRINT " SEUS PROBLEM
AS VAO TERMINAR     QUANDO VOCE
 ENCONTRAR O          LEGENDARIO
 OLHO CRAVEJADO DE    BRILHANTE
S DE UM TOTEM INCA.     DEPOIS D
E FAZE-LO VOCE TEM QUE  ENCONTR
AR A SAIDA."
6030 PRINT:PRINT"   CUIDADO  CO
M  O  COLETOR                DE
IMPOSTOS !"
6040 PRINT @451,"APERTE QUALQUE
R TECLA PARA       CONTINUAR."
6050 A$=INKEY$:IF A$="" THEN 60
50
6060 RETURN
```

```
10  HOME : PRINT "QUER INSTRUCO
ES (S/N)?"
20  GET A$: IF A$ = "" THEN 20
30  IF A$ = "S" THEN  GOSUB 600
0
6000  REM  ** INSTRUCOES **
6010  HOME : PRINT "DEVIDO A UM
 COLAPSO FINANCEIRO, VOCE "
6020  PRINT "DEIXOU O PAIS."
6025  PRINT "SEUS PROBLEMAS VAO
 TERMINAR"
6030  PRINT "QUANDO VOCE ENCONT
RAR O LEGENDARIO"
6040  PRINT "OLHO CRAVEJADO DE
BRILHANTES, DE UM "
6050  PRINT "TOTEM INCA. DEPOIS
 DE FAZE-LO, VOCE "
6060  PRINT "TERA QUE ENCONTRAR
 A SAIDA"
6070  PRINT : PRINT "CUIDADO CO
M O COLETOR DE IMPOSTOS !"
6080  PRINT : PRINT "APERTE QUA
LQUER TECLA PARA CONTINUAR"
6090  GET A$: IF A$ = "" THEN 6
090
6100  RETURN
```

Agora, grave em fita ou disco a aventura completa. A estrutura desse jogo — *O Olho Perdido do Totem Inca* — pode servir de base para suas próprias aventuras.