AS-1000

ENGEBRÁS

Rio, 31/03/86

ÍNDICE

PAG.

- INTRODUÇÃO 1
- CAPÍTULO 1: COMO INSTALAR O SEU AS-1000 3
- CAPÍTULO 2: USANDO PROGRAMAS PRONTOS 6
- CAPÍTULO 3: DANDO INSTRUÇÕES AO COMPUTADOR 11
- CAPÍTULO 4: ESCREVENDO UM PROGRAMA 15
- CAPÍTULO 5: PONTUAÇÃO E ARRUMAÇÃO NA TELA 17
- CAPÍTULO 6: LOOPS E IF 19
- CAPÍTULO 7: MAIS ALGUMA COISA A RESPEITO DE IF 23
- CAPÍTULO 8: SLOW E FAST 30
- CAPÍTULO 9: SUBROTINAS 32
- CAPÍTULO 10: QUANDO A MEMÓRIA SE ESGOTA 38
- CAPÍTULO 11: MATEMÁTICA COM O AS-1000 41
- CAPÍTULO 12: TÉCNICAS AVANÇADAS DE IMPRESSÃO 45
- CAPÍTULO 13: O CONJUNTO DE CARACTERES 47
- CAPÍTULO 14: GRÁFICOS 51
- CAPÍTULO 15: FUNÇÕES 56
- CAPÍTULO 16: TEMPO E MOVIMENTO 60
- CAPÍTULO 17: ARRAYS 64
- CAPÍTULO 18: CADEIA DE CARACTERES (STRINGS) 68
- CAPÍTULO 19: SUB-STRINGS (SUB-CADEIAS) 71
- CAPÍTULO 20: OS COMANDOS DE IMPRESSÃO 75
- CAPÍTULO 21: O AS-1000 PARA AQUELES QUE ENTENDEM DE BASIC 77
- CAPÍTULO 22: TRABALHANDO COM FLUXOGRAMAS 89
- CAPÍTULO 23: SISTEMAS DE NÚMEROS 92
- CAPÍTULO 24: COMO FUNCIONA O COMPUTADOR 95
- CAPÍTULO 25: USANDO CÓDIGOS DE MÁQUINA 97
- CAPÍTULO 26: ORGANIZAÇÃO DE ARMAZENAMENTO 99
- CAPÍTULO 27: VARIÁVEIS DO SISTEMA 102
- ANEXO : O CONJUNTO DE CARACTERES 106
- ANEXO : CÓDIGOS INDICADORES 112

Introdução

INICIANDO

Seja bem vindo ao mundo da computação. Antes de ligar seu novo AS-1ØØØ, por favor, pare um momento para pensar a respeito dessa nova e excitante aventura. Queremos garantir a você que:

1. Você gostará da computação.
2. Você a achará tão fácil quanto divertida.
3. Você não precisa ter medo do computador, você é mais inteligente do que ele. Assim ele é seu auxiliar.
4. Você cometerá erros durante o aprendizado. O computador não rirá de você.
5. Seus erros não causarão nenhum dano ao computador. Você não poderá quebrá-lo pressionando a Tecla errada.
6. Você está para dar um passo gigante no futuro. Todo mundo logo estará usando computador em todos os setores de sua vida diária e você já estará preparado para isso.

Você não precisa saber como programar um computador para usar o AS-1ØØØ, mais do que você precisa saber para dirigir um carro. Você pode querer aprender a programar — isso não é difícil e pode ser muito divertido — porém você pode usar o computador pelo resto de sua vida sem ter que aprender programação.

O computador é uma ferramenta de informação, e é a ferramenta mais versátil inventada até hoje. Devido ao fato de ele poder fazer muitas coisas, ele precisa de uma seqüência de instruções para realizar uma tarefa. Essas instruções são chamadas programas.

Existem muitos programas disponíveis para o uso cotidiano com seu AS-1ØØØ. Você pode usá-lo nos estudos, na administração doméstica ou da empresa (como cálculo de balanço, manutenção de arquivo, contabilidade/guarda de livro, taxas, inventário pessoal ou da empresa, etc.). Você pode manter estatísticas, receitas, endereços ou lista de cartões de natal, preparar relatórios, aprender e usar matemática, jogar e fazer muitas outras coisas.

Um dos usos mais importantes do AS-1ØØØ é como uma ferramenta de trabalho. Exatamente agora seus filhos estão começando a usar computadores na escola. Eles estão aprendendo a respeito de computadores e estão usando computadores para ajudá-los a aprender outras matérias. Seu AS-1ØØØ pode ajudar seus filhos a aprender em casa, quer suas escolas tenham computadores ou não. Muitos programas educativos estão disponíveis, tanto para ajuda tutorial quanto para aprendizado avançado. Você pode encontrar todos esses programas na mesma loja que você comprou seu AS-1ØØØ. Atualmente muitos outros programas estão sendo desenvolvidos para o AS-1ØØØ devido ao fato de ele ser compatível com os mais vendidos no mundo. No futuro próximo o seu computador pessoal será capaz de discar e atender o telefone, monitorar o alarme contra assaltos, controlar utensílios, molhar sua grama e realizar muitas outras tarefas para você. Esteja sempre em contato com seu revendedor!

A respeito desse Guia

Esse livro está dividido em várias seções como você pode constatar pelo índice. A primeira seção explica como ligar e preparar seu AS-1ØØØ e como usar programas de livros e fitas. A segunda parte é uma introdução elementar à programação, de modo que você possa, se quizer, ter uma idéia básica e entender melhor como o computador faz seu trabalho. As outras seções lidam com programação mais avançada e dão

referência de material para "experts".

Alguns dos últimos capítulos podem ser especialmente válidos para você se você pretende programá-lo para usos especiais:

Capítulo 21, "O AS-1ØØØ para Aqueles Que Entenden BASIC" é uma introdução às qualidades do BASIC usado pelo AS-1ØØØ, e é um bom ponto de começo para programadores experientes -- mas também é uma boa revisão para os iniciantes quando eles chegam naquele ponto do livro.

Se você pretende treinar escrevendo jogos com gráficos animados, ou fazer desenhos artísticos na tela, você deverá consultar o Capítulo 13, "O Conjunto de Caracteres", 14, "Gráficos", e 16, "Tempo e Movimento". Eventualmente você precisará do Capítulo 25, "Usando Códigos da Máquina".

Para aqueles com inclinações matemáticas ou científicas, os Capítulos 11, "Matemática com o AS-1ØØØ", e 15, "Funções" serão de maior interesse, e se você pretende manipular material de texto, você deve consultar o Capítulo 12, "Técnicas Avançadas de Impressão",18,"STRINGS" e 19, "SUBSTRINGS", e talvez 17, "ARRAYS".

Divirta-se a medida que você for conhecendo o seu AS-1ØØØ.

Capítulo 1

COMO INSTALAR O SEU COMPUTADOR AS-1000

Fornecemos tudo o que é necessário para que você comece imediatamente a usar o seu computador AS-1000 acoplado a um aparelho de TV e a um gravador comum.

Eis aqui os componentes do seu sistema de computação de uso pessoal:

1. O seu aparelho de TV. Você pode usar um aparelho a cores ou em preto e branco, porém o AS-1000 irá transmitir apenas em preto e branco.

2. Um cabo condutor de aproximadamente 1 metro de comprimento que liga o computador ao seu aparelho de TV, este cabo possui em uma das extremidades uma pequena caixa plástica denominada "BALLUN".

3. O computador. Você verá três tomadas fêmeas com as inscrições JOYSTICK, EAR e MIC, uma tomada maior com a inscrição TV, um "PENTE" para conexão de periférico e uma chave no fundo para comutar o Video normal e reverso.

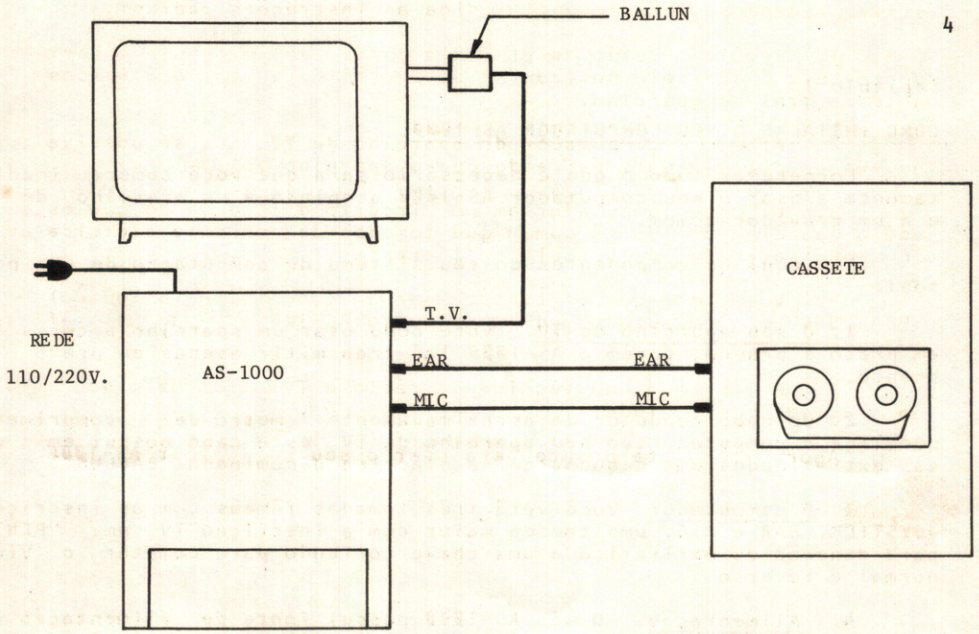
4. Alimentação. O seu AS-1000 possui fonte de alimentação interna com seletor de voltagem (110/220v) localizado no painel traseiro. Verifique a voltagem de sua cidade antes de ligar o computador. A chave Liga/Desliga está localizada na parte lateral esquerda. Você saberá que o seu AS-1000 está ligado quando o indicador vermelho no teclado estiver aceso.

5. Dois cabos de aproximadamente 0,5m de comprimento cada um, com "plugs" de 3,5 mm em cada extremidade, para ligar o computador AS-1000 ao seu gravador.

6. O seu gravador. Será preciso um gravador que recebaos "plugs" de 3,5 mm tanto na tomada do fone de ouvido ~~quanto na do fone de ouvido~~ quanto na do microfone. (Se os "plugs" não se adaptam ao seu gravador, experimente levar os cabos e o gravador a uma loja de eletrônica. Provavelmente você encontrará adaptadores de baixo custo, o que é melhor do que comprar um outro gravador). É bom ressaltar que um gravador à pilha e de baixo custo funcionará melhor com o computador AS-1000 do que um equipamento estéreo caro e possante, que na verdade pode até danificar o computador. Seria interessante, mas não essencial, que o seu gravador tivesse um marcador. Os números do marcador podem ajudar você a encontrar os programas de suas fitas, pois ele indica a posição da fita.

Para instalar o seu sistema você deve retirar os fios da antena VHF de TV de seu aparelho de televisão. Conectar os fios "BALLUN" aos terminais de TV e, em seguida, conectar a outra extremidade do cabo do AS-1000 no "PLUG" denominado "TV".

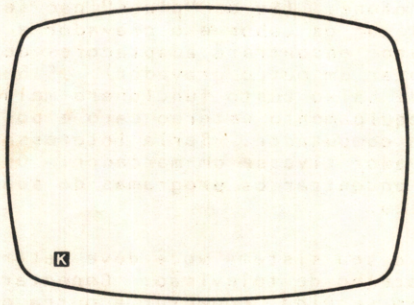
Se você tiver alguma dificuldade em instalar o seu AS-1000, consulte o seu revendedor que poderá ajudá-lo.



Em seguida ligue a tomada de alimentação na rede elétrica (certifique-se sobre a voltagem de sua cidade: 110 ou 220 V) e a seguir ligue a chave do AS-1000.

2. Abaixe todo o volume. Ligue a televisão e sintonize no canal 2.

A tela de TV deve ter este símbolo:



O K na parte de baixo do canto esquerdo da tela é chamado de cursor e indica que o computador AS-1000 está pronto para ser operado.

Para ligar o gravador ao AS-1000, use os cabos de 0,5 m. Ligando o "PLUG MIC" do gravador ao correspondente no AS-1000, proceder da mesma forma com o "PLUG EAR" (Tomada do fone de ouvido).

OBSERVAÇÕES: a imagem na tela de TV deve estar nítida. Se o aparelho estiver recebendo interferências siga as instruções abaixo:

1. Regule o botão de sintonia da TV, então regule o brilho, o contraste e o controle horizontal que geralmente são localizados na parte de trás do aparelho.

2. Afaste o computador do aparelho de TV; ou, se possível, coloque-o em posição mais baixa do que o aparelho de TV.

3. Ligue o computador em um circuito diferente do que esta ligado o aparelho de TV; é comum que tomadas de corrente elétrica em paredes opostas sejam de circuitos diferentes.

4. Você pode querer tentar usar um cabo maior (blindado) entre o "BALLUNG" e o computador, de modo que o AS-1000 fique mais afastado do aparelho de TV.

5. Procure um bom técnico de rádio e TV. Talvez o seu aparelho precise ser regulado.

Agora você está pronto para usar o seu computador AS-1000.

Capítulo 2

USANDO PROGRAMAS PRONTOS PARA SEREM OPERADOS

Atualmente há muitos programas, sejam eles de divertimento ou de utilidade, à venda para o computador AS-1ØØØ.

É enorme a quantidade de programas lançados a cada mês. Você vai ver que os programas feitos para outros computadores funcionarão em seu AS-1ØØØ.

Às vezes você introduzirá no computador um programa de uma fita cassete. Você pode então usar o mesmo programa quantas vezes quizer. Durante o tempo em que o computador estiver ligado o programa ficará armazenado em sua memória. Quando você tiver terminado e desligar o computador, o programa desaparecerá da memória interna do AS-1ØØØ mas, é claro, você terá ainda o programa na fita para carregar no computador e usar outras vezes. Esta, por exemplo, é a maneira pela qual normalmente você utilizará os jogos:

Às vezes você irá digitar o programa de um livro no computador para usá-lo depois e por isso desejará gravá-lo em fita. Assim, na próxima vez que você desejar esse programa, não precisará digitá-lo novamente no computador, mas simplesmente usar a fita para carregamento.

Às vezes haverá programas que você irá passar de uma fita, acrescentar dados, e então gravar o programa com os dados acrescidos em uma ou outra parte da fita, separada do programa original.

Vejamos como cada um desses programas é feito.

Carregando um programa no tape

Todo programa tem que ter um nome e qualquer fita que tiver mais de um programa deve ter uma lista dos nomes de todos os programas. Esta lista nem sempre irá indicar a localização de cada programa pelo número do marcador do gravador. Se você colocar o marcador em ØØØ e adiantar a fita até o número do programa que você deseja usar, você estará perto da localização do programa escolhido. Precaução: uma vez que os marcadores dos gravadores nem sempre são precisos, será melhor parar alguns números antes do número indicado. Por exemplo: no nºØ33, se a listagem indica o nº Ø37.

Com todos os componentes do seu sistema conectados e ligados, como foi visto no Capítulo 1, certifique-se de que a sua fita está no começo e que o cursor K está na tela de TV.

Ligue a tomada EAR do computador à tomada de fone de ouvido ou de "Head phone" do gravador a cerca de três quartos do volume máximo. Se o gravador tiver controle de tonalidade, regule-a de modo que o agudo fique alto e o grave fique baixo.

Então digite

LOAD

Que é o que aparece quando o cursor K está na tela? (Todas as vezes que o cursor K estiver na tela, e que qualquer tecla for acionada aparecerá a palavra-chave de comando que está impressa acima da tecla). Repare que o computador imprimiu.

LOAD

na tela.

Você notará que o cursor mudou para L . Isso significa que o computador interpretará qualquer tecla que você pressionar como a letra ou símbolo principal (maior) em qualquer tecla. (Se você pressionou a tecla J outra vez, por exemplo, aparecerá um J).

Se você desejar um dos símbolos impressos em vermelho em uma tecla, pressione a tecla SHIFT (observe que o SHIFT também está em vermelho) e pressione a tecla desejada. É o que faremos a seguir.

Se você desejar instruir o computador para gravar o programa que você escolheu, então precisa colocar o nome do programa entre aspas. Suponhamos que você deseja operar um programa para um jogo chamado STAR ZAP. Pressione a techa SHIFT e a seguir a tecla P, então aparecerão as aspas. Sem SHIFT digite ~~o nome~~ o nome do programa, certificando-se de que o nome esteja correto e no espaço certo. Digite "SHIFT P" outra vez para aspas.

A sua tela estará assim:

LOAD "STAR ZAP"

Se você cometer um erro, você poderá recomeçar pressionando a tecla NEW, e em seguida a tecla NEWLINE ou desligando o computador por um instante. No Capítulo 3 mostraremos como fazer correções facilmente, sem ter que recomeçar. Agora ligue o gravador em PLAY, e então digite

NEW LINE

Você verá listras em preto e branco na sua tela de TV. Listras finas e em forma de rabisco enquanto o computador estiver procurando o programa que você especificou, e listras retas e grossas enquanto o programa estiver sendo introduzido no AS-1ØØØ. Quando estiver pronto, as listras em preto e branco irão desaparecer e na parte de baixo do canto esquerda da tela o "registro" ØØ. Isso significa que o computador terminou o serviço de carga do programa com sucesso. Então você pode desligar o gravador.

Se a fita toda rodou ou grande parte dela, e você suspeita que o seu programa não foi carregado corretamente e você não obtém o registro ØØ, pressione a tecla BREAK. Algo saiu errado e precisa de uma pequena averiguação. (Um programa curto não leva mais de quinze segundos para ser lido, logo que as linhas grossas tenham aparecido.

É possível, por exemplo, que a fita não tenha sido posicionada corretamente. (Lembra-se de que dissemos que esses marcadores nem sempre são exatos?) talvez o computador não tenha encontrado o início do programa desejado. (Isso ocorre quando as listras finas e em forma de rabiscos não mudam para listras grossas e retas). Verifique outra vez a localização do programa. Volte a fita até um ponto anterior e tente outra vez. É mais comum que esse problema ocorra quando o volume está alto demais ou demasiadamente baixo. É preciso que o volume esteja (a) alto o suficiente para que o computador capte o programa, (b) não tão alto a ponto de deturpar o programa, (o que é raro), e (c) baixo o suficiente para que a parte silenciosa seja reconhecida como silenciosa pelo computador.

A melhor maneira de regular é pôr o volume em um nível que não torne barulhentas as partes silenciosas da fita. Você pode fazer uma verificação desligando o "plug" que está ligado à tomada do fone de

ouvido do gravador e ouvindo a fita pelo alto falante. Se o silêncio estiver com muitos ruídos, ou o volume ainda está muito alto ou existem outros problemas como os seguintes:

Alguns gravadores podem formar um circuito de realimentação com o computador AS-1000 (o que significa que a saída e a entrada do gravador estão misturadas, resultando na distorção do sinal). É por isso que recomendamos ligar um cabo de cada vez; se você tiver gravado algo com os conectores de EAR e MIC simultâneamente, é provável que você não consiga fazer a carga do programa.

Alguns gravadores gravam um sinal de 60 ciclos AC produzindo um zunido. Pode-se evitar isto operando-os com pilhas. Alguns gravadores, especificamente os velhos e gastos, são intrinsecamente barulhentos e reproduzem muitos ruídos estranhos nas fitas. Talvez você tenham que comprar um outro gravador.

Talvez você tenha de mover o "plug" na tomada do fone de ouvido, pois em alguns gravadores se perde o contato se o "plug" for empurrado muito para dentro. Se você puxá-lo um pouco para fora, o "plug" poderá ficar em uma posição mais correta. É possível carregar um programa sem usar o seu nome. Se você digitar.

LOAD" "

Então ligue o gravador e aperte a tecla NEWLINE, o computador AS-1000 carregará o primeiro programa que aparecer. Caso você esqueça o nome do programa você poderá lançar mão deste meio. Você pode até carregar uma série de programas em uma fita e usá-los um após o outro. Quando você desligar o gravador, após ter terminado o carregamento de um programa, a fita estará na posição de carregar o programa seguinte. Quando você tiver carregado um programa com sucesso (verifique isso pelo registro 00), você pode usá-lo ao pressionar.

RUN e NEWLINE

E então siga as instruções que o próprio programa apresenta na tela e qualquer instrução impressa (Você pode olhar o programa ao digitar

LIST e NEWLINE

e então use-o se você quiser).

Digitando um programa e armazenando-o

São encontrados muitos programas curtos em livros e revistas. Você pode usá-los simplesmente digitando-os. Digite-os exatamente da forma que apareçam na publicação, certificando-se de que escreveu corretamente e que todas as pontuações e espaços estão corretos. A listagem pode ser verificada comparando o que está na tela com a versão impressa. Veja o Capítulo 3 para saber como fazer correções com facilidade.

Quando você tiver terminado de digitar o programa você o executará ao pressionar

RUN e NEWLINE

da forma acima. Quando você terminar de usar o programa, quer chegue ao final quer interrompa o programa ao pressionar a tecla com a inscrição BREAK, você pode obter a listagem de volta na tela ao pressionar.

NEWLINE

outra vez. Então, você pode verificar que o programa funciona usando-o, e que você o digitou corretamente. Você pode armazená-lo na fita para uso posterior, (você não tem que testá-lo imediatamente, mas é claro que você pode correr o risco de armazenar um programa que não funciona).

Armazenando um programa na fita

Como dissemos anteriormente, cada programa tem que ter um nome. O computador AS-1ØØØ, na verdade, não armazenará um programa, na fita, que não tenha nome. Você pode dar um nome a um programa que você criou, usar o nome de um programa que você digitou, como aquele acima, ou mesmo mudar o seu nome para outro que você preferir. Seja qual for o nome que você deu ao programa quando o armazenou em fita, será este nome que você terá que pedir para carregá-lo mais tarde.

OBSERVAÇÃO: É uma boa idéia colocar o nome de um programa na listagem daquele programa, de modo que você possa verificar duas vezes se tem o programa certo. A maneira mais fácil é usar a linha REM no início. REM — a palavra-chave acima da tecla E — significa advertência ou lembrete. Qualquer linha em um programa que começe com REM não é usada pelo computador, mas é mostrada na tela para ajudar as pessoas a usarem o programa. O computador AS-1ØØØ põe automaticamente a linha de menor número de um programa em primeiro lugar e dispõe todas as linhas em ordem numérica. Você pode colocar uma nova primeira linha — uma informação REM — em um programa, após tê-lo carregado, digitando

5 REM "STAR ZAP" NEWLINE

usando um número de linha menor do que a menor de todas as linhas na listagem (você usaria 5 se, por exemplo, a primeira linha do programa começasse com o número 1Ø) e naturalmente, o nome de seu programa.

Ligue a tomada MIC do computador à tomada do microfone do gravador. Posicione a fita em uma parte que esteja em branco, ou em uma parte que você queira gravar por cima

Digite

SAVE "STAR ZAP"

Ligue o gravador em RECORD, então pressione

NEWLINE

Observe a tela da TV. Você verá as listras em preto e branco, e eventualmente essas listras desaparecerão e a tela mostrará Ø/Ø.

Para verificar se o gravador recebeu o programa corretamente, você pode ouvir o programa pelo alto falante.

Volte a fita até onde você começou e ponha-a para tocar.

Você ouvirá em seqüência:

1. um zumbido suave. Este é o sinal que aparece antes de você ter acionado a tecla ~~ENTER~~. NEW LINE

2. Seis segundos de silêncio.

3. um zunido alto e muito estridente, que é o próprio programa. Quanto maior for o programa, maior o tempo de duração do ruído.

4. o zunido suave novamente.

OBSERVAÇÃO: Uma técnica que torna mais fácil encontrar programas que você grava é dizer o nome do programa na fita, através do microfone, antes de ligar o computador ao gravador. Então você pode procurar pela sua informação, o local em que se encontra o programa. Você pode testar o seu êxito em armazenar um programa seguindo as instruções de LOAD acima.

Armazenar programas com seus próprios dados registrados

Alguns programas são feitos para que você registre os seus próprios dados — armazene listragens gráficas, etc. Estes programas são facilmente usados seguindo os mesmos procedimentos que já discutimos.

1. LOAD carregue o programa como descrevemos.

2. RUN opere o programa registrando seus próprios dados conforme é pedido.

3. SAVE o programa com os dados, usando um novo nome para distinguí-lo do programa original. Se por exemplo você carrega um programa chamado "Calculador" e então preenche com os seus próprios registros financeiros, você pode desejar guardar a versão preenchida sob o nome de "Finanças".

Como você vê, há várias maneiras de usar o seu computador AS-1000 sem aprender programação de computador. Mas se você quizer se aprofundar no assunto veja os próximos capítulos.

Capítulo 3

DANDO INSTRUÇÕES AO COMPUTADOR

Agora que a tela se encontra como é apresentada na figura do capítulo 1, você poderá introduzir algumas "instruções" especiais de computação. Por exemplo:

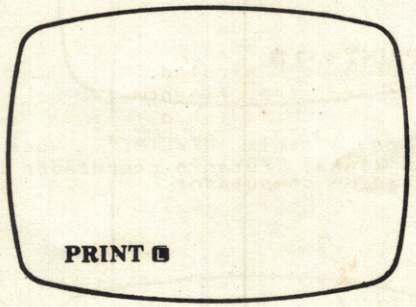
PRINT 2+2

faz o computador realizar a soma 2+2 e mostra (PRINT) o resultado na tela. Uma instrução como essa, que o computador opera imediatamente, é tecnicamente chamada de comando BASIC.

Existem outras instruções BASIC que não são executadas automaticamente. Essas instruções serão vistas nos capítulos 4-7).

Para digitar esse comando:

1. Primeiro acione a tecla PRINT. Como você pode observar, existe uma tecla para cada letra no teclado, porém não soletre a palavra P,R,I,N,T. Logo que você acionar P a palavra inteira aparecerá na tela, seguida por um espaço, então a tela estará assim:



O fato é que, no início de cada comando o computador precisa de uma palavra-chave — uma palavra que indique o que se pretende. As palavras-chave estão impressas acima de cada tecla e, como você pode observar, PRINT está acima da tecla P, por isso, para obter PRINT você terá que acionar a tecla P.

O computador indica que é necessário uma palavra chave através do K que você usou para iniciar. Sempre há uma letra branca sobre fundo preto (vídeo inverso) K ou L (ou como veremos mais tarde, F ou G), chamados de cursor. K significa que qualquer que seja a tecla acionada, esta será tomada como uma palavra-chave. Como você viu, depois que você acionou P para obter PRINT, o K passou para L .

Este sistema de acionar apenas uma tecla para obter mais de um símbolo é muito usado no AS-1000. No fim do manual as palavras com suas próprias teclas serão impressas em NEGRITO.

Palavras-chave não podem ser digitadas letra por letra

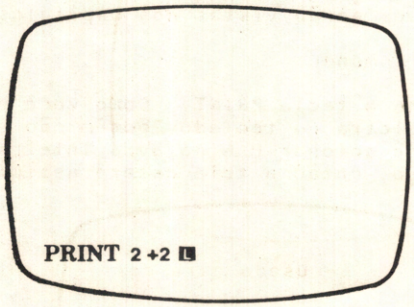
2. Agora digite 2. Isso não deverá causar problemas. Novamente você verá o 2 aparecer na tela e L se moverá para a direita.

Observe também como um espaço é automaticamente inserido entre PRINT e 2. Isso é feito tanto quanto possível, de modo que dificilmen

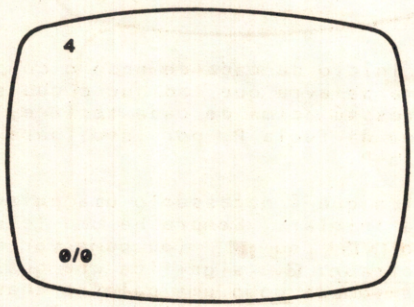
te você terá que digitar um espaço. Se você o digita ele aparecerá na tela, porém não afetará o comando.

3. Agora digite +. Esse é um caracter "SHIFTADO" (tais caracteres estão marcados em vermelho — a cor da techa SHIFT — no canto superior direito de cada tecla), e para obter "+" você terá que manter a techa SHIFT pressionada e pressionar a tecla K.

4. Agora digite 2 novamente. A tela ficará assim:



5. Agora pressione a tecla 'NEWLINE'. Você pressiona essa tecla quando termina uma linha. Então o computador computará. Em nosso caso a tela mudará para

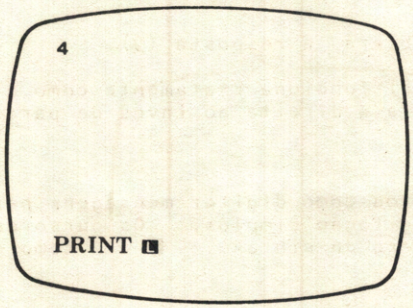


4 é a resposta — mas é claro que você não precisa de um computador para chegar a essa conclusão.

Ø/Ø é a mensagem que o computador usa para informar como ele chegou a tal resultado. (Observe que zero é cortado por uma barra para ser distinguido de O maiúsculo. Isso é normal em computação). O primeiro Ø indica "OK, nenhum problema". (Bem no fim do livro há uma lista de outras mensagens de códigos que podem aparecer, que você pode consultar, por exemplo, se alguma coisa não der certo). O segundo Ø significa que "a última coisa executada é a linha Ø". Você verá mais tarde — quando começar a escrever programas — que uma instrução pode

receber um número e ser armazenada para execução posterior: Esta é então a linha do programa. Os comandos que não se encontrem nos programas não têm números, então, para fins de mensagem, o computador os chama de "linha Ø".

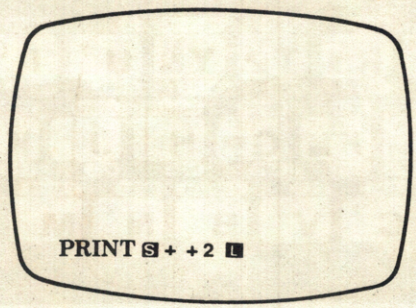
Você deve considerar a denotação como se ela estivesse escondendo o cursor K — se você acionar P para PRINT, nesse caso, a mensagem desaparecerá e a tela mudará para



O cursor pode também ser usado para corrigir erros: digite + +2 para obter

PRINT + +2 L

na última linha. Quando você pressiona NEWLINE você obtém



O S é o sinal de erro de sintaxe (sintaxe é a mensagem gramatical que indica o que é permitido e o que não é); mostra que o computador recebeu até a mensagem PRINT mas depois disso não houve uma mensagem "válida".

O que você fará, é claro, é apagar o primeiro + e substituí-lo, vamos dizer, com 3. Primeiro você terá que mover o cursor de modo que ele fique exatamente à direita do primeiro +; há duas chaves, <= e => (SHIFT 5 e SHIFT 8) para mover o cursor para a esquerda e para a direita. Com SHIFT pressionado acione a tecla duas vezes. Isso moverá o cursor dois espaços para a esquerda para dar

PRINT + L + 2

Agora pressione a tecla RUBOUT (SHIFT Ø) e obterá

PRINT L +2

RUBOUT remove o caracter (ou a palavra-chave) imediatamente à esquerda do cursor.

Se você agora pressionar 3, um "3" será inserido, de novo imediatamente à esquerda do cursor, dando

PRINT 3 L +2

e pressionando NEWLINE obterá a resposta (5).

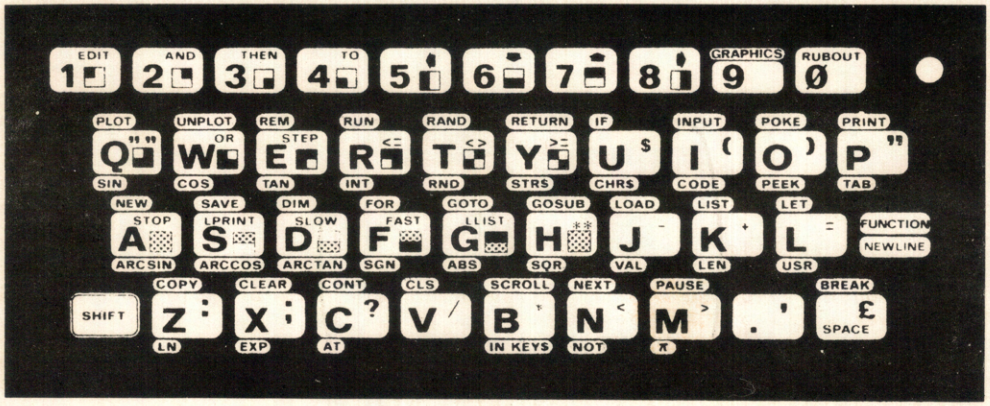
A tecla =>(SHIFT 8) funciona exatamente como a techa,<= salvo que ela move o cursor para a direita ao invés de para a esquerda.

Resumo

Esse capítulo mostrou como digitar mensagens para o AS-1ØØØ, explicando o sistema de digitação simples. Os cursores K e L , mensagens; o sinal de erro de sintaxe, S e como corrigir erros usando <= , => e RUBOUT.

O teclado

Aqui está uma foto do teclado.



(figura da p.20)

Se você não estiver familiarizado com máquinas de escrever você poderá usar essa ilustração para aprender a localização das teclas. Lembre-se que para usar SHIFT você terá que mantê-lo pressionado enquanto aciona uma outra tecla. Não confunda o dígito Ø com a letra O.

Se você é datilógrafo lembre-se de que você não poderá usar o L minúsculo para o numeral 1.

Capítulo 4

ESCREVENDO UM PROGRAMA

Vamos agora escrever o nosso primeiro programa de computador, que é uma série de instruções para o computador. Demos instruções ao computador, no último capítulo, que são executadas imediatamente. Agora vamos começar a criar mensagens que serão postas em ordem pelo AS-1ØØØ. Apesar de começarmos com um programa pequeno, há programas com muitos milhares de linhas, com procedimentos complicados. A força da computação está na habilidade da máquina de receber, armazenar e executar muitos programas diferentes e complexos.

Programaremos o AS-1ØØØ em uma "linguagem" de computador chamada BASIC (BEGINNER'S ALL-PURPOSE SYMBOLIC INSTRUCTION CODE). A linguagem BASIC foi inventada em Dartmouth College e é mais parecida com uma linguagem inglesa do que com qualquer outra linguagem de computador, podendo ser usada de modo interativo quando o usuário digita alguma coisa e o computador responde imediatamente via vídeo, como um aparelho de televisão. Quando você tiver aprendido a programar o AS-1ØØØ você verá que programas em BASIC de outros livros e revistas também podem ser usados. É quase certo que você tenha que alterar esses programas ligeiramente para fazer com que eles funcionem no AS-1ØØØ, uma vez que toda máquina tem sua versão de BASIC ligeiramente diferente.

Iniciemos. Em primeiro lugar digite NEW e em seguida NEWLINE. Isso apaga tudo que pode ter ficado no Capítulo 3 no computador e prepara o computador para entrar em ação.

Então digite

```
1Ø PRINT "HELLO, JACK"
```

Se você é datilógrafo certifique-se de que está usando o numeral 1, e não o L minúsculo no 1Ø, por um único motivo: não há minúsculo no AS-1ØØØ...). Esteja certo também de estar usando zero (Ø cortado por uma barra) e não a letra O. O 1Ø na frente do comando que você digitou antes transforma esse comando em uma linha de programa. Digite NEWLINE e veja o que acontece.

O cursor está pronto de novo, na parte de baixo da tela. Digite:

```
2Ø  GOTO 1Ø.          NEWLINE
```

Observe que GOTO é uma palavra chave (sobre a tecla G) e que não deve ser escrito letra por letra.

Agora você tem um programa! Para executá-lo digite RUN (a palavra chave que está sobre R: Viu como a maioria das palavras-chave estão sobre as letras que as iniciam?) e NEWLINE.

O que é que você acha? Há muita matéria pelo pouco esforço que você dispendeu. Como dissemos o computador não é muito inteligente. Porém é rápido, exato e incansável na realização de repetições. Um dos comandos mais poderosos em BASIC é GOTO, que manda o computador voltar a uma linha anterior e repetir o que já foi feito.

Veja a mensagem 5/1Ø na parte de baixo da tela. Isso significa que o computador parou porque a tela está cheia (5), e executou por último a linha 1Ø. Você pode limpar a tela para permitir que o computador continue, digitando CONT seguido de NEWLINE.

Mais ainda: enquanto o computador estiver mostrando "Hello, Jack" na tela, pressione a tecla BREAK e veja que a lista parou exatamente no lugar em que se encontrava quando você pressionou BREAK. O computador verifica, ao final de cada linha de programa, se BREAK foi pressionado; se isso aconteceu ele pára o programa. (Você pode usar BREAK também para parar um programa que está escapando, ou "um loop sem fim", ou um "LOADING" (carregamento) falho da fita. Se BREAK não funcionar você terá que desligar o computador, o que causará a perda de tudo o que estiver na memória do computador).

Você pode recomeçar o programa após o BREAK, digitando CONT e NEWLINE em seguida. CONTinue permite que você continue mesmo quando a tela está cheia, quando você interrompe o programa com BREAK ou quando há interrupção com o comando STOP.

Muito bem. Agora que você está farto de brincar com BREAK e CONT, deixe a tela ficar cheia e acione NEWLINE de novo. Seu programa está de novo na tela. Você pode operá-lo de novo (continue: RUN, NEWLINE) e mostre-o de novo (NEWLINE). Você pode manter alguma coisa no computador até que você a apague, que a memória do computador se esgote ou que você desligue o aparelho.

Capítulo 5

PONTUAÇÃO E ARRUMAÇÃO DE SAÍDAS NA TELA

Você pode fazer maravilhas, com os sinais de pontuação, em um programa BASIC no AS-1ØØØ. Por exemplo, limpemos com NEW, NEWLINE, agora digite

```
1Ø PRINT "JACK",

2Ø GOTO 1Ø
```

Observe que resolvemos dispensar o "HELLO". E, ao invés de JACK, por que não usar o seu próprio nome? Em um programa o computador tratará tudo que estiver entre aspas como uma cadeia (uma cadeia de caracteres) e uma cadeia não precisa ter tamanho específico nem tão pouco tem que ser sentido. O AS-1ØØØ imprimirá JACK, felizmente, da mesma forma ele imprimirá HERMAN, JAQUELINE, ou KLJY66??F.

Uma observação importante: se você quiser colocar a vírgula, esta deve ser fora e após as aspas. Digite RUN, NEWLINE.

A tela tem uma dimensão de 32 caracteres para operação no AS-1ØØØ numerados de Ø a 31 e a vírgula se move do início da próxima entrada para a posição 16 ou para a posição Ø (da próxima linha), dependendo / de onde terminou a última entrada. Desse modo você pode fazer duas colunas, do tamanho do seu nome, mas que não tenha mais de 16 letras.

Vejamos o que faz um ponto e vírgula. Não apague o programa com NEW, apenas pressione NEWLINE para trazê-lo de volta.

Aqui está ele. Bem na parte de cima da tela e também está um tipo diferente de cursor, > , mostrando que a linha 2Ø foi a última / que você deu entrada. Prenda a tecla SHIFT e pressione a seta ↑ , (SHIFT 7), movendo o cursor para a linha 1Ø. Prenda SHIFT e pressione EDIT (SHIFT 1), e a linha 1Ø (aquela que está com o cursor) vai para o fim da tela, onde você pode trabalhar com ela. Agora use a seta → , (SHIFT 8), para mover o cursor para o fim da linha, em seguida RUBOUT (SHIFT Ø) para apagar a vírgula. No lugar da vírgula, digite um ponto e vírgula (SHIFT X) e NEWLINE. A linha editada substituirá a linha antiga no programa. Ponha o programa em operação (RUN).

O ponto e vírgula inicia a segunda entrada PRINT exatamente após a primeira. A escrita, porém, fica muito comprimida. Precisamos acrescentar um espaço. Pressione NEWLINE para obter a listagem do programa de volta. Use o comando EDIT e as setas cursoras para trazer para baixo a linha 1Ø e adicionar um espaço entre o K em JACK (ou entre a última letra do seu nome) e as aspas. Posicione o cursor entre K e " e pressione SPACE. Em seguida NEWLINE, RUN e NEWLINE.

Há um truque que você pode usar com a função EDIT. Se você está operando uma linha muito longa do programa e ela está toda confusa, de tal modo que você deseja apagá-la e começar de novo, pressione RUBOUT quantas vezes for necessário para atingir a margem esquerda, ou então apenas pressione EDIT, atingindo a última linha correta e apagando aquela que você deseja apagar. Pressione NEWLINE, e volte a linha não EDITada ao programa. Isso proverá um espaço em branco para você usar , na parte inferior da tela.

Ainda temos um problema com JACK, e provavelmente com o seu nome também. O nome é dividido de diferentes maneiras nos finais das linhas. Suponhamos que queiramos alinhar os nomes em colunas. Podemos

conseguir duas colunas com uma vírgula. Como conseguir mais colunas?

Com o comando TAB, que ~~na verdade~~ é uma função. (Na verdade ele não é uma função, mas é considerado como tal devido ao local em que se encontra no teclado).

Limpe o programa (NEW, NEWLINE) e digite

```
1Ø PRINT
```

TAB está escrito sob a tecla P. Para obter essa função (e para obter todas as outras funções que estão escritas sob as teclas), mantenha SHIFT pressionado e pressione FUNCTION (SHIFT NEWLINE). O cursor mudará para F, significando que o AS-1ØØØ interpretará a próxima tecla como uma função. (Agora você pode soltar o SHIFT e o cursor continuará F). Pressione TAB. Essa função aparecerá na linha do programa e o cursor mudará de novo para L. (Você não pode ter duas funções seguidas sem algo para elas trabalhar, assim o computador, automaticamente, desvia do modo de função). Digite um 1 e um ponto e vírgula, "JACK" ou o seu nome. Então a linha está conforme a seguir (mas não acione NEWLINE; não estamos prontos ainda):

```
1Ø PRINT TAB 1; "JACK"
```

Agora queremos separar as colunas de nomes com um espaço. Como JACK tem quatro letras adicionamos um espaço fazendo a próxima entrada em TAB 6, usando ponto e vírgula entre cada entrada:

```
1Ø PRINT TAB 1; "JACK"; TAB 6; "JACK"
```

Se o seu nome não tem o mesmo número de letras que JACK, apenas adicione o número de letras, mais uma para o espaço, para a última posição de TAB para obter a posição da próxima função TAB. Continue fazendo isso, mas verifique se você não tem mais de 31 letras ou espaços ao todo. Não interessa quantas vezes você coloca o seu nome (se você quiser, apenas elimine uma entrada usando RUBOUT, as setas cursoras e EDIT se necessário):

```
1Ø PRINT TAB 1; "JACK"; TAB 6; "JACK" TAB 11;
"JACK"; TAB 16; "JACK"; TAB 21; "JACK"; TAB 26; "JACK"
```

e acione NEWLINE. Vamos por ora, recordar a respeito de NEWLINE. Pressione NEWLINE após cada linha que você tenha feito. Digite

```
2Ø GOTO 1Ø
```

Opere (RUN) o programa. Não está bem arrumadinho?



Capítulo 6

"LOOPS" e IF

Dissemos anteriormente que GOTO pode ser um comando muito poderoso, pois ele faz o computador realizar uma operação repetidas vezes. Isso é chamado "looping" e a parte do programa que é repetida e chamada "loop". Geralmente, quando você não tem um meio de terminar as repetições (um meio que não seja BREAK ou desligando a tomada) você tem o que chamamos "loop sem fim", que é uma prática confusa na programação. Normalmente você quer a ação repetida um número específico de vezes (duas às vezes ou uma dezena de vezes - é a mesma coisa para um computador). Você pode usar GOTO e usar um indicador ou controle de variável. Digite:

```
10 LET I=1
```

A instrução LET determina um valor para uma variável. Nesse caso, I é o nome da variável e foi determinado para ele o valor de 1. (Você pode usar qualquer número de letras ou de dígitos no nome de uma variável no BASIC do AS-1000 desde que a primeira seja uma letra). Algumas vezes, para maior clareza, você escreverá uma palavra por completo; outras vezes você apenas usa uma letra, para maior rapidez. Para variáveis regulares, você pode apenas determinar valores numéricos, porém você pode determinar qualquer coisa, entre "aspas", para formar uma cadeia de variáveis, que sempre são denominados por uma única letra seguida pelo símbolo de dólar (SHIFT U). No programa anterior, se seu nome foi ABERCROMBIE, ao invés de escrevê-lo todas as vezes, você poderia ter feito isto:

```
10 LET A="ABERCROMBIE"

20 PRINT TAB 1; A$; TAB 13; A$; ... e assim

por diante
```

Continuemos com o programa desse capítulo. Digite:

```
20 PRINT "HELLO, JACK"

30 LET I=I+1
```

Que isso, camarada! Quer que eu acredite que I=I+1? Que matemática é essa?

Mas isso não é matemática! Isso é programação e para uma instrução BASIC a expressão está correta. Ela significa "seja o novo valor de I igual ao valor antigo mais 1". Em outras palavras, acrescente 1 ao valor de I. Desse modo cada vez que passarmos o programa, em um "loop" repetido, o indicador será aumentado de 1. Já vamos ver a razão disso. Digite:

```
40 IF I=6 THEN GOTO 60
```

Ah! Então cada vez que chegarmos à linha 40 o AS-1000 checará o valor de I. Quando atingir 6 - na sexta vez de repetição do programa - ele pula para a linha 60. Até I igualar a 6 o programa irá passando para a próxima linha após a linha 40.

```
50 GOTO 20

60 STOP
```

Se I=6 vamos para 6Ø e STOP. Se ele não é igual a 6, passamos de 4Ø para 5Ø que nos mandará pular para 2Ø e começar a partir de lá. Vejamos como ele funciona.

Imprimimos "HELLO JACK" cinco vezes (na sexta vez fomos para 6Ø). Pressione NEWLINE para receber a listagem de novo. A instrução IF, com sua habilidade de tomar decisões, é uma outra característica importante do BASIC.

Mas esse é um "loop" razoavelmente mal feito. Há uma abordagem mais clara, chamada "loop" FOR/NEXT. Digite isso sem apagar o programa acima:

```
1ØØ FOR I=1 TO 5
```

(Use SHIFT 4 para TO, não o escreva por extenso)

```
11Ø PRINT "HELLO, JACK"

12Ø NEXT I
```

(I é comumente usado como contador ou controle de variável, por acaso; controle de variáveis de "loops" FOR NEXT devem ter os nomes com apenas uma letra).

Para operar aquela parte do programa digite RUN 1ØØ ao invés de apenas RUN (o computador está armazenando todas as linhas e começará a passar porqualquer linha que você tenha escolhido). Depois de ter visto como aquele programa de três linhas executa o mesmo que o de seis linhas faz, dê novamente uma olhada na listagem digitando LIST 1ØØ. Se você pretende verificar as duas listagens (ou melhor, tudo aquilo na verdade é uma única listagem), pressione NEWLINE de novo- ou LIST sem número de linha, e em seguida, NEWLINE. É dessa forma que a maioria dos BASICs manipulam as listagens de função. O AS-1ØØØ poder chamar uma listagem apenas com NEWLINE, é novidade. OBSERVAÇÃO ESPECIAL: Quando você usa "RUN 1ØØ" você apaga todas as variáveis que estão antes da operação. Se por algum motivo você deseja armazenar variáveis para as quais você tenha atribuído valores, use "GOTO 1ØØ". O comando RUN significa, para o AS-1ØØØ, CLEAR (que você pode introduzir através do teclado se quiser apagar todas as variáveis. Por exemplo, se você precisa do espaço - mas também precisa manter o programa e a tela intactos), então use GOTO.

Mais uma pequena sutileza é que o controle de variável não tem que iniciar por 1 todas as vezes: você pode mudar este 1 para o que você quiser usando uma parte STEP na instrução FOR. Se a linha 1ØØ está escrita da seguinte forma:

```
1ØØ FOR I=1 TO 5 STEP 2
```

você obteria 3 "HELLO, JACK"s antes do "loop" ter atingido seu limite e parado. O STEP não precisa estar em números inteiros, e nem o controle de valores atingir o limite - ele continua o "looping" conquanto que ele seja menor do que ou igual ao limite.

Se você estiver operando dois "loops" FOR/NEXT conjuntamente, um dentro do outro, você precisa ter muito cuidado. Experimente este programa, que imprime um conjunto completo de dominós de 6 pontos:

```
1Ø FOR M = Ø TO 6

2Ø FOR N = Ø TO M
```

```
30 PRINT M; ""; ""; N; "";
40 NEXT N
50 PRINT
60 NEXT M
```

Você pode observar que o "loop" N está completamente dentro do "loop" M - em outras palavras, eles estão corretamente inseridos. O que tem que ser evitado é dois "loops" FOR/NEXT que se sobrepõem sem que um esteja completamente dentro do outro, como se segue:

```
10 FOR M=0 TO 6
20 FOR N=0 TO M
30 PRINT M; ""; ""; N; "";
40 NEXT M
50 PRINT
60 NEXT N
```

ERRADO!

Dois "loops" FOR/NEXT têm que aparecer um dentro do outro, ou então completamente separados. Uma outra coisa a ser evitada é pular de fora do "loop" FOR/NEXT para o meio dele. (Você pode fazer isso com um GOTO dirigindo o computador para uma linha após a instrução FOR, e quando você chega à instrução NEXT, provavelmente você cometerá um erro de mensagem 1 ou 2).

Agora vamos dar uma olhadinha em um programa que pode ser de alguma utilidade e também em algumas instruções que não vimos antes. Primeiramente digite-o e vamos analisá-lo em seguida:

```
10 REM  PROGRAME PARA MULTIPLICAR POR PI (escreva pi por extenso
nas linhas 10 e 20)
20 PRINT "ESTE PROGRAMA MULTIPLICA QUALQUER NÚMERO POR PI"
30 PRINT
40 PRINT "INTRODUZA UM NÚMERO"
50 INPUT A
60 PRINT A; "VEZES PI="; A*PI
```

(Use a função PI - a função P.I. sob a tecla M - apesar de o computador imprimi-lo exatamente como ele aparece na versão escrita por extenso nas linhas 10 e 20)

```
70 PRINT
80 GOTO 30
```

Falemos um pouco a respeito desse programa antes de operá-lo.

1. Observe que os números de linha que temos usado são múltiplos de 10. Faça sempre um rascunho disso. Devido ao fato de o computador colocar as linhas do programa em ordem numérica automaticamente,

você pode inserir uma linha 15 se você quiser ou precisar mais tarde - depois dela, uma linha 12 e uma linha 13.

2. A palavra-chave REM para "REMARK" ou "REMINDER", identifica uma linha que está impressa na listagem do programa para ajudar um futuro usuário a entender o programa, mas é ignorada pelo computador durante um RUN.

3. O comando palavra-chave INPUT na linha 5Ø faz o computador parar e esperar que o usuário introduza um número. O cursor L na parte de baixo da tela, indica que o AS-1ØØØ está esperando uma entrada durante a operação do programa, porém é melhor indicar mais claramente ao usuário que ele terá que dar a entrada. Essa é a função da linha 4Ø. OBSERVAÇÃO ESPECIAL: Quando você quiser introduzir uma cadeia de caracteres, como veremos mais tarde, você deverá indicar uma variável de cadeia no programa - A$ em vez de A - e o cursor aparecerá prontamente na parte inferior da tela, entre aspas. Se nesse ponto você pretende parar o programa (STOP), não basta que você introduza a palavra STOP - o computador a considerará como uma cadeia! Você deve usar a tecla da seta que indica para a esquerda movendo o cursor para um ponto fora e antes das aspas e digitar STOP em seguida.

4. As linhas 3Ø e 7Ø, PRINT com nada a ser impresso, mandam o computador imprimir uma linha em branco, ou então pular uma linha. Isso simplesmente torna mais fácil a leitura das saídas na tela devido ao espaço extra. Após ter operado o programa talvez você queira mudar a listagem digitando 3Ø NEWLINE e 7Ø NEWLINE (eliminando as duas linhas) e ver o quanto a tela se apresenta mais compacta.

5. Na linha 6Ø se encontra mais um espaço para legibilidade. Observe que há um espaço digitado após a primeira aspa e antes da segunda. Não os retire.

Agora opere (RUN) o programa.

Você pode fazer a mesma coisa com uma calculadora de bolso que tenha memória (armazene um valor preciso para PI na memória, introduza um novo número, acione a tecla de multiplicação e a tecla de "chamada de memória" para cada resposta). Mas mesmo para essa parte simples da matemática é mais fácil e mais vantajoso usar o computador. Além disso o computador não se cansará de guiar você através dos exercícios, lembrando-o durante o tempo todo do que você está fazendo. Imagine a vantagem que é você ter o computador para fazer tudo isso para você em uma tarefa mais complexa!

Capítulo 7

MAIS ALGUMA COISA A RESPEITO DE IF

Até agora todos os programas vistos têm sido bastante previsíveis, simples. Seguindo todas as instruções do começo ao fim e às vezes retornando novamente ao ponto inicial. Isto não é muito útil na prática, esperar-se-ia que o computador estivesse apto a reconhecer casos diferentes e agir corretamente; ele o faz usando a informação IF.

Limpe o computador (usando NEW), digite e execute este programa:

```
10 PRINT "ESTOU PENSANDO EM A"
20 PRINT "NUMERE DE 1 a 5"
30 PRINT "VOCÊ PODE ADIVINHAR?"
40 LET A=INT (RND*5) + 1
50 INPUT B
60 IF A=B THEN GOTO 90
70 PRINT "ADIVINHE DE NOVO"
80 GOTO 50
90 PRINT "CORRETO"
```

Para discussão sobre RND veja Capítulo 5, "Funções", exercício 4.

Conforme você pode ver, uma informação IF toma o seguinte formato: IF condição THEN informação.

As informações aqui são GOTO. Mas poderiam ser quaisquer outras, até mesmo IF. A condição é algo que deve ser provado se é verdadeira ou falsa. Se for verdadeira, a instrução após THEN será executada; caso contrário, será ignorada.

As condições mais usadas comparam dois números ou duas cadeias de caracteres podendo testar se dois números são iguais ou se um é maior do que outro; pode também testar se duas cadeias de caracteres são iguais ou se uma vem antes da outra em ordem alfabética. Para tal usam-se relações = <, >, < =, > = e <> = o que usamos duas vezes no programa (uma vez para número e outra para cadeias de caracteres) significa "igual". Não é o mesmo que o = na informação LET. < siginifica "menor do que", assim:

```
1 < 2
-2 < -1
-3 < 1
```

juntos (todos são verdadeiros) mas

```
1 < 0 e
0 < -2
```

não (são falsos). Para ver como isto funciona, vamos escrever um pro-

grama para introduzir números e exibir o maior até o momento.

```
10 PRINT "NÚMERO", "MAIOR ATÉ AGORA"
20 INPUT A
30 LET MAIOR = A
40 PRINT A, MAIOR
50 INPUT A
60 IF MAIOR  < A THEN LET MAIOR = A
70 GOTO 40
```

A parte crucial está na linha 60, que atualiza o MAIOR se seu antigo valor foi menor que o novo número introduzido em A.

> (M maiúsculo) significa "é maior que" e é semelhante a <, porém invertido. Você pode distingui-los, lembrando-se de que a ponta aguda está voltada para o número menor.

< = (SHIFT R — não digite < seguido por =) significa "é menor ou igual a", assim ele é, igual a < exceto que este pode ser usado mesmo que os dois números sejam iguais: assim 2 < = 2 funciona, porém 2 < não.

> = (SHIFT Y) significa "é maior que ou igual a" e é similar a >.

<> (SHIFT T) significa "não é igual a", oposto, em termos de significado, a =.

Normalmente os matemáticos escrevem < = e <> como $\leq$, $\geq$ e $\neq$. Também "2<3<4" para significar "2 < 3 e 3 < 4", mas isto não é possível em BASIC.

Essas relações podem ser contornadas através do uso de operações lógicas AND, OR e NOT.

Uma relação AND outra relação

funciona sempre que ambas forem verdadeiras.

Uma relação OR outra relação

funciona sempre que uma das duas ou ambas relações forem verdadeiras.

NOT relação

funciona sempre que a relação não funcionar.

Expressões lógicas podem ser feitas com relações e AND, OR e NOT, da mesma forma que as expressões numéricas podem utilizar números e +, - e assim por diante. Você pode até mesmo usar parênteses se for o caso. NOT tem prioridade 4; AND, 3 e OR, 2.

Para ilustrar, limpe o computador e experimente esse programa, acrescido do programa que está no início do capítulo:

```
10 PRINT "ESTOU PENSANDO EM A"
20 PRINT "NÚMERO DE 1 A 5"
```

```
30 PRINT "VOCÊ PODE ADIVINHAR?"
35 LET T = 0
40 LET A = INT (RND*5) + 1
50 INPUT B
55 LET T = T + 1
60 IF A <> B AND T >= 3 THEN GOTO 100
65 IF A = B THEN GOTO 90
70 PRINT "ADIVINHE NOVAMENTE"
80 GOTO 50
90 PRINT "CORRETO"
95 STOP
100 PRINT "LAMENTO, O JOGO TERMINOU"
110 PRINT "VOCÊ NÃO CONSEGUIU A RESPOSTA"
120 PRINT "TRÊS PALPITES"
130 PRINT "A RESPOSTA ERA"; A
```

Observe a informação STOP na linha 95. De outra forma, quando o programa atingisse a linha 90 (como a resposta não estava correta), ele então seguiria para 100 e diria que a resposta correta não fora dada. No mínimo, tal fato causaria confusão.

Finalmente podemos comparar não apenas números, mas também cadeias de caracteres. Vimos como =, <, >, <>, <= e >= funcionam quando comparando números.

O que "menos que" significa para a cadeia de caracteres? Uma coisa que ele não significa é "mais curto que", portanto, não cometa este erro. Distinguimos uma cadeia de caracteres como sendo menos que outra se esta vier primeiro e em ordem alfabética; assim:

```
"SMITH"      <      "SMYTHE"
"SMYTHE"     >      "SMITH"
"BILLION"    <      "MILLION"
"DOLLAR"     <      "POUND"
```

todas são verdadeiras <= significa "é menos que ou igual a", e assim por diante, da mesma maneira em relação a números.

OBSERVAÇÃO: Em algumas versões do BASIC (não no AS-1000) a informação IF pode assumir a forma

```
IF condição THEN número da linha
```

significa o mesmo que

IF condição THEN GOTO número da linha

Você deve usar THEN no BASIC do AS-1000.

Resumo:

Operações: -, < , > ,<=,>=,<>, AND, OR

Funções: NOT

Exercícios

1.<>e = são opostos no sentido de que NOT A=B é o mesmo que A<>B e

NOT A<>B é o mesmo que A=B

Demonstre a si mesmo que <e>= , e > e<= são opostos da mesma forma, de modo que você sempre possa remover NOT da frente da relação, trocando-a.

Da mesma maneira,

NOT (uma primeira expressão lógica, AND uma segunda expressão)

é o mesmo que

NOT (a primeira) OR NOT (a segunda) e

NOT (uma primeira expressão lógica e OR uma segunda)

é o mesmo que

NOT (a primeira) AND NOT (a segunda).

Assim usando o NOT, você poderá operar com ele entre parênteses, quando eventualmente for aplicado numa relação, então você poderá dispor dele. Portanto, logicamente falando, NOT é desnecessário; porém você pode ainda considerar que usando-o, o programa se tornará mais claro.

2. Em BASIC, algumas vezes, podem aparecer frases diferentes do inglês. Considere, por exemplo, a preposição "se A não é igual a B ou C".

"IF A<> B OR C", nem "IF A<> B OR A<> C"

Não se preocupe se você não entender os exercícios 3, 4 e 5. Os pontos abordados neles são bastantes avançados.

3. (Pule este, a menos que você conheça BASIC perfeitamente).

Tente

PRINT 1 = 2,1<> 2

o que você pode esperar que apresente um erro de sintaxe. De fato, até onde diz respeito ao computador, valor lógico é algo que não existe.

(i) =, < , > ,<= ,>= e<> são todas operações binárias / submetidas a valor numérico, com prioridade 5. O resultado é 1 (para verdadeiro) se a relação funciona , e 0 (para .....

falso) se ela não funciona.

(ii) em

IF condição THEN informação

a condição pode, na realidade, ser qualquer expressão numérica. Se seu valor for Ø, então é considerado como falso e qualquer outro valor como verdadeiro. Assim, a informação IF significa exatamente o mesmo que

IF condição <> Ø THEN informação

(iii) AND, OR e NOT são também operações submetidas a valor numérico.

X AND Y têm valor	X se Y não for zero (considerada verdadeiro)
	Ø se Y for zero (considerada falsa)
X OR Y têm valor	1 se Y não for zero
	X se Y for zero
e	Ø se X não for zero
NOT X têm valor	1 se X for zero

Tendo isso em mente, leia de novo o capítulo, certificando-se de que tudo isso funciona.

Nas expressões X AND Y, X OR Y e NOT X, X e Y, cada uma normalmente tornará o valor Ø ou 1, para falso ou verdadeiro. Idealize as dez combinações diferentes e veja se elas realizam o que você espera que façam AND, OR e NOT.

4. Tente este programa:

```
1Ø INPUT A
2Ø INPUT B
3Ø PRINT (A AND A >= B) + (B AND A < B)
4Ø GOTO 1Ø
```

Cada vez ele imprime o maior dos dois números A e B. Por que?

Convença-se de que você pode pensar em

X AND Y

como significando

'X se Y (de outra forma o resultado é Ø)'

e

X OR Y

como significando

'X a menos que Y (caso em que o resultado é 1)'

Uma expressão usando AND e OR neste caso é chamada de expressão condicional. Um exemplo usando OR poderia ser

```
LET PREÇO DE VAREJO = PREÇO SEM TAXA * (1,Ø5 OR V$ = "NÃO-TRIBUTÁVEL").
```

Agora note que AND tende à adição (porque seu valor neutro é Ø), e OR tende à multiplicação (porque seu valor neutro é 1).

5. Você pode também fazer expressões condicionais com valor de cadeia de caracteres mas apenas usando AND.

```
X$ AND Y tem o valor      X$ se Y não for zero
                          "" se Y for zero
```

então isso significa "X$ se Y (também cadeia de caracteres vazia)".

Experimente esse programa, que introduz duas cadeias de caracteres e as põe em ordem alfabética.

```
1Ø INPUT A$
2Ø INPUT B$
3Ø IF A$ < = B$ THEN GOTO 7Ø
4Ø LET C$  = A$
5Ø LET A$  = B$
6Ø LET B$  = C$
7Ø PRINT  A$; " "; ( " < " AND A$ < B$) +
(" = " AND A$ = B$); ""; B$)
8Ø GOTO 1Ø
```

6. Tente este programa:

```
1Ø PRINT,"X"
2Ø STOP
3Ø PRINT "Y"
```

Quando você operá-lo ele exibirá "X" e parará com a indicação 9/2Ø. Agora digite.

```
CONT
```

Você pode esperar que isto funcione como "GOTO 2Ø", de modo que o computador iria apenas parar novamente, sem exibir "Y"; mas isso não seria muito útil. Assim, o programa é arranjado de tal maneira que, em indicações com o código 9 (informação STOP executada), o número de linha é acrescido de 1 para uma instrução CONT. Assim em nosso exemplo, "CONT" funciona como "GOTO 21" que uma vez não existindo linha entre 2Ø e 3Ø, funciona como "GOTO 3Ø").

7. Muitas versões do BASIC (mas não o BASIC do AS-1ØØØ) apresentam uma instrução ON, que assume a forma ON expressão numérica GOTO número da linha, número de linha, ..., número de linha.

Nesta, a expressão numérica é avaliada; suponha que seu valor seja n, então o efeito é o de que GOTO enésimo número de linha.

Por exemplo:

```
ON A GOTO 1ØØ, 2ØØ, 3ØØ, 4ØØ, 5ØØ
```

Aqui se A tem valor 2, então "GOTO 2ØØ" é executado. No BASIC AS-1ØØØ isto pode ser substituído por

GOTO 1ØØ* A

No caso de os números da linha não forem progredindo de cem em cem, imagine uma maneira de utilizar

GOTO uma expressão condicional em lugar da outra anterior.

Capítulo 8

SLOW E FAST

O AS-1ØØØ pode operar em duas velocidades — SLOW e FAST. Quando inicialmente ligado, trabalha de modo SLOW e pode computar e mostrar informações na tela simultaneamente. Este modo é ideal para os "gráficos que se movimentam" porque dá prioridade à manutenção da exibição e desenvolve a operação de computação durante os períodos em que a televisão está produzindo as partes vazias da figura na parte de cima e na parte de baixo da tela.

Contudo, um programa pode ser executado quatro vezes mais rápido, o que é feito dando prioridade à computação e apenas mantendo a figura quando nada mais existe para fazer. Para ver isto ocorrer, digite

FAST

Agora, sempre que você acionar uma tecla a tela piscará — isto ocorre porque o computador parou de exibir uma figura enquanto determinava qual a tecla que você acionou

Digite um programa, digamos,

```
1Ø FOR N = Ø TO 255
2Ø PRINT CHR$ N;
3Ø NEXT N
```

Este programa imprime o conjunto de caracteres.
Ver o programa da pag. 47

Observe que, conforme citamos de passagem no capítulo 4, CHR$ é uma função.. Acione SHIFT e NEWLINE para obter o curso F, então digite U para obter CHR$, que está abaixo do U.

Quando você operar esse programa a tela tomará uma tonalidade cinzenta até o fim do programa, quando a informação de PRINT aparecerá na tela.

A imagem também é mostrada durante uma imformação INPUT, enquanto o computador aguarda você digitar os dados de INPUT. Experimente esse programa:

```
1Ø INPUT A
2Ø PRINT A
3Ø GOTO 1Ø
```

A variável A refere-se a numerais

Para retornar ao método normal, computar e exibir, digite

SLOW

Normalmente será uma questão de opção se você deseja o método "compute e mostre" para continuidade na tela, ou o método "rápido" para velocidade; porém, de modo geral você usará o método "rápido".

(i) quando seu programa apresenta muitos cálculos numéricos, especialmente se ele não é impresso com freqüência o tempo não parecerá passar tão devagar se você puder ver saídas surgindo na tela com bastante freqüência.

(ii) quando você estiver digitando um programa extenso — você já deve ter notado que a listagem é refeita todas as vezes que você introduz uma nova linha de programa. Isto pode ser incômodo.

Você pode usar informações SLOW e FAST nos programas.

Por exemplo:

```
1Ø SLOW
2Ø FOR N = 1 TO 64
3Ø PRINT "A";
4Ø IF N = 32 THEN FAST
5Ø NEXT N
6Ø GOTO 1Ø
```

Resumo:

Instruções: FAST e SLOW.

Capítulo 9

SUB-ROTINAS

Algumas vezes, partes diferentes de seu programa terão tarefas semelhantes a realizar, de modo que você se encontrará digitando as mesmas linhas duas ou mais vezes. Isso não é necessário. Você pode digitar as linhas uma única vez, da forma conhecida como sub-rotina, e então usá-las ou chamá-las, em qualquer outro lugar do programa, sem ter que digitá-las novamente.

Para isso, use as instruções

GOSUB e RETURN

GOSUB n

em que n é o número da primeira linha da sub-rotina, como GOTO n, exceto pelo fato de o computador armazenar o número da linha da instrução GOSUB, para que possa retornar a ela após tê-la terminado. Isso é feito colocando o número da linha (o enderaço de retorno) sob o topo de uma pilha de instruções (o "stack" GOSUB)

RETURN

Toma o número da linha do topo do "stack" GOSUB e vai para a linha seguinte.

Exemplo:

```
10 PRINT "ESTE É O PROGRAMA PRINCIPAL"
20 GOSUB 1000
30 PRINT "E NOVAMENTE"
40 GOSUB 1000
50 PRINT "E ISSO É TUDO"
60 STOP
1000 REM "A SUB-ROTINA COMEÇA AQUI"
1010 PRINT "ESTA É A SUB-ROTINA"
1020 RETURN
```

Sem a instrução STOP na linha 60 o programa percorreria a sub-rotina e causaria o erro 7 quando encontrasse a instrução RETURN

Como exemplo, suponha que você deseja computar um programa para lidar com jardas, pés e polegadas. Você terá três variáveis Y, F e I (e talvez outras - Y1, F1, I1, e assim por diante). Aritmética é muito fácil. Primeiro opere as quantidades em jardas, pés e polegadas — por exemplo, para somar duas quantidades de distância, você soma as polegadas, soma os pés e soma as jardas; para multiplicar a distância você multiplica as polegadas, multiplica os pés e multiplica as jardas. Então converte as quantidades para a forma correta, de forma que as polegadas estejam entre 0 e 12 e os pés entre 0 e 3. Esse último estágio é comum para todas as operações, de modo que podemos realizá-lo em sub-rotina.

Deixando de lado, por um momento, a noção de sub-rotina, tente, por si mesmo, desenvolver o programa. Dados valores arbitrários para jarda, pé e polegada, como você os converteria para números corretos de jarda, pé e polegada?

O que logo virá à mente é algo como 1Y..4F..14I, que você preten

de converter para 2Y..2F..2I. Isso não é tão difícil, mas suponha que você tenha números negativos. Vc`temos aos nossos valores iniciais 1Y..4F..14I; na forma negativa eles são -1Y..-4F..-14I, que quais devem ser convertidos para -2Y..-2F..-2I. E a respeito de frações? Se você dividir 3Y..1F..7I por dois, você obtém 1,5Y..Ø,5..3,5I; e apesar do valor em pés estar entre Ø e 12 e o cem jardas ser 1,5, certamente esses valores não serão tão bons quanto 1Y..2F..3,5I. Tende encontrar respostas para tudo isso e use essas respostas em um programa de computador, antes de continuar a leitura desse manual.

Aqui está uma solução obtida através de funções, as quais serão descritas mais detalhadamente no capítulo 15. No capítulo 5 ensina-se como programá-las.

```
1ØØØ REM SUB-ROTINA PARA CONVERTER Y, F, I À FORMA NORMAL EM JARDAS, PÉS E POLEGADAS
1Ø1Ø LET I = 36*Y+12*F+I
1Ø2Ø REM AGORA TUDO ESTÁ EM POLEGADAS
1Ø3Ø LET E = SGN I
1Ø4Ø LET I = ABS I
1Ø5Ø REM TRABALHAMOS COM I POSITIVA, MANTENDO SEU SINAL EM E
1Ø6Ø LET F = INT (I/12)
1Ø7Ø LET I = (I-12*F) *E
1Ø8Ø LET Y = INT(F/3) *E
1Ø9Ø LET F = F*E-3*Y
11ØØ RETURN
```

Isto, por si só, não se apresenta muito útil, porque não há programa para estabelecer previamente Y, F, e I ou para fazer algo com eles posteriormente. Digite esse programa principal e também uma outra sub-rotina, para imprimir Y, F e I.

```
1Ø INPUT Y
2Ø INPUT F
3Ø INPUT I
4Ø GOSUB 2ØØØ
45 REM IMPRIMA OS VALORES
5Ø PRINT TAB 12; "=";
6Ø GOSUB 1ØØØ
65 REM O AJUSTE
7Ø GOSUB 2ØØØ
75 REM IMPRIMA OS VALORES
```

```
8Ø PRINT

9Ø GOTO 1Ø

2ØØØ REM SUB-ROTINA PARA IMPRIMIR Y, F, I

2Ø1Ø PRINT "";Y;"Y..";F;"F..";I;"I";

2Ø2Ø RETURN
```

Não há dúvida de que armazenamos os valores através do uso da sub-rotina de impressão na linha 2ØØØ, mas a sub-rotina de conversão, na verdade, torna o programa mais extenso através de um GOSUB e de um RETURN. Porém, a extensão do programa não é a única coisa a ser considerada. Usadas habilmente, as sub-rotinas podem tornar os programas mais fáceis de serem compreendidos.

O principal programa é tornado mais simples pelo fato de usar instruções mais poderosas. Cada GOSUB representa algumas instruções BASIC complicadas. Mas você pode esquecer isso, pois apenas o resultado importa. Por isso mesmo é muito mais fácil seguir a estrutura principal do programa.

Por outro lado, as sub-rotinas são simplificadas por razões muito diferentes, especificamente porque elas são mais curtas. Elas ainda usam instruções LET e PRINT mas têm de cumprir apensas parte da tarefa, sendo assim mais fáceis de escrever.

A habilidade está em escolher o nível — ou níveis — onde escrever as sub-rotinas. Elas devem ser grandes o suficiente para produzir um impacto significante sobre o programa principal e ainda pequenas o suficiente para serem significativamente mais fáceis de escrever do que um programa completo sem sub-rotinas. Estes exemplos (não recomendados) servem para ilustrar:

Primeiro:

```
1Ø GOSUB 1ØØØ

2Ø GOTO 1Ø

1ØØØ INPUT Y

1Ø1Ø INPUT F

1Ø2Ø INPUT I

1Ø3Ø PRINT "";Y;"Y..";F;"F..";I;"I";TAB12;"=";

1Ø4Ø LET I=36*Y+12*F+I
  :   :
  :   :

2ØØØ RETURN
```

E segundo:

```
1Ø GOSUB 1Ø1Ø

2Ø GOSUB 1Ø2Ø
```

```
30 GOSUB 1030
40 GOSUB 1040
50 GOSUB 1050
 :   :
 :   :
300 GOTO 10
1010 INPUT Y
1015 RETURN
1020 INPUT F
1025 RETURN
1030 INPUT I
1035 RETURN
1040 PRINT "";Y;"Y..";F;"F..";I;"I";TAB12;"=";
1045 RETURN
1050 LET I=36*Y+12*F+I
1055 RETURN
  :    :
  :    :
```

O primeiro, com sua única e poderosa sub-rotina e o segundo, com suas sub-rotinas simples e em uma certa quantidade, demonstram extremos amplamente opostos, mas de igual futilidade.

Uma sub-rotina pode chamar outra, ou até ela mesma (uma sub-rotina que chama a si mesma é chamada de recursiva); portanto, não tenha receio de ter várias camadas.

Resumo

Instruções: GOSUB, RETURN.

Exercícios

1. O programa exemplificativo é virtualmente um calculador de distância universal. Como você o usaria?

(i) para converter jardas e polegadas em jardas, pés e polegadas?

(ii) para converter metros em jardas e pés?

(iii) para encontrar frações de uma jarda? (exemplo: um terço de uma jarda, ou um pe?)

Acescente uma linha para arredondar as polegadas às mais próximas de seus valores.

2. Acrescente duas instruções ao programa

4 LET CONVERTA = 1ØØØ

7 LET YFIPRINT = 2ØØØ

e mude

GOSUB 1ØØØ para GOSUB CONVERTA

GOSUB 2ØØØ para GOSUB YFUPRINT

Isso funciona exatamente como você esperava. De fato, o número de linha em GOSUB (ou GOTO ou RUN) pode ser qualquer expressão numérica. (Isto pode não funcionar nos computadores que não sejam o AS-1ØØØ, porque ele não é um BASIC padronizado).

Este tipo de coisa pode fazer maravilhas em prol da clareza de seus programas.

3. Reescreva o programa principal do exemplo para fazer algo deveras diferente, mas usando ainda a mesma sub-rotina.

4. ..GOSUB n

..RETURN

em linhas consecutivas podem ser substituídos por

..GOTO n

Por quê?

5. Uma sub-rotina pode ter vários pontos de entrada. Por exemplo, devido à forma pela qual nosso principal programa as usa, com GOSUB 1ØØØ, seguido imediatamente por GOSUB 2ØØØ, podemos substituir nossas duas sub-rotinas por uma sub-rotina grande que ajuste Y, F e I e então, imprimi-las. Ela teria dois pontos de entrada: um no início, para toda a sub-rotina; e um outro mais adiante, apenas para a parte da impressão.

Desenvolva os arranjos necessários.

6. Execute este programa.

1Ø GOSUB 2Ø

2Ø GOSUB 1Ø

Os endereços de retorno são postos no "stack" GOSUB em quantidades, mas nunca são retirados novamente. Eventualmente não há mais espaço para o que quer que seja no computador. O programa então pára com indicação de erro 4 (veja relatórios de códigos).

CÓDIGOS (INDICADORES - PAG. 112)

Você pode ter dificuldades em retirar aquilo que você não precisa sem perder tudo o que está gravado, mas uma das soluções seguintes funcionará:

(i) Cancele as duas instruções GOSUB

(ii) Acrescente duas novas linhas

11 RETURN

21 RETURN

(iii) Digite

RETURN

Os endereços de retorno serão listados até que você obtenha erro 7.

(iv) Altere o programa de modo que a mesma coisa não ocorra novamente.

Funcionou? Como?

Capítulo 10

QUANDO A MEMÓRIA DO COMPUTADOR SE ESGOTA

O AS-1ØØØ tem armazenagem interna limitada e não é difícil esgotá-la. A melhor indicação de que isso ocorreu é normalmente uma informação de erro 4, porém podem ocorrer outras coisas e algumas dessas são mais estranhas ainda.

O arquivo de exibição - A área dentro do computador onde estão armazenadas as imagens de televisão - é projetada de modo a tomar es-- paço apenas para aquilo que foi impresso até aquele momento: uma linha no vídeo consiste de até 32 caracteres e de um carater NEWLINE. Isso / significa que você pode usar o espaço até o fim, imprimindo alguma coisa, e o tempo mais óbvio para fazer isso é enquanto está fazendo uma listagem. Digite

```
NEW
DIM A (3069)
1Ø FOR I=1 TO 15
2Ø PRINT I
```

(A instrução DIM então delimitada espaço na memória do computador, como será vista no capítulo 17. No momento o estamos usando como um meio rápido de (esgotar) a memória para a ilustração desse capítulo. Digite e não se preocupe com a memória).

Aqui está a primeira surpresa: a linha 1Ø desaparece da listagem, tem que incluir a linha 2Ø e não há espaço para as duas. Agora digite.

```
3Ø NEXT I
```

Novamente só há espaço para a linha 3Ø na listagem.

Agora digite.

```
4Ø REM X (sem NEWLINE)
```

e verá a linha 3Ø desaparecer a linha 4Ø passar para a parte de cima da tela. Ela não entrou no progrma - Você ainda tem o cursor L reverso e pode movê-lo. Tudo o que você viu é algum mecanismo obscuro que dá à metade inferior da tela, 24 linhas para dar prioridade sobre a metade superior. Agora digite

```
XXXXXX (ainda sem NEWLINE)
```

e o cursor desaparecerá - não há espaço para exibi-lo. Digite um outro X, sem NEWLINE, e um dos Xs desaparecerá. Agora digite NEWLINE. Tudo desaparecerá, mas o programa ainda está no computador, o que pode ser comprovado eliminando a linha 1Ø e usando ↓ e ↑ . Agora digite

```
1Ø FOR I = 1 TO 15
```

Novamente será movido para o topo da tela como a linha 4Ø fez . Mas quando digitar NEWLINE ele não será introduzido apesar de não haver mensagem de erro ou indicador reverso S para dizer que não há nada errado. Esse é o resultado de não haver espaço para checar a sintaxe de uma linha. Isso geralmente ocorre apenas com linhas que contém um

número (que não sejam linhas de número que estão no início.

A solução é arranjar espaço de alguma maneira, porém, primeiro apague a linha 1Ø, que não será inserida. Acione EDIT: a tela ficará em branco porque não há espaço para retirar a linha.

Acione NEWLINE e terá de volta, a parte da listagem. Agora apague a linha 4Ø (que de qualquer modo você realmente não precisa dela) digitando

4Ø (e NEWLINE)

Agora tente digitar a linha 1Ø novamente — Ela ainda não será introduzida. Apague-a de novo. Você ainda tem que encontrar espaço extra em algum lugar. Tenha em mente que a razão da linha 1Ø ter sido rejeitada provavelmente a falta de espaço para checar a síntese dos dois números (1 e 15): logo se você apagar a linha 2Ø no programa, você pode conseguir espaço para introduzir a linha 1Ø e ainda ter espaço para reintroduzir a linha 2Ø (que não contém número) posteriormente. Tente isso. Digite.

```
2Ø

1Ø FOR I=1 TO 15

2Ø PRINT I
```

e o programa será introduzido corretamente.

Digite

```
GOTO 1Ø
```

e novamente você verá que essa linha será rejeitada porque sua sintaxe não pode ser checada; todavia, se você apaga e digita

```
RUN
```

dará certo. (RUN limpa conseguindo bastante espaço).

Agora digite o mesmo que antes de NEW até a linha 3Ø, e em seguida

```
4Ø REM XXXXXXXXXXX
```

(11Xs), que terminará como 4Ø RE. Quando você pressiona NEWLINE, a listagem consistirá apenas da linha 3Ø, e na verdade a linha 4Ø terá sido completamente perdida. Isso ocorre simplesmente porque ela é muito longa para se adequar ao programa. O efeito será ainda pior quando a linha for uma versão aumentada de uma linha que já está no programa, pois você perderá tanto a linha que já estava no programa quanto a nova linha que a substituiria.

RESUMINDO:

1. Se a listagem é mostrada apenas parcialmente ou os sinais começam a saltar, o espaço está se esgotando.

2. Se parece não surtir nenhum efeito no fim de uma linha, provavelmente não há mais espaço para lidar com um número. Apague a linha, usando EDIT-NEWLINE ou RUBOUT.

3. NEWLINE pode perder uma linha completa.

Para todos esses problemas a solução é a mesma: Não entre em pânico e procure por algum espaço extra.

A primeira coisa a ser considerada é CLEAR. Se você tem variáveis e não se incomoda de perder alguma delas, é isso que deve ser feito.

Se isso fracassar, procure por instruções desnecessárias no programa, como instruções REM, por exemplo e apague algumas delas.

Resumo

Quando a memória se esgota pode ocorrer coisas estranhas, porém, elas normalmente não são fatais.

Capítulo 11

Matemática com o AS-1ØØØ

Ligue o computador. Agora você pode usá-lo como uma calculadora matemática, ao longo das linhas descritas no Capítulo 3: digite PRINT, em seguida, qualquer problema que você queira resolver e depois, NEWLINE

Além de adicionar, o AS-1ØØØ pode também subtrair, multiplicar (usando um asterisco * ao invés do sinal de multiplicar comum — isso é muito comum em computadores) e dividir (usando / em lugar de ÷). Tente pôr isso em prática.

+, -, * e / são operações, e os números que eles operam são seus operandos.

O computador pode também elevar um número à potência de outro usando a operação ** (SHIFT H): digite

```
PRINT 2**3
```

e obterá a resposta 8 (2^3, ou 2 ao cubo).

O AS-1ØØØ computará combinações de operações. Por exemplo

```
PRINT 2Ø-2*3**2+4/2*3
```

terá como resposta 8. O computador não executa as operações na ordem em que elas se encontram. Primeiro ele opera as potências (**) da esquerda para a direita; em seguida todas as multiplicações e divisões (* e /) também da esquerda para a direita; e então opera as adições e as subtrações (+ ou -), da mesma forma que as outras, da esquerda para a direita. Desse modo o nosso exemplo é operado nos seguintes estágios:

```
2Ø-2*3**2+4/2*3      primeiramente as potências
2Ø-2*9   +4/2*3
                     em seguida as multiplicações e as divisões
2Ø-18    +4/2*3
2Ø-18    +2  *3
2Ø-18    +6
                     e então as adições e as subtrações
2        +6
8
```

Isso é estabelecido dando a cada operação uma prioridade, um número entre 1 e 16. As operações com maior prioridade são realizadas em primeiro lugar e operações com prioridades iguais são realizadas da esquerda para a direita.

```
**      tem prioridade 1Ø
* e /   têm prioridade 8
+ e -   têm prioridade 6
```

Quando - é usado para tornar algum número negativo, como quando

você escreve -1, ele tem prioridade 9. Isso é o unário menos em oposição ao binário menos em 3-1. (Uma operação unária tem um operando, enquanto que uma operação binária tem dois operandos). Observe que no AS-1ØØØ você não pode usar + como uma operação unária.

Essa ordem é absolutamente rígida, porém você pode escapar dela usando parênteses: Tudo o que aparece entre parênteses é operado em primeiro lugar e em seguida considerado como um único número, então

```
PRINT 3*2+2
```

provê a resposta 6+2 = 8, porém

```
PRINT 3*(2+2)
```

provê a resposta 3*4 = 12.

Uma combinação como essa é denominada de expressão — Nesse caso, uma expressão aritmética ou numérica, pois a resposta é um número. Em geral, sempre que o computador estiver esperando que você lhe dê um número, você pode lhe dar uma expressão e ele obterá a resposta para esta expressão.

Você pode escrever números com pontos decimais, e pode também usar anotações específicas, como é muito comum em calculadoras de bolso. Nesse caso, após um número ordinário (com ou sem um ponto decimal), você pode escrever uma parte expoente consistindo da letra E, em seguida ser + ou -, e então o número sem ponto decimal. Aqui o E significa "*1Ø**, ("vezes 1Ø à potência de") de modo que

```
2.34EØ = 2.34*1Ø**Ø = 2.34

2.24E3 = 2.34*1Ø**3 = 234Ø

2.34E-2 = 2.34*1Ø**-2 = Ø,ØØ234  e assim por diante.
```

(Faça um teste imprimindo estas operações no AS-1ØØØ).

Você entenderá com mais facilidade se imaginar a parte expoente movendo o ponto decimal para a direita (para um expoente positivo) ou para a esquerda (para um expoente negativo).

Você pode também imprimir mais de um registro de uma só vez, separando-os por vírgulas (,) ou ponto e vírgulas (; em SHIFT X) Se você usar uma vírgula, o próximo número será exibido iniciando ou na margem da esquerda ou no meio da linha na 16a. coluna. Se você usar um ponto e vírgula, o próximo número será exibido seguindo imediatamente o último. Experimente

```
PRINT 1;2;3;4;5;6;7;8;9;1Ø

PRINT 1,2,3,4,5,6,7,8,9,1Ø
```

para ver a diferença. Você pode misturar vírgulas com ponto e vírgulas em uma mesma instrução PRINT.

Resumo

Funções: +,-,*,/,**

Expressões, notações científicas.

Exercícios

1. Experimente

```
PRINT 2.34EØ
PRINT 2.34E1
PRINT 2.34E2
```

e assim por diante até

```
PRINT 2.34E15
```

Você verá após um certo tempo o AS-1ØØØ também começa a usar notações científicas. Isso é devido ao fato de que ele nunca usa mais do que 14 espaços para escrever um número. Da mesma maneira, experimente

```
PRINT 2.34E-1
PRINT 2.34E-2
```

e assim por diante.

2. Tente

```
PRINT 1,,2,,3,,,4,,,,5
```

Uma vírgula sempre remeterá você para o próximo número.

Agora tente

```
PRINT 1;;2;;3;;;4;;;;5
```

Por que uma cadeia de ponto e vírgulas não é diferente de um único ponto e vírgula?

3. PRINT provê apenas 8 dígitos significantes. Experimente

```
PRINT 4294967295,4294967295 - 429E7
```

Isto prova que o computador pode aceitar todos os dígitos de 4294967295, mesmo que ele não esteja preparado para exibi-los de uma só vez.

4. Caso você tenha algumas tabelas de log, teste esta regra:

Elevar 1Ø à potência de um número é o mesmo que tomar o antilog daquele número.

Tente, por exemplo

```
PRINT 1Ø**Ø.3Ø1Ø
```

e observe o antilog de Ø.3Ø1Ø. Por que as respostas não são exatamente iguais?

5. O AS-1ØØØ usa pontos flutuantes aritméticos o que significa que ele toma os dígitos de um número em separado (sua mantissa e a posição do ponto (o expoente). A resposta nem sempre é exata, mesmo para números inteiros.

Digite

```
PRINT 1E1Ø+1-1E1Ø,1E1Ø-1E1Ø+1
```

Os números são mantidos em cerca de 9 1/2 dígitos exatos, então 1E1Ø é muito grande para ser posto exatamente à direita. A diferença (realmente cerca de 2) é maior do que 1, então os números 1E1Ø e 1E1Ø+1 parecem iguais para o computador.

Para um exemplo ainda mais específico digite

```
PRINT 5E9+1-5E9
```

Nesse caso a diferença em 5E9 é apenas de cerca de 1, então o 1 a ser adicionado, na verdade, é arredondado para 2. Aqui os números 5E9+1 e 5E9+2 parecem iguais para o computador.

O maior número inteiro que pode ser mantido completamente exato é $2^{32}-1$ (4.294.967.295). O AS-1ØØØ arredonda esse número para 8 dígitos significantes e então ele é mostrado como 4.294.967.300.

Capítulo 12

TÉCNICAS AVANÇADAS DE IMPRESSÃO

Você se lembrará que uma informação PRINT tem uma lista de itens, cada uma expressão (ou possivelmente nada) e que elas são separadas por vírgulas ou ponto e vírgulas. Existem mais dois ítens em PRINT usados para dizer ao computador não o que, mas onde imprimir. Por exemplo, PRINT AT 11, 16;"*"imprime um asterisco no meio da tela.

AT Linha, coluna

Muda a posição de PRINT (o local onde o próximo item será impresso) para a linha e coluna especificadas. As linhas são numeradas de Ø (no topo) a 21, e as colunas de Ø (à esquerda) a 31.

Coluna TAB

Muda a posição PRINT para a coluna especificada. Ela permanece na mesma linha ou se isso envolver retrocesso, muda para a próxima linha. Observe que o computador reduz o número de colunas do módulo 32 (ele divide por 32 e toma o restante); então TAB 33 é a mesma coisa que TAB 1. Por exemplo (imprima a página de índice encabeçada com 1 como sendo o número da página: imprimindo a página do índice)

```
PRINT TAB 12; "ÍNDICE"; AT 3,1;

"CAPÍTULO"; TAB 24; "PÁGINA"
```

Alguns pontos:

(i) Você deve normalmente usar ponto e vírgula com esses novos itens, como fizemos acima. Você pode usar vírgulas (ou nada, no fim da informação), porém isso significa que após você ter estabelecido cuidadosamente a posição de PRINT você mudou imediatamente ou para a borda esquerda ou para o centro da tela.

(ii) Ainda que AT e TAB não sejam funções você terá que acionar a tecla de função (SHIFT NEWLINE) para consegui-los.

ATENÇÃO

(iii) Você não pode imprimir as duas linhas inferiores (22 e 23) da tela. Referências para a "linha inferior" normalmente significam linha 21.

(iv) Você pode usar AT para colocar a posição PRINT mesmo onde já existir alguma coisa impressa; A impressão antiga será substituída pela nova.

Ainda existem duas informações relacionadas com PRINT, são elas: CLS e SCROLL.

CLS limpa a tela (e nada mais).

SCROLL muda a amostragem completa até uma linha (perdendo a linha superior) e muda a posição PRINT para o início da linha inferior.

Para ver como ele funciona, opere esse programa:

```
1Ø SCROLL

2Ø INPUT A$
```

```
30 PRINT A%
40 GOTO 10
```

Resumo

Itens PRINT: AT, TAB

Informações: CLS, Scroll

Exercícios

1. Tente operar isto:

```
10 FOR I = 0  TO  20
20 PRINT TAB 8*I;I;
30 NEXT I
```

Isso mostra o que significa números TAB serem reduzidos no módulo 32. Para um exemplo mais interessante, troque o 8 da linha 20 para 6.

Capítulo 13

O CONJUNTO DE CARACTERES

As letras, os dígitos, os sinais de pontuação e tudo o mais que podem aparecer em teclas são chamados caracteres, e formam o alfabeto, ou o conjunto de caracteres, que o AS-1000 usa. Muitos desses caracteres são símbolos únicos mas tem alguns, chamados sinais, que representam palavras completas, como PRINT, STOP, ** e assim por diante.

No total há 256 caracteres e cada um tem um código entre 0 e 255. no Apêndice aparece uma lista completa desses caracteres. Para converter entre códigos e caracteres existem duas funções, CODE e CHR$.

CODE - é aplicado a uma tecla e dá o código do primeiro caracter na tecla (ou 0 se a tecla está vazia).

CHR$ - É aplicado a um número e a única tecla do caracter cujo código é aquele número.

Esse programa imprime o conjunto de caracteres completo.

```
10 LET A = 0

20 PRINT CHR$ A;

30 LET A = A+1

40 IF A < 256 THEN GOTO 20
```

ver o programa da pg. 30

Na parte superior você pode ver os símbolos ", £, $ e assim por diante até Z; todos aparecem no teclado e podem ser digitados quando você tem o cursor L. Mas tarde você poderá ver os mesmos caracteres, porém em preto em branco (vídeo inverso); isso também é obtido através do teclado. Se você pressiona GRAPHICS (SHIFT 9), o cursor virá como G: isso significa modo de gráficos. Se você digita um símbolo, ele aparecerá em sua forma inversa de vídeo, e isso ocorrerá até que você pressione GRAPHICS de novo. RUBOUT terá seu significado comum. Tenha cuidado para não perder o cursos G juntos com todos caracteres de vídeo inversos que você acabou de digitar.

Após você ter experimentado um pouco você deve ainda ter o conjunto de caracteres na parte superior; se não, opere o programa de novo. no início existem espaços e os padrões pretos, brancos e cinzas; depois disso ainda existem mais onze símbolos que são chamados de símbolos gráficos e são usados para desenhar quadros. Você pode introduzí-los através do teclado, usando modo de gráficos de novo (salvo para espaço que é um símbolo comum usando o cursor L; o quadrado preto é espaço inverso). Use as 20 teclas que têm símbolos gráficos. Por exemplo, suponhamos que você queira o símbolo ▗ que é a tecla T. Pressione GRAPHICS para obter o cursor reverso G; e em seguida pressione o SHIFT T. Pela descrição anterior sobre o modo de gráficos você pode esperar obter um símbolo de vídeo inverso, porém o SHFT normalmente é <> , um sinal, e sinais não têm inverso: logo, ao invés disso, você obterá os símbolos gráficos ▚ ;

Aqui estão os 22 símbolos gráficos.

Aqui estão os 22 símbolos

Símbolo	Código	Como obter	Símbolo	Código	Como obter
	0	K ou L SPACE		128	G SPACE
	1	G SHIFT 1		129	G SHIFT Q
	2	G SHIFT 2		130	G SHIFT W
	3	G SHIFT 7		131	G SHIFT 6
	4	G SHIFT 4		132	G SHIFT R
	5	G SHIFT 5		133	G SHIFT 8
	6	G SHIFT T		134	G SHIFT Y
	7	G SHIFT E		135	G SHIFT 3
	8	G SHIFT A		136	G SHIFT H
	9	G SHIFT D		137	G SHIFT G
	10	G SHIFT S		138	G SHIFT F

Agora olhe para o conjunto de caracteres novamente. Os sinais se encontram quase que claramente em dois blocos: um pequeno grupo de três (RND, INKEY$ e PI) depois de Z e um grupo maior (começando com a imagem Z citada, e continuando de AT até COPY).

Todo o resto dos caracteres parece ser? Esse é realmente o modo pelo qual eles são impressos; O verdadeiro ponto de interrogação está entre: e (. Dos outros, alguns são para caracteres de controle como ← , EDIT e NEWLINE, e o resto são para caracteres que não têm nenhum significado especial para o AS-1000.

Resumo

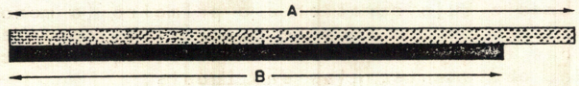
Funções: CODE, CHR$

Exercícios

1. Imagine o espaço para um símbolo dividido em quatro quartos: ⊞ Se cada quarto pode ser tanto preto quanto branco, existem 2*2*2*2 = 16 possibilidades. Encontre todas elas no conjunto de caracteres.

2. Imagine o espaço para um símbolo dividido em dois, horizontalmente: ⊟ . Se cada metade tanto pode ser preta, quanto branca ou cinza , existem 3*3 = 9 possibilidades. Encontre-as.

3. Os caracteres no exercício 2 são projetados para serem usados em mapas de barra horizontais usando duas cores, preto e branco. Faça um programa que introduz dois números A e B (ambos entre Ø e 32) e desenhe um gráfico para elas.



Você terá que iniciar imprimindo " ■ "; em seguida troque ou " ▩ " ou " ▄ ", dependendo se A é maior ou menor que B.

O que fará o seu programa se A e B não são números inteiros? Ou se eles não estão entre Ø e 32? Um bom programa - "usuário amigável" é o termo apropriado - fará algo sensível e útil.

4. Existem dois caracteres diferentes, totalmente cinza, no teclado, em A e em H. Se você observá-los cuidadosamente verá que o caracter em H é semelhante a uma miniatura de tabuleiro de xadrez, enquanto que o caracter em A é semelhante a um tabuleiro de xadrez paralelo. Se você imprime um próximo do outro você verá que eles não se juntam corretamente. O caracter em A normalmente se junta claramente a ▩ e ▄ (em S e em D), enquanto que o caracter em H se junta claramente a ■ e ■ (em F e em G).

5. Processe esse programa:

```
1Ø INPUT A
2Ø PRINT CHR$ A;
3Ø GOTO 1Ø
```

Qualquer nº de 1 a 255 imprime os caracteres correspondentes.

Se você for experimentá-lo você descobrirá que para CHR$, A é arredondado para o número inteiro mais próximo. Se A não está entre Ø e 255, o programa para com a informação B, "número fora de limite".

6. Usando os códigos dos caracteres podemos estender o conceito de "ordem alfabética" para enquadrar teclas contendo qualquer caracter, não apenas letras. Se ao invés de pensarmos em termos do alfabeto comum de 26 letras usamos o alfabeto expandido de 256 caracteres na mesma ordem de seus códigos, o princípio é exatamente o mesmo. Por exemplo, essas teclas estão em ordem alfabética.

"ZACHARY"

"■ "

"(ASIDE)"

"123 TAXI SERVICE"

"AASVOGEL"

"AARDVARK"

"AA RDVARK"

Aqui está a regra. Primeiro, compare os primeiro caracteres nas duas cadeias. Se eles são diferentes, o código de um deles é menor do que o do outro, e a cadeia cujo caracter é o primeiro (a menor) de duas cadeias. Se eles são os mesmos compare os próximos caracteres. Se nesse processo uma das teclas vem antes da outra, aquela tecla é a primeira; de outro modo, obviamente, elas são iguais.

Digite de novo o programa no exercício 5 do Capítulo 7 (aquele que introduz duas teclas e as imprime em ordem) e as use para testar.

7. Esse programa imprime uma tela repleta de caracteres gráficos preto e branco aleatóreos.

```
10 LET A = INT (16*RND)
20 IF A > = 8 THEN TO LET A = A + 120
30 PRINT CHR$ A;
40 GOTO 10
```

(Como?)

Capítulo 14

GRÁFICOS

Aqui estão alguns dos aspectos mais interessantes do AS-1ØØØ; eles utilizam "pixels" (elementos de tela). A tela que você usa para exibição tem 22 linhas e 32 colunas, que somam 22*32=7Ø4 posições de caracteres, cada uma contendo 4 "pixels".

Um "pixel" é especificado por dois números, seus coordenadores. O primeiro, sua coordenada X, denota a distância dele em relação à última coluna esquerda (lembre-se, X é TRANSVERSAL); e o segundo, sua coordenada Y indica a sua distância em relação à parte inferior. Essas coordenadas são normalmente escritas como um par entre parênteses, então (Ø,Ø), (63,Ø), (Ø,43) e (63,43) são os cantos inferior esquerdo, inferior direito, superior esquerdo e superior direito.

A informação PLOT coordenada X, coordenada Y aparece no "pixel" com essas coordenadas, enquanto a informação UNPLOT coordenada X, coordenada Y a apaga.

Tente esse programa simples:

```
1Ø PLOT INT (RND*64), INT (RND*44)

2Ø INPUT A$

3Ø GOTO 1Ø
```

Isso assinala um ponto aleatório cada vez que você aciona NEWLINE.

Aqui está um programa mais útil. Ele assinala um gráfico da função SIN (uma FUNÇÃO seno) para valores entre 0 e 2 pi radianos.

```
1Ø FOR N=Ø TO 63

2Ø PLOT N, 22+2Ø*SIN (N/32*PI)

30 NEXT N
```

Observe que as coordenadas "pixel" são diferentes da linha e da coluna em um item AT. Você pode achar o diagrama no fim deste capítulo útil para operar coordenadas "pixel" e linhas e colunas de números.

Exercícios

1. Há três diferenças entre os números em um iten AT e coordenadas "pixel". Quais são elas?

Suponha que uma posição PRINT corresponde a AT L, C (para linha e coluna). Certifique-se de que os quatro "pixels" naquela posição têm coordenadas x 2*C ou 2*C+1, e coordenadas Y 2*(21-L) ou 2*(21-L) + 1. (Olhe o diagrama).

2. Altere o programa simples de modo que ele primeiramente encha a tela com preto (um quadrado preto é um espaço inverso de vídeo), e em seguida desmarque pontos aleatórios.

3. Modifique o programa do gráfico SIN de modo que antes de traçar o próprio gráfico ele imprima uma linha horizontal de "—"s para um eixo X e uma linha vertical de "/" para um eixo Y.

4. Faça programas para traçar gráficos com mais funções: por exemplo, COS, EXP, LN, ATN, INT e assim por diante. Você tem que estar certo de que os gráficos se enquadram na tela, então você precisa considerar

(i) Sobre que distância você vai tomar as funções (correspondendo à distância de 0 a 2 pi radianos para o gráfico SIN).

(ii) Em que lugar da tela colocar o eixo X (correspondendo a 22 na linha 20 no SIN do gráfico do programa).

(iii) Como fazer a escala do eixo Y do gráfico (correspondendo a 20 na linha 20 do SIN do gráfico do programa).

Você verá que COS é o mais fácil - é exatamente como SIN.

5. Opere isto:

```
10 PLOT 21,21
20 PRINT "HEAVY QUOTES"
30 PLOT 46,21
```

Este programa
endireita no video
a expressão entre aspas " "

PLOT muda a posição PRINT para o primeiro espaço após a localização de PLOT. (UNPLOT também faz isso).

6. Essa subrotina desenha uma linha uniformemente reta do "pixel" (A, B) ao "pixel" (C,D). Use-a como parte de algum programa principal que provê os valores A,B,C,D.

```
1000 LET U=C-A
1005 REM U MOSTRA QUANTOS PASSOS ALEM PRECISAMOS IR
1010 LET V=D-B
1015 REM V MOSTRA QUANTOS PASSOS ACIMA
1020 LET D1X=SGN U
1030 LET D1Y=SGN V
1035 REM (D1X,D1Y) É UM ÚNICO PASSO NUMA DIREÇÃO DIAGONAL
1040 LET D2X=SGN U
1050 LET D2Y=0
1055 REM (D2X,D2Y) É UM ÚNICO PASSO À ESQUERDA OU DIREITA
1060 LET M=ABS U
1070 LET N=ABS V
1080 IF  M>N THEN GOTO 1230
1090 LET D2X=0
1100 LET D2Y=SGN V
1105 REM AGORA (D2X,D2Y) É UM ÚNICO PASSO PARA CIMA OU PARA BAIXO
1110 LET M=ABS V
```

```
1120 LET N=ABS U

1130 REM M É O MAIOR DE ABS U E ABS V, N É O MENOR

1140 LET S=INT (M/2)

1145 REM QUEREMOS MOVER DE (A,B) PARA (C,D) EM M PASSOS USANDO N
PASSO N PARA CIMA-PARA BAIXO OU PASSOS D2 DIREITA-ESQUERDA, E
M-N PASSOS D1 NA DIAGONAL, DISTRIBUÍDOS TÃO EQUITATIVAMENTE QUAN
TO POSSÍVEL.

1150 FOR N=0 TO M

1160 PLOT A,B

1170 LET S=S+N

1180 IF S < M THEN GOTO 1230

1190 LET S=S-M

1200 LET A=A+D1X

1210 LET B=B+D1Y

1215 REM UM PASSO DIAGONAL

1220 GOTO 1250

1230 LET A=A+D2X

1240 LET B=B+D2Y

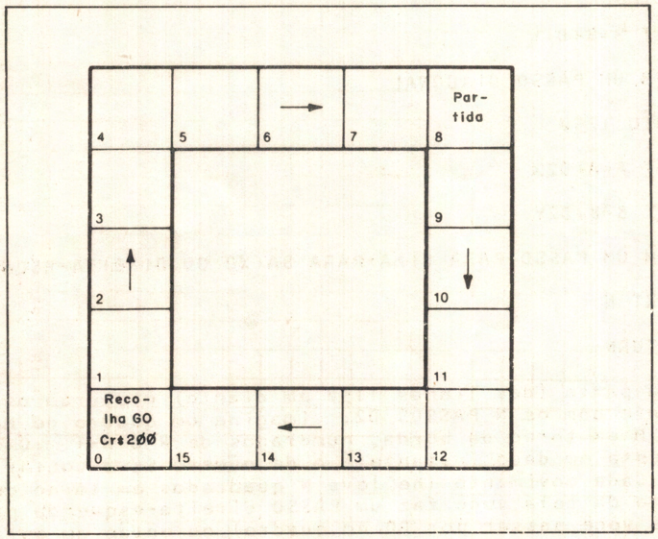
1245 REM UM PASSO PARA CIMA-PARA BAIXO OU DIREITA-ESQUERDA

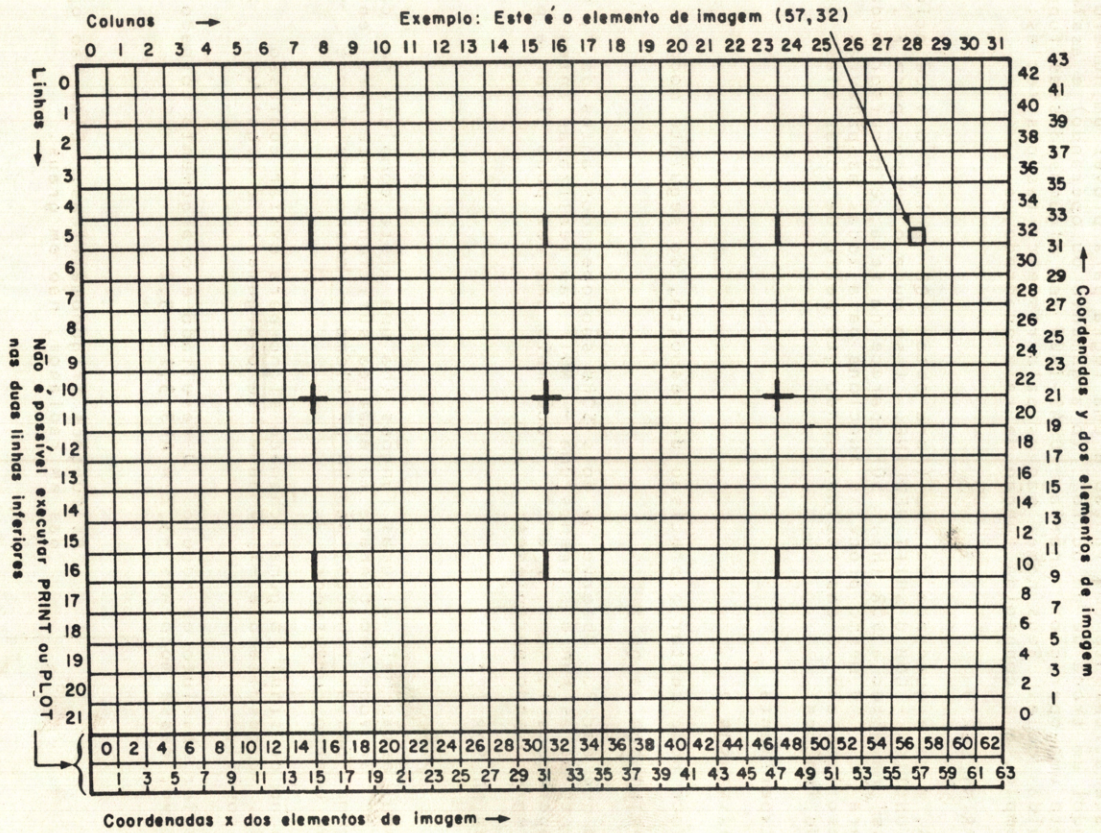
1250 NEXT N

1260 RETURN
```

A última parte (das linhas 1150 em diante) misturam os M-N PASSOS D1 uniformemente com os N PASSOS D2. Imagine um quadro de Monopólio / com quadrados M em torno da borda, numerados de 0 a M-1. O quadrado / no qual você esta em dado instante é o de número S; a contar do canto oposto a GO. Cada movimento lhe leva N quadrados em torno do quadro e na linha reta da tela você faz um PASSO direita-esquerda para cima-para baixo (se você passar por GO no quadro) ou então em estágio diagonal. Sendo que sua jornada total no quadro é de M*N PASSOS ou todo o percurso N vezes, você passa por GO N vezes e uniformemente espaçado em seus PASSOS M estão N PASSOS direita-esquerda/para cima-para baixo.

Ajuste o programa de modo que se um outro parâmetro, E, é 1, a linha é traçada em preto (como aqui), e se ele é 0, a linha é traçada em branco (usando unplot). Em seguida você pode apagar uma linha que você acabou de traçar desfazendo-a.


Par-
tida
8
4
5
6
7
3
9
2
10
1
11
Reco-
lha GO
Cr$2ØØ
0
15
14
13
12


Colunas →
Exemplo: Este é o elemento de imagem (57,32)
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31
Linhas →
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21
43 42 41 40 39 38 37 36 35 34 33 32 31 30 29 28 27 26 25 24 23 22 21 20 19 18 17 16 15 14 13 12 11 10 9 8 7 6 5 4 3 2 1 0
↑ Coordenadas y dos elementos de imagem
Não é possível executar PRINT ou PLOT nas duas linhas inferiores
0 2 4 6 8 10 12 14 16 18 20 22 24 26 28 30 32 34 36 38 40 42 44 46 48 50 52 54 56 58 60 62
1 3 5 7 9 11 13 15 17 19 21 23 25 27 29 31 33 35 37 39 41 43 45 47 49 51 53 55 57 59 61 63
Coordenadas x dos elementos de imagem →

Capítulo 15

FUNÇÕES

Matematicamente uma função é uma regra para prover um número (o resultado) em troca de um outro, (o argumento ou operando) e assim, é realmente uma operação unária. O AS-1ØØØ tem algumas dessas funções construídas nele. Seus nomes são as palavras escritas nas teclas. SQR, por exemplo, é a conhecida função de raiz quadrada, e

```
PRINT SQR 9
```

provê 3, a raiz quadrada de 9. (Para obter SQR primeiro você aciona a tecla FUNCTION - COM O SHIFT NEWLINE. Isso muda o cursor para F . Agora pressione a tecla SQR (H): SQR aparece na tela, e o cursor muda de novo para L . O mesmo processo funciona para todas as palavras que estão escritas abaixo das teclas, sendo que perto de todas elas estão nomes de funções).

Experimente

```
PRINT SQR 2
```

Você pode testar a precisão da resposta, do seguinte modo:

```
PRINT SQR 2*SQR 2
```

que tem que dar 2. Observe os dois SQRs são operados antes do *, e na verdade todas as funções (menos uma — NOT) são realizadas antes das cinco operações +, -, *, / e **. Você mais uma vez pode tirar a prova dessa regra usando parênteses:

```
PRINT SQR (2*2)
```

dá 2.

Aqui estão mais algumas funções (há uma lista completa no capítulo 21). Se você não conhece matemática suficientemente para entender algumas dessas funções, não há problema — mesmo assim você será capaz de usar o computador.

SGN função de sinal (algumas chamado signo para evitar confusão com SIN). O resultado é -1, Ø ou +1 se o argumento é negativo, zero ou positivo.

ABS O valor absoluto ou módulo. O resultado é o argumento tornado positivo, desse modo ABS -3,2 = ABS 3,2 = 3,2

SIN
COS
TAN
ASN arco de seno
ACS arco de coseno
ATN arco de tangente

} As funções trigonométricas. Elas são operadas em radianos, não em graus.

LN logaritmo natural (para base 2,718281828..., infinitamente)

EXP função exponencial

SQR raiz quadrada

INT parte inteira. Esse sempre arredonda o número. Desse modo INT 3,9 = 3 e INT -3,9 = -4. (um número inteiro é um número completo, podendo ser negativo).

PI π = 3,14159265..., a circunferência em polegadas de um círculo de uma polegada de diâmetro. PI não tem argumento.

RND RND também não tem argumento. Ele provê um número aleatório entre Ø (cujo valor ele pode tomar) e 1 (que não pode tomar o valor).

Toda essas funções, salvo PI e RND são operações unitárias com prioridade 11. (PI e RND são operações nulárias porque não têm operando).

As funções trigonométricas, assim como EXP, LN e SQR geralmente são calculadas com 8 dígitos de precisão.

RND e RAND: Ambas estão na mesma tecla, porém enquanto RND é uma função, RAND é uma palavra chave assim como PRINT. RAND é usado para controlar a aleatoriedade de RND.

Na verdade RND não é aleatório, pelo contrário, ele segue uma seqüência fixa de 65536 números que podem ser tão misturados que parecem ser aleatórios (RND é pseudo-aleatório). Você pode usar RAND para iniciar RND em um lugar definido na seqüência acionando RAND, seguido de um número entre 1 e 65535 e em seguida acionar NEWLINE. Não é tão importante saber onde um dado número inicia RND, assim como o mesmo número após RAND sempre iniciará RND no mesmo lugar. Por exemplo, digite

```
RAND 1    (e NEWLINE)
```

e em seguida

```
PRINT RND
```

e digite tudo isso várias vezes, um de cada vez. (Lembre-se de usar FUNCTION para obter RND). A resposta de RND sempre será Ø,ØØ22735596, uma seqüência não muito aleatórea.

```
RAND Ø
```

(Ou você pode abandonar o Ø) funciona ligeiramente diferente: ela determina onde iniciar RND de acordo com o tempo em que a televisão está ligada e isso seria realmente aleatório.

Observações: Em alguns dialetos de BASIC você pode sempre incluir o argumento de uma função entre colchetes. Esse não é o caso do BASIC do AS-1ØØØ

Resumo:

Informação: RAND

Funções: SGN, ABS, SIN, COS, TAN, ASN, ACS, ATN, LN, EXP, SQR, INT, PI, RND

Exercícios

1. Para obter logaritmos comuns (na base 10) aqueles que você procura na tábua de logaritmos, divida o logaritmo natural por LN 10. Por exemplo, para encontrar log 2.

   PRINT 1N 2/LN 10

   e obterá a resposta 0,301003.

   Tente fazer multiplicações e divisões usando logs, utilizando para isso o AS-1000 como um conjunto de tábuas de logaritmos. Confira as respostas usando * e /. O modo direto é mais preciso.

2. EXP e LN são funções inversas no sentido de que, se você aplica uma e em seguida a outra, você obterá de novo o número original. Por exemplo.

   LN EXP 2 = EXP LN 2 = 2

   O mesmo ocorre para SIN e ASN, para COS e ACS e para TAN e ATN. Você pode usar isto para testar a precisão do computador na operação dessas funções.

3. π radianos são 180°. Para converter de graus para radianos você divide por 180 e multiplica por π: então

   PRINT TAN (45/180*PI)

   obterá a tangente 45° (1)

   Para converter de radianos para graus divida por π e multiplique por 180.

4. Tente

   PRINT RND

   várias vezes para ver como as respostas variam. Você pode detectar um comportamento padrão?
   Como você usaria RND e INT para obter um número inteiro aleatório entre 1 e 6, para representar a o arremesso de um dado. (Resposta: INT (RND *6) +1).

5. Tente esta regra:

   Suponhamos que você escolha um número entre 1 e 872 e digite

   RAND e em seguida o número escolhido (e NEWLINE)

   O próximo valor de RND será

   (75 * (o número escolhido + 1) - 1)/65536

6. (Apenas para matemáticos).

   Seja p um número primo e (grande) e a um radical do módulo p. Então se b_i é o resto de um módulo p ($1 = b_i = p - 1$), a seqüência

$$\frac{b_i - 1}{p - 1}$$

é uma sequencia cíclica números distintos p-1 na faixa de Ø a 1 (exclusive 1). Se você escolher apropriadamente poderá fazê-los parecer razoavelmente aleatório.

65537 é um radical Mersenne. 2^{16}-1. Use isso seguindo a lei de Gauss de reciprocidade quadrática para mostrar que 75 é um radical arbitrário do módulo 65537.

O AS-1ØØØ usa p=65537 e a=75, e armazena algum b_i=1 na memória. A função RND envolve a substituição de b_i-1 na memória por b_{i+1}-1 obtendo o resultado (b_{i+1}-1) / (p-1). RAND n (com 1 < = n < =65535) torna b_i igual n+1.

7. INT sempre arredonda. Para arredondar para o número inteiro mais próximo adicione, primeiro, Ø,5.

   Por exemplo,

   INT (2,9 + Ø,5) = 3 INT (2,4 + Ø,5) = 2

   INT (-2,9 + Ø,5) = -3 INT (-2,4 + Ø,5) = -2

   Compare isto com as respostas que você obtém quando não adiciona Ø,5.

8. Tente

   PRINT PI, PI -3, Pi -3, 1, PI -3, 14, Pi -3, 141

   Isso mostra a precisão com que o computador armazena π.

Capítulo 16

TEMPO E MOVIMENTO

Algumas vezes você desejará que seu programa tenha uma extensão de tempo determinado. A instrução PAUSE será útil para esse fim.

PAUSE n

Pára a computação por n imagens da televisão (a 60 imagens por segundo). n pode ir até 32767, que dá uma parada de até 11 minutos; porém, n maior do que isso significa parada contínua.

Sempre que você quiser tornar uma pausa mais curta, basta pressionar uma tecla qualquer. (Lembre-se que um espaço ou uma £ também causarão uma parada). A tecla deverá ser pressionada após o início da pausa.

A tela piscará no fim da pausa.

Esse programa funciona como um ponteiro de segundos de um relógio (aqui apenas um único ponto na borda).

```
5 REM PRIMEIRO DESENHAMOS A FACE DO RELÓGIO
10 FOR N=1 TO 12
20 PRINT AT 10-10*COS (N/6*PI), 10+10*SIN (N/6*PI); N
30 NEXT N
35 REM AGORA LIGAMOS O RELÓGIO
40 FOR T=0 TO 10000
45 REM T É O TEMPO EM SEGUNDOS
50 LET A=T/30*PI
60 LET SX=21+18*SIN A
70 LET SY=22+18*COS A
200 PLOT SX, SY
300 PAUSE 42
310 UNPLOT SX, SY
400 NEXT T
```

Esse relógio funcionará por cerca de duas horas e quarenta e cinco minutos devido à linha 40, mas é muito fácil fazê-lo funcionar por mais tempo: observe como o tempo é controlado pela linha 300. Você pode esperar que a PAUSE 50 emita um sinal por segundo, mas a computação também precisa de tempo. A melhor maneira de fazer isso é por tentativa e erro, regulando o relógio do computador com o relógio verdadeiro e ajustando a linha 300 até que os dois relógios estejam marcando o mesmo horário. (Isso não pode ser feito com muita precisão, pois qualquer ajuste de uma imagem em um segundo é de 2% ou meia hora em um dia).

A função INKEY$ (que não tem argumento) lê o teclado. Se você es

tá pressionando uma tecla o resultado é o caracter que essa tecla representa no modo L; de outro modo o resultado será uma cadeia vazia. O controle de caracteres não tem seu efeito normal, o que ele provê é resultados como CHR$ 118 para NEWLINE - eles são impressos como '?'.

Experimente esse programa que funciona como uma máquina de escrever.

```
10 IF INKEY$ <> "" THEN GOTO 10

20 IF INKEY$ = "" THEN GOTO 20

30 PRINT INKEY$

40 GOTO 10
```

Aqui a linha 10 espera que você retire as mãos do teclado e a linha 20 que você pressione uma nova tecla.

Lembre-se que de forma diferente de INPUT, INKEY$ não espera por você. Então não digite NEWLINE. Por outro lado, se você não pretende digitar nada, você perdeu sua chance. Nesse programa, a linha 10 "espera por você", pela repetição GOTO 10, se nenhuma tecla for pressionada.

Exercícios:

1. O que acontece se você omitir a linha 10 no programa da máquina de escrever?

2. Por que você não pode digitar espaço ou £ no programa da máquina de escrever?

   Aqui está um programa modificado que provê um espaço quando você digita o cursor de movimento para a direita (SHIFT 8).

```
10 IF INKEY$ <> "" THEN GOTO 10

20 IF INKEY$ =  "" GOTO 20

30 LET A$=INKEY$

40 IF A$=CHR$ 115 THEN GOTO 110

90 PRINT A$

100 GOTO 10

110 PRINT "";

120 GOTO 10
```

   Observe que lemos INKEY$ em A$ na linha 30. Seria possível omitir isso e subtituir A$ por INKEY$ nas linhas 40 e 90, mas sempre haveria possibilidade de INKEY$ se mudar entre as linhas.

   Junte mais algum programa de maneira que você obtenha uma nova linha ao digitar NEWLINE (CHR$ 118).

3. Você pode também usar INKEY$ em conjunto com PAUSE como nesse programa alternativo de máquina de máquina de escrever.

```
10 PAUSE 40000

20 PRINT INKEY$

30 GOTO 10
```

Porque é essencial para fazer esse trabalho, que uma pausa não termine no momento em que você já está pressionando uma tecla no momento de seu início.

Esse processo tem a desvantagem de a tela piscar, porém no processo rápido esse é o único meio de fazê-lo. Quando você opera um programa no processo rápido, observe que o computador usa a pausa para exibir a imagem de televisão.

4. O programa seguinte faz o computador exibir um número que você (ou uma vítima inocente) terá que digitar em resposta. Para iniciar, você tem um segundo para digitá-lo mas caso você erre, obterá um tempo maior para o próximo número; enquanto que se você o fizer corretamente você obterá menos tempo para o próximo número. O Objetivo é fazer esse jogo no tempo mais curto possível e em seguida pressionar Q para encontrar o seu escore - quanto maior melhor.

```
10 LET T=60

15 REM T=NÚMERO DE IMAGENS POR VEZ - INICIALMENTE 60 POR SEGUNDO

20 LET A$=CHR$ INT (RND*10+CODE "0")

30 REM A$ É UM DÍGITO ALEATÓREO

40 PRINT A$

50 PAUSE T

60 LET B$=INKEY$

70 IF B$="Q" THEN GOTO 200

80 IF A$=B$  THEN GOTO 150

90 PRINT "RUIM"

100 LET T=T*1.1

110 GOTO 20

150 PRINT "OK"

160 LET T=T*0.9

170 GOTO 20

200 SCROLL

210 PRINT "SEU ESCORE ""; INT (500/T)
```

5. (Para divertir-se). Experimente esse:

```
10 IF INKEY$ = ""THEN GOTO 10
```

```
2Ø PRINT AT 11,8; "TECLADO ACIONADO"
3Ø IF INKEY$ <> "" THEN GOTO 3Ø
4Ø PRINT AT 11,14; ''
5Ø GOTO 1Ø
```

Capítulo 17

ARRAYS

Array é um conjunto de variáveis, ou de elementos, todos tendo o mesmo nome e distinguidos uns dos outros apenas por um número (o subscrito) escrito em parênteses após o nome. Por exemplo, o nome poderia ser A (como o controle de variáveis dos "loops" de FOR-NEXT, o nome de array tem que ser uma única letra), e doze variáveis seria A(1), A(2), e assim por diante até A(12).

Os elementos de um array são chamados de variáveis subscritas , sendo um oposto às variáveis simples com as quais você já está familiarizado.

Antes de poder usar um array você terá que reservar algum espaço no computador. Para tal você usa uma instrução DIM (para dimensão).

```
DIM A (12)
```

provê um array chamado A com uma dimensão 12 (isto é, existem 12 variáveis subscritas A(1),..., A (12))e inicia os 12 valores para Ø. Esse array também apaga qualquer array denominado A que tenha existido previamente. (Não uma variável simples. Um array e uma variável numérica simples com o mesmo nome podem coexistir e não há confusão entre os dois porque a variável do array sempre tem um subscrito).

O subscrito pode ser uma expressão numérica arbitrária, desse modo você pode escrever

```
1Ø FOR N=1 TO 12

2Ø PRINT A(N)

3Ø NEXT N
```

Você pode também estabelecer arrays com mais de uma dimensão. Em um array de duas dimensões são necessários números para especificar um dos elementos - de preferência igual à linha e à coluna de números para especificar uma posição do caracter na tela de televisão - desse modo o array de duas dimensões tem a forma de uma tabela. Como alternativa, se você imagina uma linha e uma coluna de números (de duas dimensões) como se referindo a uma página impressa, você poderia ter uma dimwnsão extra para os números da página. Mas é claro, estamos falando em arrays numéricos; os elementos não deveriam ser caracteres impressos como em um livro, mas sim números. Pense nos elementos de um array C de três dimensões comosendo especificado por C (número da página, linha de número, número da coluna).

Por exemplo, para estabelecer um array B de duas dimensões, com dimensão 3 e 6, você usa uma instrução DIM.

```
DIM B (3,6)
```

Então isso resulta em 3*6 = 18 variáveis subscritas

```
B(1,1), B(1,2), ..., B(1,6)

B(2,1), B(2,2), ..., B(2,6)

B(3,1), B(3,2), ..., B(3,6)
```

O mesmo princípio funciona para qualquer número de dimensões.

Apesar de você poder ter um número e um array com o mesmo nome, você não pode ter dois arrays com o mesmo nome, mesmo que eles tenham números de dimensão diferentes.

Também há cadeia de arrays. As cadeias em um array são diferentes das cadeias simples nas quais eles têm uma extensão fixa, e as denominações que elas recebem são denominações procrustianas. Um outro meio de pensar nessas cadeias é como arrays de um único caracter com uma dimensão extra. O nome da cadeia de um array é uma única letra seguida de $. Uma cadeia de arrays e uma cadeia de variável simples não podem ter o mesmo nome (diferente do que ocorre no caso dos números).

Então suponha que você quer um array A$ de cinco cadeias. Você terá que decidir a extensão dessas cadeias - supoonhamos que 1Ø caracteres para cada cadeia seja suficiente. Então você diz

```
DIM  A$ (5,1Ø)   (digite-o)
```

Isso estabelece um array de 5*1Ø caracteres, mas você pode pensar em cada série como sendo uma cadeia:

A$(1) =	A$(1,1)	A$(1,2)	...	A$(1,1Ø)
A$(2) =	A$(2,1)	A$(2,2)	...	A$(2,1Ø)
	.	.	.	.
	.	.	.	.
A$(5) =	A$(5,1)	A$(5,2)	...	A$(5,1Ø)

Se você atribui o mesmo número de subscritos (dois nesse caso) como se houvesse dimensões na instrução DIM, você obtém um único caracter; porém se você omitir o último você obtém uma cadeia de comprimento fixo. Então, por exemplo, A$(2,7) é o sétimo caracter na cadeia A$(2); usando a notação, fracionadora você poderia também escrever A$(2) (7). Agora digite

```
LET A$(2) = "123456789Ø" e

PRINT A$(2), A$(2,7)
```

você obtém

```
123456789Ø       7
```

Para o último subscrito (aquele que você pode omitir), você também pode ter uma parte, de modo que, por exemplo,

```
A$(2,4 TO 8) = A$(2) (4 TO 8) = "45678"
```

Lembre-se

Em uma cadeia de arrays todas as cadeias têm a mesma extensão , sendo essa extensão fixa.

A instrução DIM tem um número extra (o último) para especificar essa extensão.

Quando você escreve uma variável subscrita para um array de ca-

deia, você pode acrescentar um número extra ou uma fração, para corresponder com o número extra na instrução DIM.

Sumário

Arrays (o meio pelo qual o AS-1ØØØ manipula arrays de cadeia é ligeiramente não padronizado).

Instruções: DIM

Exercícios:

1. Estabeleça um array M$ de 12 cadeias no qual M$(1) é o nome do mês.

   (Uma pista: a instrução DIM será DIM M$(12,9)).

   Faça um teste imprimindo todos os M$(1) (use um loop). Digite

   ```
   PRINT "AGORA É O MÊS DE ";M$(5); "ING"; "QUANDO AS CRIANCINHAS ESTÃO BRINCANDO"
   ```

   O que você pode fazer a respeito daqueles espaços?

2. Você pode ter arrays de cadeia sem dimensões.

   Digite

   ```
   DIM A$(1Ø)
   ```

   e você descobrirá que A$ age do mesmo modo que uma variável de cadeia, exceto que ele sempre tem um comprimento de 1Ø, e a sua determinação é sempre procustiana.

3. READ, DATA e RESTORE

   A maioria dos computadores BASIC (não o BASIC do AS-1ØØØ) tem três instruções chamadas READ, DATA e RESTORE.

   Uma instrução DATA é a lista de expressões, e tomando todas as instruções DATA no programa você obterá uma longa lista de expressões, a data list (lista de DADOS).

   A instrução READ é usada para atribuir essas expressões, uma a uma a variáveis:

   ```
   READ X
   ```

   por exemplo, determina a expressão comum na lista de DADOS para a variável X e move para a próxima expressão para a próxima instrução READ.

   (RESTORE leva de novo ao início da lista de DADOS). Em teoria sempre é possível substituir instruções READ e DATA por instruções / LET; todavia, um dos maiores usos é para iniciar arrays, como nesse programa:

   ```
   5 REM  ESSE PROGRAMA NÃO FUNCIONARÁ NO BASIC DO AS-1ØØØ
   1Ø DIM  M$(12,3)
   2Ø FOR N=1 TO 12
   3Ø READ M$(N)
   ```

```
40 NEXT N

50 DATA "JAN"; "FEV", "MAR", "ABR"

60 DATA "MAI", "JUN", "JUL", "AGO"

70 DATA "SET", "OUT", "NOV", "DEZ"
```

Se você pretende operar esse programa apenas uma vez você terá que substituir a linha 30 por

```
30 INPUT M$ (N)
```

e então você não terá que digitar mais nada. Todavia, se você pretende armazenar o programa, certamente você não pretende digitar os meses todas as vezes que for operá-lo.

Sugerimos então que você use o seguinte processo:

(i) Inicie o array usando um "loop" FOR-NEXT e uma instrução INPUT como descrito anteriormente.

(ii) Apague o "loop" FOR-NEXT e a instrução INPUT (mas não usando NEW, pois você pretende preservar o array).

(iii) Digite o resto do programa e armazene-o. Isso fará com que as variáveis sejam armazenadas, incluindo os arrays.

(iv) Quando você carregar de novo o programa, você também estará carregando o array.

(v) Ao operar o programa não use RUN, a fim de não apagar também as variáveis.

Capítulo 18

CADEIA DE CARACTERES (STRINGS)

Uma das coisas que o AS-ØØØ pode fazer e que uma máquina de calcular não pode é lidar com textos. Digite

PRINT" ÖI! EU SOU O SEU AS-1ØØØ".

("digitadas é SHIFT P).

A saudação entre aspas é chamada cadeia (significando uma cadeia de caracteres) e pode conter qualquer caracter que você queira exceto a cadeia de aspas". (Mas você pode usar a denominada imagem de aspas, "" SHIFT Q), e terá impresso: ").

Um erro comum de datilografia com cadeias é a omissão de uma das aspas, o que causará o sinal S Erro de Sintaxe.

Se você está imprimindo números, essas cadeias podem ser usadas para explicar o significado desses números. Por exemplo, se você possui uma loja de artigos esportivos, você deverá digitar

LET PREÇOGOLFE = 12,5Ø

e em seguida

PRINT "O PREÇO DE BOLAS DE GOLFE É";

PREÇOGOLFE"; "POR DÖZIA."

(Não se preocupe se for usada uma segunda linha).

Essa instrução exige três itens PRINT: a cadeira "O PREÇO DE BOLAS DE GOLFE É", o número 12,5Ø (o valor da variável PREÇOGOLFE), e em seguida a cadeia "POR DÖZIA". Na verdade você pode imprimir qualquer número de itens e qualquer combinação de cadeias e de números (ou de expressões). Observe como os espaços em uma cadeia fazem parte dela tanto quanto as letras, não sendo ignorados mesmo no final.

Há uma série de coisas que você pode fazer com cadeias.

1. Você pode atribuí-las a variáveis. Todavia o número da variável deve ser especial para mostrar que seu valor é uma cadeia e não um número: deverá ser uma única letra seguida por $(SHIFT U). Por exemplo, digite

LET A$ = "UNIFORME DE BASQUETEBOL" e

PRINT A$

2. Você pode juntá-las. Geralmente isso é chamado de concatenação, o que significa "encadeiamento", que é exatamente o que ela faz. Experimente

PRINT "JERSEY" + "E SHORTS"

Veja que você está perdendo um espaço. Agora tente

PRINT "JERSEY" + " E SHORTS"

Observe o espaço antes de E.

Você não pode subtrair, multiplicar ou dividir cadeias, nem elevá-las a potências.

3. Você pode aplicar algumas funções às cadeias para obter números e vice-versa.

LEN É aplicado a uma cadeia e o resultado é a extensão (o número de caracteres na cadeia), por exemplo:

```
LEN "LUVAS" = 5

LEN "BASTÕES" = 7
```

VAL é aplicado a uma cadeia para avaliar essa cadeia como uma expressão numérica. Por exemplo, (se A=9), VAL "1/2 + SQRA"=3,5. Se a cadeia a qual VAL é aplicada contém variáveis, duas regras terão que ser obedecidas:

(i) se a função VAL for parte de uma expressão maior, ela deverá ser o primeiro item; por exemplo 1Ø LET X = 7 + VAL "Y" deve ser mudada para 1Ø LET X = VAL "Y" + 7.

(ii) VAL pode aparecer apenas na primeira coordenada de uma instrução PRINT AT, PLOT ou UNPLOT (veja os capítulos 12 e 14) por exemplo:

```
1Ø PLOT 5, VAL "X" deve ser mudado para

1Ø LET Y = VAL "X"

15 PLOT 5, Y
```

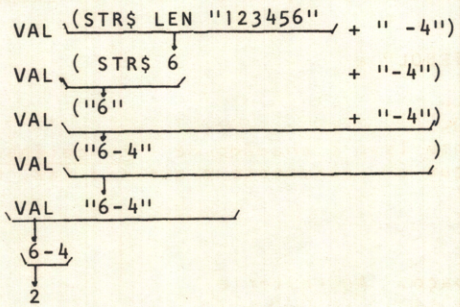
STR$ Quando essa instrução é aplicada a um número o resultado é o que aparecerá na tela se o número tiver sido exibido por uma instrução PRINT.

Por exemplo STR$ 3,5 = "3,5".

4. Exatamente do mesmo modo que para os números, você pode combinar isto para formar expressões em cadeia, como

```
VAL (STR$ LEN "123456" + "-4")
```

que é avaliado como



Resumo

Operações: + (para cadeias)

Funções: LEN, VAL, STR$

Exercícios:

1. Digite

```
LET A$ = "2+2"
```

e em seguida

```
PRINT A$; " = "; VAL A$
```

Tente mudar A$ para itens mais complicados e fazer o mesmo; por exemplo:

```
LEN A$= "ATN 1*4"
```

(Nesse caso a resposta seria π).

2. A cadeia "" sem caracteres é denominada cadeia vazia ou nula, sendo a única cadeia cujo comprimento é Ø. Lembre-se que os espaços são importantes e que cada uma cadeia vazia não é a mesma coisa que um espaço preenchido.

Não confunda com sinal de aspas "" (um único símbolo, SHIFT Q. Esse é um dispositivo especial para permitir usar aspas no meio de uma cadeia, devido ao fato de as aspas comuns não poderem ser usadas em tal lugar. Qual a razão disso? Quando o sinal de aspas aparece em uma cadeia que tem suas aspas no fim (por exemplo , na listagem de um programa), ela aparece como dois símbolos de aspas, para diferenciá-la das aspas comuns; porém quando ela é exibida por uma instrução PRINT, ela é apenas um símbolo de aspas. Tente

```
PINT "X"; "";"X"; """";""""";"";"""
```

3. Digite

```
PRINT "2+2=";2+1
```

Capítulo 19

SUB-STRINGS (SUBCADEIAS)

Uma sub-string consiste de um número de caracteres consecutivos de uma string, tomados em sequência.

Existe uma notação denominada SLICING para descrever sub-strings que podem ser aplicadas a expressões strings arbitrárias. A forma geral é:

Expressão STRING (início TO fim) de modo que, por exemplo,

"ABCDEF" (2 TO 5) = "BCDE"

Se você omitir o número inicial o número 1 é assumido. Por outro lado, se você omitir o número final, é assumido o comprimento da string. Desse modo

"ABCDEF" (TO 5) = "ABCDEF" (1 TO 5) = "ABCDE"

"ABCDEF" (2 TO) = "ABCDEF" (2 TO 6) = "BCDEF"

"ABCDEF" (TO) = "ABCDEF" (1 TO 6) = "ABCDEF"

Um outro modo de escrever esta última string é: "ABCDEF" (). Essa forma é ligeiramente diferente pelo fato de omitir TO e ter apenas um número:

"ABCDEF" (3) = "ABCDEF" (3 TO3) = "C"

Tanto o início quanto o fim devem se referir a partes existentes de uma string, porém esta regra é anulada por outra: caso o início seja maior do que o fim, o resultado será uma string vazia. Logo

"ABCDEF" (5 a 7)

resulta em erro 3 (erro subscrito) porque se a string contém apenas 6 caracteres, 7 é demais, porém

"ABCDEF" (8 a 7) = " "

e "ABCDEF" (1 a Ø) = " "

o início e o fim não podem ser negativos, caso isso ocorra você incorrerá no erro B.

O programa seguinte torna B$ igual a A$, mas omite qualquer espaço em branco.

```
1Ø INPUT A$
2Ø FOR N=LEN A$ TO 1 STEP -1
3Ø IF A$ (N) 1Ø<>""THEN GOTO 5Ø
4Ø NEXT N
```

```
70 GOTO 10
```

Observe que se A$ for somente espaços, na linha 50 haverá N=0 e A$ (TO N) = A$(1 TO 0) = "". No caso de variáveis strings, além de podermos extrair strings, podemos também atribuir caracteres substitutos para elas. Por exemplo, digite:

```
10 LET A$="ESTA ULTRAPASSA"
```

e então 20 LET A$(5 TO 8) ="******"

```
e      30 PRINT A$
```

Observe que como a sub-string A$ (5 TO 8) tem uma extensão de apenas 4 caracteres, foram usadas apenas os primeiros quatro asteristicos. Essa é uma característica que determina uma sub-string: após a operação ela tem que ter exatamente a mesma extensão que tinha antes. Para garantir esse fato, a string que está sendo atribuída para a sub-string é cortada no lado direito caso seja muito extensa ou completa com espaços caso seja muito curta. A isso damos o nome de <u>atribuição procustiana</u>.

Se você mudar a linha 20 e 30 para

```
20 LET A$ ()="TERRÍVEL"
```

e

```
      30 PRINT A$;"."
```

verá que a mesma coisa ocorreu de novo (desta vez acrecida de espaços) devido ao fato de A$() contar como uma sub-string.

```
20 LET A$="TERRÍVEL"
```

colocará o ponto corretamente, porém a string da linha 10 foi substituída pela string da linha 20

O SLICING pode ser considerado como tendo prioridade 12, então , por exemplo,

```
LEN "ABCDEF" (2 TO 5) = LEN ("ABCDEF" (2 TO 5)) = 4
```

Expressões de string complicadas terão que ser colocadas entre parênteses antes de uma operação SLICING. Por exemplo,

```
"ABC" + "DEF" (1 TO 2) = "ABCDE"

("ABC" + "DEF") (1 TO 2) = "AB"
```

<u>Resumo</u>

SLICING, usando TO. Observe que essa notação é usada apenas no AS-1000.

Exercícios:

1. Alguns tipos de BASIC (Não o BASIC do AS-1000) têm três funções denominadas LEFT$, RIGHT$ & MID$. LEFT$(A$,N) provê a sub-string / de A$ que consiste do primeiro caracteres de N enquanto que RIGHT$ (A$,N) provê a sub-string de A$ que consiste de caracteres do Nesimo em diante e MID$(A$,N1,N2) provê a sub-string de A$ consistindo de N2 caracteres iniciando no caracter N1.

   Como você escreveria isso no BASIC do AS-1000?

2. Tente essa sequência de comandos:

```
10 LET A$= "X*+*Y"

20 LET A$(2) = CHR$ 11 (O CARACTER ASPAS)

30 LET A$(4) = CHR$ 11

40 PRINT A$
```

Nesse caso A$ é uma string com as aspas dentro dela! Você poderá resolver esse problema esforçando-se o suficiente para isso, mas se você de início digitou

```
10 LET A$="X"+"Y"
```

a parte da direita do sinal de igual seria tratado como expressão , dando a A$ o valor "XY".

Digite:

```
50 LET B$="X""+""Y"

60 PRINT B$
```

você descobrirá que, apesar de A$ e B$ parecerem os mesmos quando impressos, eles não são iguais.

Veja:

```
70 PRINT A$=B$
```

O computador responde com "0" (a equação é falsa), porque B$ contém símbolo de caracteres de aspas simples (com código 192),enquanto A$ contém strings de aspas genuínos (com código 11).

Opere esse programa:

```
10 LET A$="LEN""ABCD""

100 PRINT A$;"=";VAL A$
```

Isso não dará certo porque VAL não considera símbolos de aspas "" como aspas.

Acrescente algumas linhas extras entre 10 e 100 para substituir os símbolos de aspas em A$ por strings de aspas (que você deverá chamar CHR$ 11), e tente de novo.

. Digite uma sub-rotina que ignora espaços, escreva e opere um programa que os use.

. Essa sub-rotina apaga todas as ocorrências da cadeia "SUPERMAN" a partir de A$.

```
1000 FOR N=1 TO LEN A$-7

1020 IF A$(N TO N+7)="SUPERMAN" THEN LET A$ (N TO N+7)"********"

1030 NEXT N

1040 RETURN
```

Escreva um programa que dê vários valores a A$ (por exemplo, "SUPERMAN É FORTE") e aplique a sub-rotina. Observe que "SUPERMAN" não tem que ser a primeira palavra em A$.

Capítulo 20

OS COMANDOS DE IMPRESSÃO NO BASIC DO AS-1ØØØ

Esse capítulo cobre as instruções especiais necessárias para operar uma impressora com o AS-1ØØØ.

Os dois primeiros, LPRINT e LLIST, são exatamente como PRINT e LIST, salvo que eles usam uma impressora ao invés de uma televisão. (O L é um acidente histórico. Quando o BASIC foi inventado normalmente era usada uma máquina de escrever elétrica ao invés de uma televisão, então PRINT realmente significava impressão. Se você quisesse uma grande quantidade de entradas você usaria uma impressora de linha muito rápida acoplada ao computador e uma instrução LPRINT significando "impressor de linha PRINT").

Tente esse programa, por exemplo,

```
1Ø LPRINT "ESTE PROGRAMA",,,,

2Ø LLIST

3Ø LPRINT,,"IMPRIME O CONJUNTO DE CARACTERES".,,,

4Ø FOR N=Ø TO 255

5Ø LPRINT CHR$ N;

6Ø NEXT N
```

A terceira instrução, COPY, imprime uma cópia do que aparece na tela da televisão. Por exemplo, tome uma listagem na tela, do programa acima , e digite:

```
COPY
```

A impressão pode ser interrompida ao acionar a tecla BREAK (espaço).

Se você usar essas instruções sem que a impressora esteja acoplada, normalmente você perderá a saída e prosseguirá com a próxima instrução. Todavia, algumas vezes o computador engasgará e você precisará usar a tecla BREAK para recuperá-lo.

Resumo

Instruções: LPRINT, LIST, COPY

Observação: Nenhuma dessas instruções é comum ao sistema BASIC, apesar de LPRINT ser usado por alguns outros computadores.

Exercícios

1. Tente resolver:

```
1Ø FOR N=31 TO Ø STEP -1
2Ø PRINT AT 31 -N,N;CHR$ (CODE "Ø" + N);
3Ø NEXT N
```

Você verá letras passando diagonalmente da parte superior do canto direito da tela até alcançar a parte inferior da mesma, momento em que o programa pára com erro de mensagem 5.

Mude "AT 31-N,N" na linha 2Ø para 'TAB N". O programa surtirá exatamente o mesmo efeito que antes.

Mude PRINT na linha 2Ø para LPRINT. Desta vez não haverá erro 5, que não deveria ocorrer com a impressora, e o padrão continuará sendo 10 linhas extras com os dígitos.

Agora mude 'TAB N' para 'AT 21-N,N' ainda usando LPRINT. Desta vez você obterá apenas uma única linha de símbolos. O motivo desta diferença é que a saída de LPRINT não é impressa imediatamente, mas sim arrumada em um armazenador de "buffer" uma imagem de uma linha a mais do qual o computador imprimirá:

(i) quando o "buffer" estiver cheio,

(ii) após uma instrução LPRINT que não terminar em vírgula ou em ponto e vírgula,

(iii) quando uma vírgula ou item de TAB necessitar de uma nova linha,

(iv) no fim do programa caso não tenha restado nada que não tenha sido impresso.

O número (iii) explica a razão de nosso programa com TAB funcionar do jeito que funciona. Da mesma forma que para AT, o número da linha é ignorado e a posição LPRINT (como a posição PRINT, mas para a impressora e não para a televisão) é mudada para o número de coluna. Um item AT nunca pode produzir uma linha que possa ser usada na impressora. (Na verdade o número da linha após AT não é completamente ignorado; ele tem que estar entre -21 e +21 para evitar erros. Por esse motivo sempre é mais seguro especificar a linha Ø. O item 'AT 21-N,N' na última versão do nosso programa teria sido muito melhor (apesar de menos ilustrativo) se tivesse sido substituído por 'AT Ø,N').

2. Faça um gráfico impresso de SIN (seno) operando o programa do Capítulo 16 e usando COPY.

Capítulo 21

O AS-1ØØØ

para Aqueles que Entendem de BASIC

Geral

Caso você já entenda de BASIC, você provavelmente não terá muita dificuldade em usar o AS-1ØØØ, porém ele tem algumas peculiaridades.

(i) As palavras não são soletradas, mas têm suas próprias teclas- esse fato está descrito no Capítulo 3 (para palavras-chave e para teclas pressionadas com SHIFT) e no Capítulo 15 (para nomes e funções) No texto essas palavras são impressas em negrito.

(ii) O BASIC do AS-1ØØØ não possui READ, DATA e RESTORE (mas veja o exercício 3 do Capítulo 17 a esse respeito), funções definidas de usuário (FN e DEF; apesar de VAL algumas vezes poder ser usada), e linhas de multi-instruções.

(iii) As cadeias de dispositivos de manejo são compreensíveis mas não padronizadas - veja os Capítulos 18 e 19, e também o Capítulo 17 (para cadeia de arrays).

(iv) O conjunto de caracteres do AS-1ØØØ é próprio dele.

(v) A imagem da televisão não é, em geral, mapeado em memória.

(vi) Se você está acostumado a usar PEEK e POKE em uma máquina diferente, lembre-se de que todos os endereços serão diferentes no AS-1ØØØ.

Velocidade

A máquina funciona em duas velocidades, denominadas modo de computação e exibição (SLOW), e modo rápido ("FAST").

Em SLOW a teia de TV é gerada continuamente e a computação é realizada durante os períodos de apagamento na parte superior e na parte inferior da imagem.

No modo rápido (fast mode) a imagem da televisão é desligada durante a computação e é exibida apenas ao final de um programa, enquanto estiver esperando dados de INPUT ou durante uma pausa (veja PAUSE).

Em fast opera cerca de quatro vezes mais rápido e deveria ser usado para programas que tenham muita computação como o contrário de saída, ou quando programas extensos estiverem sendo digitados.

Mudanças de velocidades são realizadas com as instruções FAST e SLOW.

O teclado

Os caracteres do AS-1ØØØ compreendem não apenas os símbolos (letras, dígitos, etc.) mas também os objetivos compostos (palavras-chave, nomes de funções, etc.; que são impressos aqui em negrito) e todos eles são introduzidos através do teclado ao invés de serem soletrados.A fim de que isso seja possível algumas teclas têm até cinco significados distintos,em parte dados pelo pressionamenro da tecla "shift" junto com a outra tecla desejada e em parte ajustando a máquina para modos dife-

rentes.

O modo é indicado pelo cursor, uma letra inversa no vídeo que mostra onde o próximo caracter do teclado será inserido.

Modo K (para palavras-chave) ocorre automaticamente quando a máquina espera por um comando ou uma linha de programa (ao invés de dados de INPUT) e desta posição na linha ele sabe que deverá esperar por um número de linha ou por uma palavra-chave. Isto se encontra no início da linha ou imediatamente após alguns dígitos no início da linha, ou imediatamente após THEN. Se a tecla "shift" não estiver pressionada, a próxima tecla será interpretada como uma palavra-chave (elas estão escritas acima das teclas) ou como um dígito.

Modo L (para letras) normalmente ocorre todas as outras vezes. Caso com a tecla "shift" não precionada, a próxima tecla será interpretada como o símbolo principal pertencente àquela tecla.

Nos modos K e L uma tecla pressionada com "shift" será interpretada como o caracter subsidiário vermelho que se encontra na parte superior do canto direito da tecla.

Modo F (para funções) ocorre após FUNCTION(SHIFT NEWLINE) ser pressionada e tem a duração de apenas um acionamento. Essa tecla será interpretada como um nome de função, que aparece abaixo das teclas.

Modo G para gráficos ocorre após GRAPHICS (SHIFT 9) ser pressionado e permanece até essa tecla ser pressionada de novo. Uma tecla não acionada simultaneamente com a tecla "shift" proverá o video inverso de seu modo L de interpretação. Uma tecla pressionada conjuntamente com "shift" indica um símbolo, porém se a essa tecla normalmente provê um objetivo, em modo de gráficos ela proverá o símbolo de gráficos que aparece na parte inferior do canto direito da tecla.

A tela

A tela possui 24 linhas, cada uma com 32 caracteres de comprimento e é dividida em duas partes. A parte superior tem no máximo 22 linhas e exibe tanto uma listagem ou uma saída de programa. A parte inferior possui pelo menos duas linhas e é usada para introduzir comandos, linhas de programa e dados de INPUT e também para apresentar os indicadores.

INPUT de teclados: aparece nas linhas que se encontrem na parte inferior da tela, ao ser digitada (símbolo simples ou objetivo composto) sendo inserido imediatamente antes do cursor. O cursor pode ser movido para a esquerda com ← (SHIFT 5) ou para a direita → (SHIFT 8). O caracter que se encontra antes do cursor pode ser retirado com RUBOUT SHIFT Ø. (observação: A linha pode ser apagada por completo digitando-se EDIT (SHIFT 1) seguido por NEWLINE).

Ao pressionar-se NEWLINE, e linha é executada, introduzida no programa ou usada como dados de INPUT como for apropriado, a menos que ela contenha um erro de sintaxe, caso em que o símbolo S aparecerá imediatamente antes do erro.

Ao serem introduzidas linhas de programa, uma listagem é exibida na metade superior da tela. A última linha a ser introduzida é denominada de linha atual e é indicada pelo símbolo >, porém isso pode ser mudado usando-se as teclas ↓ (SHIFT 6) e ↑ (SHIFT 7). Quando EDIT (SHIFT 1) é pressionada, a linha atual é conduzida à parte inferior da tela e pode ser editada.

Quando é executado um comando ou um programa é operado, a saída é exibida na metade superior da tela e permanece ali até que seja introduzida uma linha de programa, ou que NEWLINE seja pressionado com uma linha vazia, ou ↓ ou ↑ sejam pressionados. Na parte inferior aparece um relatório da forma m/n, em que m é um código que mostra o que ocorreu (veja Relatório de Códigos), e n é o número da última linha executada - ou Ø para um comando. A indicação permanece até que uma tecla seja pressionada e indique o modo. K .

Em algumas circunstâncias a tecla SPACE funciona como uma BREAK, fazendo com que o computador pare com uma indicação D. Isto é quando:

(i) no fim de uma instrução durante a operação do programa,

(ii) enquanto o computador está procurando por um programa na fita, ou,

(iii) enquanto o computador está usando a impressora (ou tentando usá-la quando ela não se encontra em uso).

O BASIC

Os números são armazenados com uma exatidão de 9 ou 1Ø dígitos. O maior número que você pode obter é cerca de $1Ø^{38}$ e o menor número (positivo) é cerca de $4*1Ø^{-39}$.

Um número é armazenado no AS-1ØØØ em ponto flutuante binário com um expoente e($1 <= e <= 255$) e quatro bytes de mantissa m ($1/2 <= m < 1$), o que representa o número $m*2^{e-128}$.

Já que $1/2 <= m < 1$, o bit mais significante de mantissa m sempre é 1. Portanto, em uma situação real podemos substituí-lo por um bit para mostrar o sinal - Ø para números positivos, 1 para números negativos.

Zero tem uma representação especial na qual todos os 5 bytes são Ø

Variáveis numéricas têm nomes de extensão arbitrária, que iniciam com uma letra e continuam com letras e dígitos. Os espaços são ignorados.

Variáveis de controle para loops FOR-NEXT têm nomes com comprimento de uma letra.

Arrays numéricos têm nomes de apenas uma letra, que pode ser o mesmo que o nome de uma variável simples. Elas podem ter muitas dimensões arbitrárias de tamanhos arbitrários. Os subscritos começam em 1.

As strings têm extensão completamente flexível. O nome de uma string consiste de uma única letra seguida por $.

Os arrays tipo string podem arbitrariamente ter muitas dimensões de tamanho arbitrário. Seu nome é uma única letra seguida por $ e não pode ser igual ao nome de uma string. Todas as strings em um dado / array têm a mesma extensão fixada, que é especificada como dimensão final extra na instrução DIM. Os subscritos começam em 1.

SLICING: Sub-strings de strings podem ser especificadas pelo uso de operações de SLICING. Tais operações podem ser:

(i) vazio ou

(ii) expressão numérica ou

(iii) expressão numérica opcional para expressão numérica opcional e é usado para expressar uma sub-string por:

(a) expressão de cadeia (operação de SLICING) ou por

(b) variável de array string (subscrito,..., subscrito,slicing)

que tem o mesmo significado de

variável de array string (subscrito,..., subscrito)(SLICING)

Em (a), suponha que a expressão de string tem o valor s$. Se o SLICER é vazio, o resultado é s$ considerado como uma sub-string dele mesmo. Se o slicer é uma expressão numérica com valor m, o resultado é o mº caracter de s$ (uma sub-string de extensão 1).

Se o slicer tem a forma (iii), suponha que a primeira expressão / numérica tem o valor m (o valor omitido é 1), e o segundo, n (o valor omitido é s$).

Se 1 <= m <= n <= a extensão de s$, o resultado é a sub-string s$, iniciando com mº caracter e terminando com o nº caracter.

Se 0<= n < m, o resultado é a string vazia.

De outro modo haverá como resultado o erro 3.

A operação de SLICING é realizada antes de funções ou operações / serem realizadas, a menos que outra ordem esteja indicada por parênte-- ses.

As sub-strings podem ser referenciadas (veja LET)

O argumento de uma função não necessita de parênteses caso ele seja uma constante ou uma variável (possivelmente subscrita ou passada para uma operação de SLICING.

Função

	Tipo de operando	Resultado
	(X)	
ABS	número	Magnitude absoluta.
ACS	número	Arcocoseno em radianos.
		Erro A se x não estiver na faixa de inteiros -1 a +1
AND	operação binária, operando da direita sempre um número	
	Operando numérico da esquerda:	A AND B = A se B ≠ Ø - Ø se B = Ø
	Operando esquerda STRING:	A$ AND B = A$ se B = Ø "" se B = Ø

ASN	número	Arcoseno em radianos. Erro A se x não estiver na faixa de -1 a +1.
ATN	número	Arcotangente em radianos.
CHR$	número	O caracter cujo código é x, arredondado para o próximo inteiro. Erro B se x não estiver na faixa de Ø a 255.
CODE	String	O código do primeiro caracter em x (ou Ø se x for uma string vazia).
COS	número (em radianos)	coseno
EXP	número	e^x.
INKEY$	nenhum	Lê o teclado. O resultado é o caracter que representa (no modo L) a tecla pressionada caso haja alguma, além da string vazia.
INT	número	Parte inteira (sempre é arredondada para menos).
LEN	String	Comprimento
LN	número	Logaritmo natural (na base e). Erro A se x < = Ø.
NOT	número	Ø se x ≠ Ø, 1 se x = Ø. NOT tem prioridade 4.
OR	operação binária, ambos números operandos	A OR B = 1 se B ≠ Ø A se B = Ø OR tem prioridade 2.
PEEK	número	O valor do byte na memória cujo endereço é x (é arredondado para o número inteiro mais próximo). Erro B se x não estiver na faixa de Ø a 65535.
PI	nenhum	π (3,14159265...)
RND	nenhum	O próximo número pseudoaleatório y em uma sequencia gerada tomando a potência do módulo 75 65537, subtraindo 1 e dividindo por 65536. 0 < = y < 1.
SGN	número	O sinal (-1, Ø ou +1) de x.
SIN	número	Seno.
SGR	número	Raiz quadrada. Erro B se x < Ø.
STR$	número	A string de caracteres que deveria ser exibida se x fosse impresso.

TAN	número (em radianos)	Tangente
USR	número	Chama o código de sub-rotina da máquina cujo endereço de início é x. Ao retornar o resultado é conteúdo do par registrador bc.
VAL	String	Calcula (sem suas aspas) como uma expressão numérica. Erro C se x contém um erro de sintaxe ou fornecer o valor da string. Outros erros são possíveis dependendo da expressão.
-	número	Negação

Os seguintes sinais são operações binárias:

+	Adição (em números), ou concatenação (em strings)
-	Subtração
*	Multiplicação
/	Divisão
**	Elevação a uma potência. Erro B se o operando da esquerda for negativo.
=	Igual
>	Maior do que
<	Menor do que
< =	Menor do que ou igual a
> =	Maior do que ou igual a
<>	Não é igual a

Os operandos terão que ser do mesmo tipo. O resultado é um número, 1 se a comparação é válida e Ø se não o for.

Funções e operações têm as seguintes prioridades:

Operação	Prioridade
Subescrever e slicing	12
Todas as funções exceto NOT e unário menos	11
**	1Ø
Unários menos	9
*,/	8
+,- (binário -)	6
=, >, <, <=, >=, <>	5

NOT	4
AND	3
OR	2

Instruções

Nesta lista,

@	representa uma única letra
v	representa uma variável
x,y,z	representa expressões numéricas
m,n	representa expressões numéricas que são arredondadas para o próximo número inteiro.
e	representa uma expressão
f	representada uma expressão com valor de string
s	representa uma instrução

Observe que expressões arbitrárias são permitidas em qualquer lugar (exceto para o número de linha no início de uma instrução).

Todas as instruções, salvo INPUT podem ser usadas tanto como comandos quanto em programas.

CLEAR	Apaga todas as variáveis, liberando o espaço que elas ocupam.
CLS	(Apaga a tela) Apaga o arquivo de exibição. Veja o Capítulo 26 no que diz respeito ao arquivo de exibição.
CONT	Suponha que a/b foi a última denotação com número diferente de zero. Então CONT terá o efeito GOTO b se a ≠ 9. GOTO b+1 se a = 9 (instrução STOP)
COPY	Envia uma cópia da tela à impressora, caso conectada; de outro modo não realiza nada. Relatório D se BREAK foi pressionado.
DIM @ $(n_1,\ldots,n_k)$	Apaga qualquer array com o nome e estabelece um array de números com dimensões K_1 $n_1,\ldots n_k$. Inicia todos os valores com Ø. Ocorre erro 4 se não houver espaço para acomodar o array. Um array é indefinido até ser dimensionado em uma instrução DIM.

DIM@ $(n_1,...,n_k)$

Apaga qualquer array ou string com o nome $ e estabelece um array de caracteres com dimensões k $n_1,...,n_k$. Inicia todos os valores com "". Isso pode ser considerado como um array de strings de extensão fixa n_k, com dimensões k-1 $n_1,...,n_{k-1}$. Ocorre erro 4 se não houver espaço para acomodar o array. Um array é indefinido até ser dimensionado em uma instrução DIM.

FAST

Inicia o modo fast (rápido), no qual o arquivo de exibição é mostrado apenas no final do programa, enquanto dados de INPUT estiverem sendo digitados ou durante uma pausa.

FOR@= x TO y

FOR @ = X TO Y STEP 1

FOR@= x TO y STEP z

Apaga qualquer variável @ e estabelece uma variável de controle x, limite y, estágio z, e endereço de looping 1 mais do que o número de linha da instrução FOR (-1 se ele é um comando). Verifica se o valor inicial é maior (se o estágio > = Ø) ou menor (se o estágio < Ø) do que o limite e caso isso ocorra, pula para a instrução NEXT no início da linha.

Veja NEXT

Ocorre erro 4 e se não houver espaço para a variável de controle.

GOSUBn

Coloca o número da instrução GOSUB em STACK. A seguir, age como GOTO n.

Ocorre erro 4 se não houver RETURNs suficientes.

GOTO n

Pula para a linha n (ou, se não houver linha n, para a primeira após esta).

IF x THEN s

Se x é verdadeiro (diferente de zero), s é executado.

A forma 'IF x THEN número de linha' não é permitida.

INPUT v

Pára (sem nenhuma sugestão especial) e espera que o usuário digite uma expressão; seu valor é atribuído a v. No modo rápido, o arquivo de exibição é mostrado. INPUT não pode ser usado como um comando; caso você tente, ocorrerá o erro 8.

Se o primeiro caracter na linha de INPUT for STOP, o programa parará com denotação D.

LET v=e

Atribui o valor de e à variável v.

LET não pode ser omitido.

Uma variável simples éindefinida até que ela seja atribuída em uma instrução LET ou INPUT.

Se v é uma variável string subscrita, a atribuição é procrustiana: O valor e da string é truncado ou preenchido com zeros à direita, para torná-lo do mesmo comprimento da variável v.

LIST

LIST Ø

LIST n

Lista o programa na tela de TV, iniciando na linha n e torna n a linha atual.

Erro 4 ou 5 se a listagem for muito grande para caber na tela; CONT fará exatamente a mesma coisa de novo.

LLIST

LLIST Ø.

LLIST n

Do mesmo modo que LIST, porém usando a impressora ao invés da televisão.

Não realizará nada se a impressora não estiver conectada.

Pára com relatório D se BREAK for pressionado.

LOAD f

Procura por um programa denominado f no tape e o carrega acompanhado de suas variáveis. Se f = "", carrega o primeiro programa disponível.

Se BREAK for pressionado,

(i) Se nenhum programa da fita tiver sido lido ainda, pára com relatório D e programa antigo;

(ii) Se parte de um programa tiver sido lida, executa NEW.

LPRINT ...

Da mesma forma que PRINT, porém usando a impressora ao invés da televisão. Uma linha do texto é enviada à impressora,

(i) quando a impressão salta de uma linha para a próxima;

(ii) após uma instrução LPRINT que não termina com uma vírgula ou com um ponto e vírgula.

(iii) quando uma vírgula ou item TAB exige uma nova linha, ou

(iv) no fim de um programa, se existir alguma coisa sem ter sido impressa.

Em um item AT apenas o número da coluna tem efeito; o número da linha é ignorado. Um item AT nunca envia uma linha de texto para a impressora.

Não haverá nenhum efeito se a impressora não estiver conectada.

Pára com denotação D se BREAK for pressionado.

NEW

Reinicia o sistema BASIC, apagando o programa e as variáveis e usando a memória até o byte cujo endereço está na variável de sistema RAMTOP, mas não incluindo esse byte (bytes 16388 e 16389).

NEXT @

(i) Encontra a variável de controle

(ii) Adiciona o passo a seu valor.

(iii) Se o passo > = Ø e o valor > limite; ou se o passo < Ø e o valor < limite, pulará para a linha de looping.

Erro 1 se não houver variável de controle

PAUSE n

Pára de computar e exibe o arquivo de imagem para n quadros (até 60 imagens por segundo) ou até que uma tecla seja pressionada.
Ø < = n < = 65535, além de erro B. Se n > = 32767, então a pausa não tem tempo determinado mas permanece até que uma tecla seja pressionada.

PLOT m,n

Torna preto o 'þixel= (|m|, |n|); move a posição de PRINT para ser ajustado depois daquele "pixel".
Ø < = |m| < = 63, Ø < = |n| < = 43, caso contrário erro B.

POKE m,n

Escreve o valor n para o byte armazenado com endereço m.
Ø < = m < = 65535, -255 < = n < = 255, caso contrário erro B.

PRINT...

O '...' é uma sequência de itens PRINT, separada por vírgulas ou ponto e vírgulas. Eles são escritos para o arquivo de imagens para ser mostrado na televisão. A posição (linha e coluna) onde o próximo caracter será impresso é denominada de posição PRINT. Um item PRINT pode ser

(i) vazio, por exemplo, nada

(ii) uma expressão numérica

Em primeiro lugar, um sinal de menos é impresso se o valor for negativo.

Agora deixa que x seja o módulo do valor.

Se $x < = 1Ø^{-5}$ ou $x > = 1Ø^{13}$, então será impresso usando notação científica. A parte da mantissa terá até oito dígitos (não acompanhados de zeros), e o ponto decimal (ausente se tiver apenas um dígito) será após o primeiro. A parte exponencial é E, seguida por + ou -, acompanhado de um ou dois dígitos.

De outro modo x é impresso em notação decimal ordinária com até oito dígitos significativos, e não acompanhado de zeros após o ponto decimal. Um ponto decimal no início, é sempre seguido de zero.

Ø é impresso como um único dígito Ø.

(iii) .uma expressão string.

Os símbolos nas strings são expandidos, possivelmente com um espaço antes ou depois deles.

O caracter de aspasé impresso como ". Caracteres não usados e caracteres de controle são impressos como?

(iv) AT m,n

A posição de impressão é mudada para a linha m (a contar da parte superior), coluna n (a contar da esquerda).

Ø < =|m|< = 21,Ø < = |n| < = 31, também o erro B. Se |m| = 22 ou 23, erro 5.

(v) TAB n

n é reduzido para o módulo 32. Então a posição de impressão é movida para a coluna n, permanecendo na mesma linha a menos que isso / envolva retrocesso, em cujo caso ela é movida para a próxima linha.

Ø < = n < = 255, Também o erro B.

Um ponto e vírgula entre dois itens deixa a posição de impressão sem mudança, de modo que o segundo item segue imediatamente após o primeiro. Uma vírgula, por outro lado, muda a posição de impressão pelo menos de um lugar e depois disso, tantos quanto sejam necessários para deixá-lo na Ø ou 16, mudando para uma nova linha caso necessário.

No fim da instrução PRINT, se ela não termina em um ponto e vírgula ou em uma vírgula, é iniciada uma nova linha.

Erro 5 significa que a tela está cheia. Nos casos a solução é CONT, que limpará a tela e permitirá a continuação do programa.

RAND

RAND n

RAND Ø

Estabelece a variável de sistema (denominado - SEED) usado para gerar o próximo valor de RND. Se n ≠ Ø, o SEED recebe o valor n; se n = Ø, ele recebe o valor de uma variável de sistema (denominado FRAMES) que conta os quadros exibidos até o momento na televisão, o que deve

ser razoavelmente aleatório (randomico).

Ocorre erro B se n não estiver na faixa de Ø a 65535.

REM . . . — Nenhum efeito '...' pode ser qualquer sequência de caracteres, exceto NEWLINE.

RETURN — Pega um número de linha no STACK de GOSUB e pula para a linha seguinte. Ocorre erro 7 quando não há número de linha no STACK. Há algum engano em seu programa; GOSUBs não são equilibrados corretamente por RETURNs.

RUN — RUN Ø

RUN n — CLEAR, e em seguida GOTO n.

SAVE f — Grava o programa e as variáveis em tape e os chama de f.

SAVE não deverá ser usado em uma rotina GOSUB.

Ocorre erro F se for uma string vazia, que não é permitido.

SCROLL — Leva o arquivo de imagem uma linha acima, perdendo a linha superior e criando uma outra na parte inferior.

Observe que a nova linha é totalmente vazia com apenas um caracter NEWLINE e nenhum espaço.

SLOW — Coloca o computador no modo computar e exibir em que o arquivo de imagem é mostrado continuamente e a computação é feita nos espaços a parte superior e na parte inferior da imagem.

STOP — Pára o programa com Relatório 9. CONT será resumido com a linha seguinte.

UNPLOT m,n — Do mesmo modo que PLOT, porém torna branco um "pixel" ao invés de torná-lo preto.

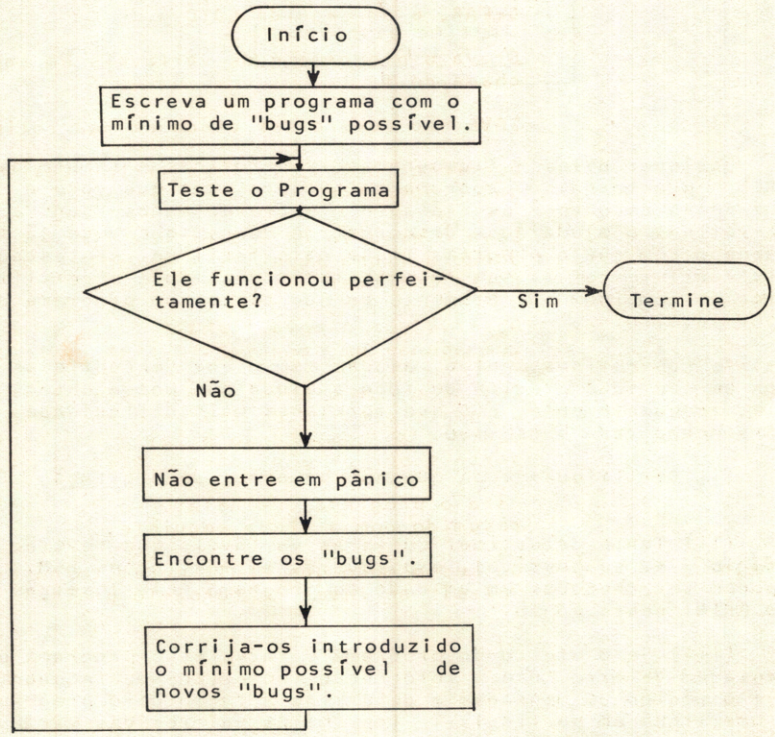
Capítulo 22

Trabalhando com Fluxogramas e Eliminando "bugs"

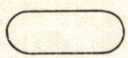
Há mais alguma coisa na arte de programar computadores do que apenas saber o que cada instrução faz. Provavelmente você já descobriu que a maioria de seus programas apresenta o que tecnicamente é conhecido por "bugs" (erro de programação, em inglês) quando você digita pela primeira vez, seja digitando erros, enganos em suas próprias idéias a respeito daquilo que o programa deveria fazer. Você deve atribuir isso a inexperiência.

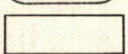
QUASE TODOS OS PROGRAMAS COMEÇAM COM "BUGS"

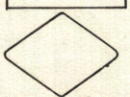
O plano geral pode ser ilustrado com um fluxograma:



A idéia é seguir as setas de bloco em bloco executando aquilo / que está indicado em cada bloco. Para cada instrução foi usado um tipo diferente de bloco.

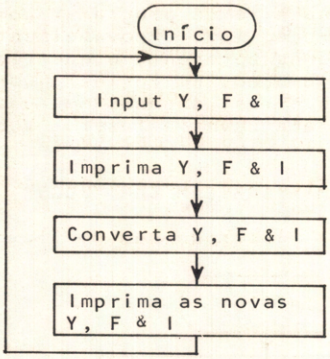
Um bloco arredondado  para começar ou para terminar.

Um bloco retangular  uma instrução normal qualquer

Um losango  exige que você tome alguma decisão antes de continuar.

(Apesar de não serem obrigatórias, essas formas são regularmente usadas).

Frequentemente usamos fluxogramas para descrever a estrutura geral do programa, com uma subrotina em quase todos os boxes. Deste modo um fluxograma para nosso exemplo de distância no capítulo 9 seria



Qualquer deles - fluxogramas, sub-rotinas e também as instruções REM - que tornam o programa mais claro, com que você o entenda melhor, e nesse caso você estará apto a escrever menos "bugs". Além disso as sub-rotinas o ajudarão a descartar os "Bugs" que você já tiver escrito, fazendo com que o programa fique mais fácil de ser testado. Será muito mais fácil testar as sub-rotinas individualmente e certificar-se de que elas se combinam corretamente do que testar um programa desestruturado por inteiro.

As sub-rotinas, então, ajudam com o box "encontre as bugs".Esse é o box em que você precisa de toda a ajuda que possa obter, uma vez que ele, frequentemente, é o que apresenta mais dificuldade. Outras dicas para encontrar "bugs" são:

(i) Certifique-se de que não há erros de digitação. Faça sempre isso.

(ii) Tente determinar quais as variáveis que deverão estar um cada estágio - e, se possível, explique-as em instruções REM. As variáveis podem ser checadas em um dado do programa pela inserção de uma instrução PRINT nesse ponto.

(iii) Se o efeito do programa é fazer esse programa parar com uma mensagem de erro, use a informação tão extensamente quanto possível. Observe o código de mensagem e determine a razão de o programa ter parado em uma linha em particular. Imprima os valores das variáveis se necessário.

(iv) Você deve estar apto a percorrer um programa linha por linha, digitando as linhas do programa como comandos.

(v) Imagine que você é o computador: opere o programa em você mesmo, usando lápis e papel para anotar os valores das variáveis.

Uma vez encontradas os "bugs", corrigi-los é quase a mesma coisa que escrever o programa original, porém você precisa testar o programa de novo. É surpreendentemente fácil corrigir um "bug" e introduzir outro em seu lugar.

Exercícios

1. Vez por outra - isso pode já ter ocorrido com você - você perderá uma extensão do programa que escreveu ou que estava terminando de escrever, disconectando o suprimento de força acidentalmente. Algumas vezes esse tipo de coisas ocorre espontaneamente; isso não é uma "bug" mas sim um problema elétrico. Não há nada que você possa fazer para evitar esse problema, porém, se ele ocorre com muita frequência, é provável que haja alguma coisa errada, mas seria válido armazenar o programa incompleto no tape no meio da programação.

2. O fluxograma para o cálculo distância não tem nenhum box de "fim". Isso tem importância? Onde você colocaria o fim no fluxograma se você o desejasse?

Capítulo 23

Sistemas de Números

Acreditamos que os seres humanos contam de dez em dez - o sistema decimal - porque têm dez dedos. Os computadores contam internamente de dois em dois , ou em binário. Isso não é porque eles têm apenas dois dedos, mas porque eles podem distinguir apenas entre dois estados de suas muitas chaves internas; ligado ou desligado (on ou off), 1 ou Ø.

Apesar de os engenheiros usarem um sistema binário na construção de computadores (veja a coluna da direita da tabela na próxima página), um outro sistema de números é útil devido ao fato de ser mais fácil de ler e poder ser facilmente convertido para binário. Esse sistema é chamado de hexadecimal (de dezesseis em dezesseis ou tendo base 16), e começa do seguinte modo:

Hex	português
Ø	zero
1	um
2	dois
:	:
:	:
9	nove
A	dez
B	onze
C	doze
D	treze
E	quatorze
F	quinze
1Ø	dezesseis
:	:
:	:
19	vinte e cinco
1A	vinte e seis
1B	vinte e sete
:	:
:	:
1F	trinta e um
2Ø	trinta e dois
21	trinta e três
:	:
:	:
9E	cento e cinqUenta e oito
9F	cento e cinqUenta e nove
AØ	cento e sessenta
A1	cento e sessenta e um
:	:
:	:
FE	duzentos e cinqUenta e quatro
FF	duzentos e cinqUenta e cinco
1ØØ	duzentos e cinqUenta e seis

Se você esta usando anotações hex e pretende tornar esse fato completamente claro, escreva "h" no fim do número e diga "hex". Por exemplo, para cento e cinqUenta e oito, escreva "9Eh" e diga "nove E hex".

Nos outros sistemas a contagem começa do seguinte modo:

Português	Decimal	Hexadecimal	Binário
zero	Ø	Ø	Ø ou ØØØØ
um	1	1	1 ou ØØØ1
dois	2	2	1Ø ou ØØ1Ø
três	3	3	11 ou ØØ11
quatro	4	4	1ØØ ou Ø1ØØ
cinco	5	5	1Ø1 ou Ø1Ø1
seis	6	6	11Ø ou Ø11Ø
sete	7	7	111 ou Ø111
oito	8	8	1ØØØ
nove	9	9	1ØØ1
dez	10	A	1Ø1Ø
onze	11	B	1Ø11
doze	12	C	11ØØ
treze	13	D	11Ø1
quatorze	14	E	111Ø
quinze	15	F	1111
dezesseis	16	1Ø	1ØØØØ

O ponto importante é que dezesseis é igual a dois elevado à quarta potência, o que torna muito fácil a conversão entre hex e binário.

Para converter hex para binário mude cada dígito hex para quatro "bits", usando a tabela acima. Nos referimos aos dígitos binários Ø e 1 como "bits".

Para converter binários para hex, divida o número binário em grupos de quatro bits, a partir da direita, e então converta cada grupo para o dígito hex correspondente.

Os bits no computador estão, em sua maioria, reunidos em grupos de oito, ou "bytes". Um único byte pode representar qualquer número de zero a duzentos e cinqUenta e cinco (11111111 binário ou hex FF); ou alternativamente, qualquer caracter no conjunto de caracteres do AS 1ØØØ, podendo seu valor ser escrito com dois dígitos hex.

Dois bytes podem ser agrupados para fazer o que é chamado tecnicamente de palavra. Uma palavra pode ser escrita usando dezesseis bits ou quatro dígitos hex e representa um número de Ø a (em decimal) $2^{16} - 1 = 65535$.

Um byte tem sempre oito bits, mas as palavras variam de computador para computador.

Resumo

Sistemas decimal, hexadecimal e binário.
Bits e bytes (Não os confunda) e palavras.

Exercícios

1. Como você converteria de libras para onças e de onças para libras novamente

   (i) Quando todos os números estão escritos em decimal?

   (ii) quando todos os números estão escritos em hex?

2. Como você faria a conversão decimal e hexadecimal? (pista: exercício 1).

Escreva programas no AS-1000 para converter valores numéricos em uma string dando sua representação hexadecimal e vice versa. (É isso que STR$ e VAL fazem com representações decimais).

Capítulo 24

Como Funciona o Computador

Está além do escopo deste manual descrever a eletrônica do AS-1000 e sua operação em detalhes, porém podemos dar uma idéia dos objetivos de seus componentes maiores.

O circuito integrado mais importante do AS-1000 é a CPU (UCP)(Unidade Central de Processamento). É um microprocessador Z80A. O processador faz a aritmética e controla eletronicamente o resto do computador de acordo com o sistema operacional.

O sistema operacional está contido na EPROM que é um dispositivo de armazenamento eletrônico em estado sólido que tem um programa (progrma residente) em ação permanente para fazer a CPU funcionar. Em forma simbólica ele é uma longa sequência de bytes. Cada byte tem um endereço que indica a sua posição na EPROM sendo que o primeiro byte tem endereço 0, o segundo, endereço 1, e assim por diante até 8191. Esta é a razão do Basic do AS-1000 ser chamado de 8K BASIC.

Você pode verificar qual o byte que está em um dado endereço usando a função PEEK. Por exemplo, esse programa imprime os primeiros 21 bytes da EPROM (e seus endereços).

```
10 PRINT "ENDEREÇO"; TAB 10; "BYTE"
20 FOR A = 0 TO 20
30 PRINT A;TAB 10;PEEK A
40 NEXT A
```

A porção RAM é um "rascunho" eletrônico que está ligado à CPU. Os programas BASIC que você digita são armazenados aqui eletronicamente, assim como suas variáveis, a imagem de televisão, e as variáveis / do sistema.

Do mesmo modo que EPROM, o armazenamento RAM é arrumado em bytes, cada um com um endereço. Esses variam de 16384 a 32767. Assim como / na EPROM você pode encontrar os valores desse bytes usando PEEK. A diferença é que você tambme pode mudá-los.

```
POKE 17300,57
```

Isso dá o byte no endereço 17300 o valor 57.

Se você digita

```
PRINT PEEK 17300
```

Você terá o número 57 de volta. (Tente com outros valores)

Observe que o endereço tem que estar entre 0 e 65535; e parte deles serão referentes a bytes da EPROM, não surtindo desse modo, nenhum efeito. O valor deve estar entre -255 e +255, e se for negativo, soma 256.

A habilidade de usar o POKE dá a você um imenso poder sobre o computador caso você saiba como usá-lo; todavia, o conhecimento necessário vai além daquilo que pode ser ensinado em um manual introdutório como esse.

Resumo

- Circuitos
- Instruções: POKE
- Funções: PEEK

Capítulo 25

Usando Códigos de Máquina

Este capítulo foi escrito para aqueles que entendem de código de máquina Z80, o conjunto de instruções que o processador chip Z80 usa. Caso você deseje aprender mais sobre códigos de máquina Z80 e programação, consulte livros especializados no assunto.

Códigos de rotinas de máquina podem ser executados em um programa BASIC usando a função USR. O argumento de USR é o endereço inicial da rotina, e seu resultado é um número inteiro não assinalado de dois bytes, o conteúdo do par registrador de bc em "return". O endereço de retorno para o BASIC é determinado pelo processo comum, de modo que retorno é obtido por uma instrução Z80 RET.

Há certas restrições nas rotinas USR:

(i) Em "return" o registrador i deverá ter o valor 1Eh.

(ii) As rotinas de display usam os registradores a', f' ix, iy e r; uma rotina USR não deverá usar esses registradores na modalidade / SLOW. (Até mesmo não é seguro ler o par af').

Todas as linhas do processador estão expostas na parte posterior do AS-1000. Desse modo, em princípio você pode fazer qualquer coisa que você faz com um Z80 usando o AS-1000.

Uma parte do código de máquina no meio da memória corre o risco de ser sobre-escrito pelo sistema BASIC. Alguns lugares mais seguros são:

(i) em uma instrução REM: digite uma instrução REM com caracteres suficientes para o seu código de máquina, que você dá POKE em seguida. Evite instruções HALT, pois essas instruções serão assumidas no fim da instrução REM.

(ii) Em uma string: Estabeleça uma string suficientemente grande e em seguida atribua um byte de código de máquina para cada caracter.

Nesses dois casos o código é seguro, mas propenso a se moverem . Isso ocorre especialmente com strings. No anexo o conjunto de caracteres, você encontrará os caracteres e as instruções Z80 escritos lado a lado e em ordem. Isso poderá ser útil quando você estiver introduzindo código.

(iii) Na parte superior da memória: Ao ser ligado, o AS- 1000 faz um teste para verificar a quantidade de memória e coloca o STACK da máquina ordenada na parte superior, de modo a não ficar espaço para rotinas USR. Ele armazena o endereço do primeiro byte inexistente (32768), em um sistema de variável conhecido como RAMTOP, nos dois bytes com endereços 16388 e 16389. NEW, por outro lado, não faz um teste completo da memória, apenas checa exatamente até antes do endereço em RAMTOP. Assim, se você fizer um POKE do endereço de um byte / existente em RAMTOP; para NEW toda a memória a partir daquele byte está fora do sistema BASIC e é deixada a parte. Por exemplo, você acabou de ligar o computador.

```
PRINT PEEK 16388+256*PEEK 16389
```

dá a você o endereço (32768) o do primeiro byte inexistente.

Agora suponha que você tem uma rotina USR de 20 bytes. Você pretende mudar RAMTOP para 32747 = 235 + 256*127 (como você operaria isso no computador?), então digite

```
POKE 16388, 235

POKE 16389, 127
```

e então NEW. Os vinte bytes da memória, do endereço 32747 a 32767, estão a sua disposição para que você faça com eles o que bem quiser. Se em seguida você digita NEW novamente, esses vinte bytes não serão afetados.

A parte superior da memória é um ótimo lugar para rotinas USR, seguro (mesmo partindo de NEW) e imóvel. Sua única desvantagem é que ela não é armazenada por SAVE.

Resumo

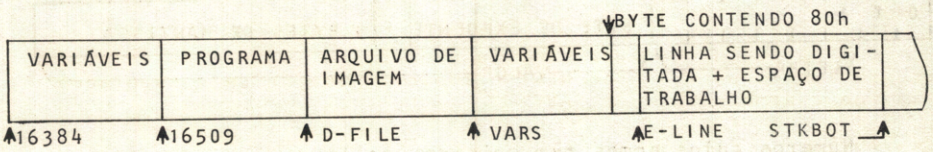
Funções: USR

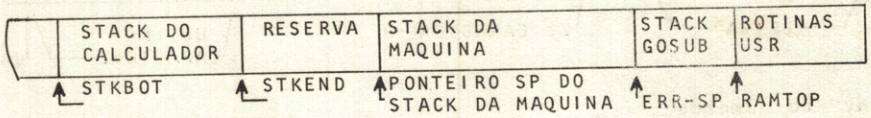
Instruções: NEW

Capítulo 26

Organização de Armazenamento

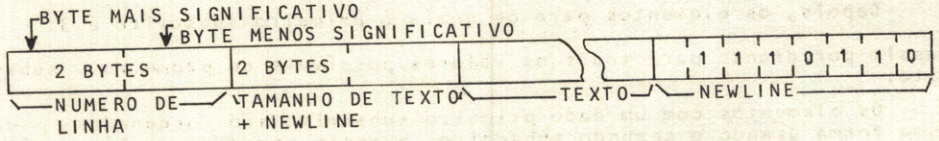
A memória é dividida em áreas diferentes para armazenar os diferentes tipos de informação. Essas áreas têm um tamanho suficiente para a informação que elas realmente contém e se você insere mais em um dado ponto (por exemplo, adicionando uma linha de programa ou variável), o espaço é feito deslocando tudo o que está acima daquele ponto. Inversamente, se você apaga a informação, tudo o que está acima do ponto é deslocado.





As variáveis do sistema contém várias informações que dizem ao computador em que tipo de estado ele está. Estas partes de informação estão completamente relacionadas no próximo capítulo, porém, por ora observe que existem algumas (denominados D.FILE, VARS, ELINE e assim / por diante) que contém os endereços dos limites entre as várias áreas da memória. Essas variáveis não são BASIC e seus nomes não serão reconhecidos pelo computador.

Cada linha é armazenada no programa do seguinte modo:



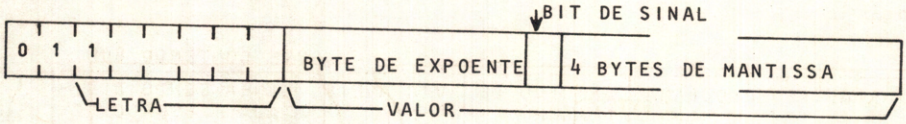
Observe que em oposição a todos os outros casos de números de dois bytes no Z80, o número de linha aqui (e também em uma variável de controle FOR-NEXT) é armazenado com o seu primeiro byte mais significativo: isto é, na ordem em que você deveria escrevê-los.

Uma constante numérica no programa é seguida por sua forma binária, usando o caracter CH$ 126 acompanhado de cinco bytes para o próprio número.

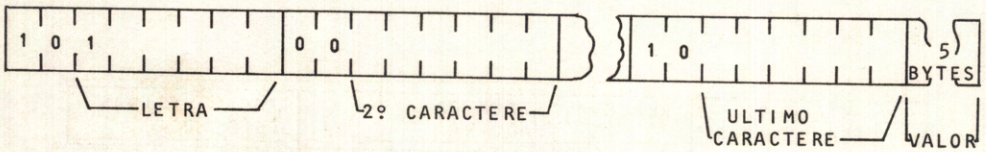
O arquivo de imagem é a cópia, na memória, da imagem da tela da TV. Ele começa com um caracter NEWLINE seguido de 24 linhas de texto, cada uma terminando com NEWLINE. O sistema é projetado de modo que a linha de texto não necessite sempre de 32 caracteres. Os espaços finais podem ser omitidos. Faz-se isso para ecomomizar espaço quando a memória é pequena.

As variáveis têm formatos diferentes, de acordo com suas diferentes / naturezas.

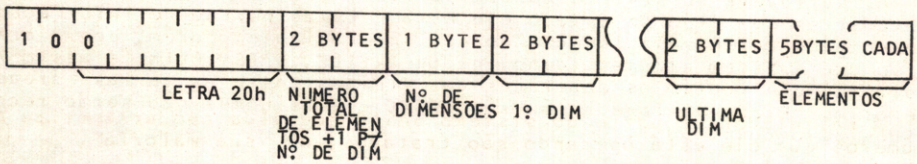
Números cujos nomes têm apenas uma letra:



Números cujos nomes têm mais de uma letra:



"Array" de números



A ordem dos elementos é:

Primeiro, o elemento para o qual o primeiro subscrito é 1.

Em seguida, os elementos para os quais o primeiro subscrito é 2.

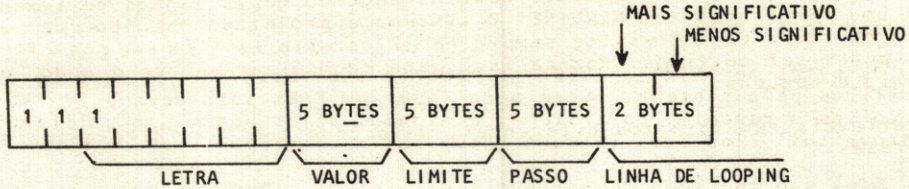
Depois, os elementos para os quais o primeiro subscrito é 3.

e assim por diante para todos os valores possíveis do promeiro subscrito.

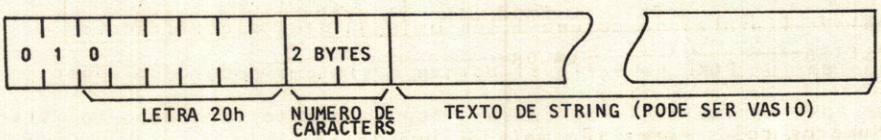
Os elementos com um dado primeiro subscrito são ordenados da mesma forma usando o segundo subscrito, e assim por diante até o último.

Por exemplo os elementos da "Array" B 3 x 6 no capítulo 22 são armazenados em ordem B(1,1), B(1,2), B(1,3), B(1,4), B(1,5), B(1,6), B(2,1), B(2,2),..., B(3,1), B(3,2),..., B(3.6).

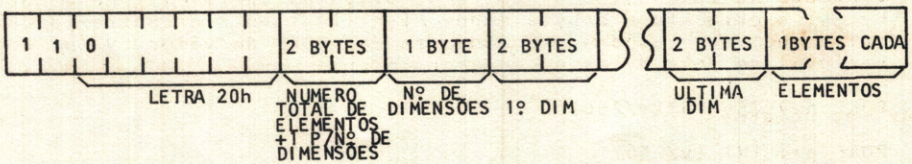
Variável de controle de um "loop" FOR-NEXT:



String:



"Array" de caracteres:



A parte que começa ELINE, contém a linha que está sendo digitada (como um comando, uma linha de programa, ou dado de INPUT) e também algum espaço para trabalho.

A calculadora é parte do sistema BASIC que lida com aritmética e os números que ele está operando são tratados, em sua maioria , no STACK da calculadora.

A parte livre contém o espaço até então não usado.

O STACK da máquina é o STACK usado pelo "chip" Z-8Ø para manter o endereço de retorno e assim por diante.

Você terá que deixar de lado os espaços para rotinas USR, usando NEW como está descrito no último capítulo.

Capítulo 27

Variáveis do Sistema

Os bytes de 16384 a 16508 na memória são reservados para usos específicos pelo sistema. Você pode fazer um PEEK para descobrir várias coisas a respeito do sistema, e algumas delas podem ser utilizadas com proveito. Elas são listradas aqui com seus respectivos usos.

Elas são chamadas de variáveis do sistema e possuem nomes, porém não as confunda com as variáveis usadas pelo BASIC. Você não pode usar os nomes em um programa BASIC, pois eles são apenas mnemônicos que são usados para tornar mais fácil a referência às variáveis.

As abreviaturas na coluna 1 têm os seguintes significados:

X Não dar um POKE na variável porque o sistema poderá colapsar

N Um POKE na variável não causará nenhum efeito duradouro (o sistema logo se rearruma)

S A variável é armazenada por SAVE

O número na coluna 1 é o número de bytes na variável. Para dois bytes, o primeiro é o byte menos significativo - o contrário daquilo que você espera. Então para dar POKE de valor V a uma variável de dois bytes no endereço n, use

```
POKE n,v-256*INT(v/256)

POKE n+1,INT (v/256)
```

e para dar PEEK de seu valor, use a expressão:

```
PEEK n+256*PEEK (n+1)
```

Observações	Endereço	Nome	Conteúdo
1	16384	ERR-NR	1 menor do que o código de denotação. Começa em 255 (para -1), então PEEK 16384, se funcionar dará, como resultado 255. POKE 16384, n pode ser usado para forçar uma parada de erro: 0 <= n <= 14 tem / como resultado uma das denotações comuns, 15 <= n <= 34 ou 99 <= n <= 127 tem como resultado uma denotação não padrão, e 35 <= n <= 98 é propenso a desordenar o arquivo de imagem.
X1	16385	FLAGS	Vários "flags" para controlar o sistema BASIC.
X2	16386	ERR-SP	Endereço do primeiro item no STACK da máquina (após retorno GOSUB)
2	16388	RAMTOP	Endereço do primeiro byte acima da área do sistema BASIC. Você pode dar um POKE para fazer com que NEW reserve espaço acima daquela área

			(veja o Capítulo 25) ou para ludibriar CLS para reservar um arquivo de imagem menor (Capítulo 26).
N1	16390	MODE	Especifica o cursor K, L, F ou G
N2	16391	PPC	Número de linha da instrução que está sendo executada. Dar POKE não resultará em nenhum efeito duradouro a não ser em última linha do programa.
S1	16393	VERSN	Ø identifica 8K EPROM em programas salvos.
S2	16394	E-PPC	Número da linha atual (com o cursor do programa).
SX2	16396	D-FILE	Veja o Capítulo 26.
S2	16398	DF-CC	Endereço da posição PRINT no arquivo de imagem. Pode ser dado POKE de modo que a saída de PRINT seja remetida a outro lugar.
SX2	16400	VARS	Veja o Capítulo 26.
SN2	16402	DEST	Endereço da variável em atribuição.
SX2	16404	E-LINE	Veja Capítulo 26.
SX2	16406	CH-ADD	Endereço do próximo caracter a ser interpretado: O caracter após o argumento de PEEK ou o NEWLINE no final de uma instrução POKE.
S2	16408	X-PTR	Endereço do caracter que precede o marcador.
SX2	16410	STKBOT	Veja Capítulo 26.
SX2	16412	STKEND	Veja Capítulo 26.
SN1	16414	BREG	Registro b da calculadora.
SN2	16415	MEM	Endereço da área usada para a memória da calculadora. (Algumas vezes, MEMBOT mas não sempre).
S1	16417	Não usado	
SX1	16418	DF-SZ	O número de linhas (incluindo uma linha em branco) na parte inferior da tela.
S2	16419	S-TOP	O número da linha superior do programa em listagens automáticas.
SN2	16421	LAST-K	Indica quais as teclas que foram pressionadas.
SN1	16423		Estabelece o estado de DEBOUNCE

			do teclado.
SN1	16424	MARGIN	Número de linhas em branco acima ou abaixo da imagem - 31.
SX2	16425	NXTLIN	Endereço da próxima linha do programa a ser executada.
S2	16427	OLDPPC	Número da linha para a qual CONT pula.
SN1	16429	FLAGX	Vários FLAGS.
SN2	16430	STRLEN	Extensão da designação do tipo de string em atribuição.
SN2	16432	T-ADDR	Endereço do próximo item na tabela de sintaxe (não muito provável ser útil).
S2	16434	SEED	Valor base para RND. Essa é a variável que é fixada por RAND.
S2	16436	FRAMES	Conta os quadros mostrados na televisão. O bit 15 é 1. Os bits 0 a 14 são decrementados para cada imagem enviada à televisão. Ele pode ser usado para contagem de tempo, porém PAUSE também o usa. PAUSE converte o bit 15 para 0 e coloca a extensão da pausa em bits de 0 a 14. Quando os bits tiverem sido contados decrescentemente até zero, a pausa cessará. Quando a pausa cessa devido ao pressionamento de uma tecla, o bit 15 é novamente remetido a 1.
S1	16438	COORDS	Coordenada X do último ponto PLOTado.
S1	16439		Coordenada Y do último ponto PLOTado.
S1	16440	PR-CC	Byte menos significativo do endereço da próxima posição para LPRINT imprimir (em PRBUFF).
SX1	16441	S-POSN	Número da coluna para a posição PRINT.
SX1	16442		Número da linha para a posição PRINT.
S1	16443	CDFLAG	Vários FLAGS. O bit 7 está em operação (1) durante a modalidade / SLOW.
S33	16444	PRBUFF	O BUFFER de impressão (33º caracter) é NEWLINE.
SN30	16477	MEMBOT	Área de memória da calculadora; u-

sada para armazenar números que, por conveniência, não podem ser colocados no STACK da calculadora.

S2 16507 não usado

Exercícios:

1. Experimente este programa

```
10 FOR N = 0 TO 21
20 PRINT PEEK (PEEK 16400+256*PEEK 16401+N)
30 NEXT N
```

Isso indica os primeiros 22 bytes da área de variáveis:

Compare o controle de variável N com a descrição constante no Capítulo 26.

2. Mude a linha 20 do programa acima para

```
20 PRINT PEEK (16509+N)
```

Isso indica os primeiros 22 bytes do programa. Compare-os com o próprio programa.

Anexo

O Conjunto de Caracteres

Segue-se o conjunto completo de caracteres do AS-1ØØØ com códigos em decimal e em hex (hexadecimal). Imaginando-se que os códigos sejam instruções de código de máquina Z8Ø, a coluna da direita fornece os mnemônicos correspondentes.

Código	*Caracter*	*Hexa-decimal*	*Z8Ø assembler*	*após CB*	*após ED*
Ø	espaço	ØØ	nop	rlc b	
1	▗	Ø1	ld bc,NN	rlc c	
2	▖	Ø2	ld (bc),a	rlc d	
3	▄	Ø3	inc bc	rlc e	
4	▝	Ø4	inc b	rlc h	
5	▐	Ø5	dec b	rlc l	
6	▞	Ø6	ld b,N	rlc (hl)	
7	▟	Ø7	rlca	rlc a	
8	▒	Ø8	ex af,af'	rrc b	
9	▓	Ø9	add hl,bc	rrc c	
1Ø	▒	ØA	ld a,(bc)	rrc d	
11	"	ØB	dec bc	rrc e	
12	£	ØC	inc c	rrc h	
13	$	ØD	dec c	rrc l	
14	:	ØE	ld c,N	rrc (hl)	
15	?	ØF	rrca	rrc a	
16	(	1Ø	djnz DIS	rl b	
17	)	11	ld de,NN	rl c	
18	>	12	ld (de),a	rl d	
19	<	13	inc de	rl e	
2Ø	=	14	inc d	rl h	
21	+	15	dec d	rl l	
22	—	16	ld d,N	rl (hl)	
23	*	17	rla	rl a	
24	/	18	jr DIS	rr b	
25	;	19	add hl,de	rr c	
26	'	1A	ld a,(de)	rr d	
27	.	1B	dec dē	rr e	
28	Ø	1C	inc e	rr h	

Código	*Caracter*
29	1
3Ø	2
31	3
32	4
33	5
34	6
35	7
36	8
37	9
38	A
39	B
4Ø	C
41	D
42	E
43	F
44	G
45	H
46	I
47	J
48	K
49	L
5Ø	M
51	N
52	O
53	P
54	Q
55	R
56	S
57	T
58	U
59	V
6Ø	W
61	X
62	Y
63	Z
64	**RND**
65	**INKEY$**
66	PI
67	não usado
68	
69	
7Ø	
71	
72	
73	
74	
75	
76	
77	
78	
79	
8Ø	
81	
82	

Hexa-decimal	*Z8Ø assembler*	*após CB*	*após ED*
1D	dec e	rr l	
1E	ld e,N	rr (hl)	
1F	rra	rr a	
2Ø	jr nz,DIS	sla b	
21	ld hl,NN	sla c	
22	ld (NN),hl	sla d	
23	inc hl	sla e	
24	inc h	sla h	
25	dec h	sla l	
26	ld,h,N	sla (hl)	
27	daa	sla a	
28	jr z,DIS	sra b	
29	add hl,hl	sra c	
2A	ld hl,(NN)	sra d	
2B	dec hl	sra e	
2C	inc l	sra h	
2D	dec I	sra l	
2E	ld l,N	sra (hl)	
2F	cpl	sra a	
3Ø	jr nc,DIS		
31	ld sp,NN		
32	ld (NN),a		
33	inc sp		
34	inc (hl)		
35	dec (hl)		
36	ld (hl),N		
37	scf		
38	jr c,DIS	srl b	
39	add hl,sp	srl c	
3A	ld a,(NN)	srl d	
3B	dec sp	srl e	
3C	inc a	srl h	
3D	dec a	srl l	
3E	ld a,N	srl (hl)	
3F	ccf	srl a	
4Ø	ld b,b	bit Ø,b	in b,(c)
41	ld b,c	bit Ø,c	out (c),b
42	ld b,d	bit Ø,d	sbc hl,bc
43	ld b,e	bit Ø,e	ld (NN),bc
44	ld b,h	bit Ø,h	neg
45	ld b,l	bit Ø,l	retn
46	ld b,(hl)	bit Ø,(hl)	im Ø
47	ld b,a	bit Ø,a	ld i,a
48	ld c,b	bit 1,b	in c,(c)
49	ld c,c	bit 1,c	out (c),c
4A	ld c,d	bit 1,d	adc hl,bc
4B	ld c,e	bit 1,e	ld bc,(NN)
4C	ld c,h	bit 1,h	
4D	ld c,l	bit 1,l	reti
4E	ld c,(hl)	bit 1,(hl)	
4F	ld c,a	bit 1,a	ld r,a
50	ld d,b	bit 2,b	in d,(c)
51	ld d,c	bit 2,c	out (c),d
52	ld d,d	bit 2,d	sbc hl,de

Código	*Caracter*	*Hexa-decimal*	*Z8Ø assembler*	*após CB*	*após ED*
83		53	ld d,e	bit 2,e	ld (NN),de
84		54	ld d,h	bit 2,h	
85		55	ld d,l	bit 2,l	
86		56	ld d,(hl)	bit 2,(hl)	im 1
87		57	ld d,a	bit 2,a	ld a,i
88		58	ld e,b	bit 3,b	in e,(c)
89		59	ld e,c	bit 3,c	out (c),e
9Ø		5A	ld e,d	bit 3,d	adc hl,de
91		5B	ld e,e	bit 3,e	ld de,(NN)
92		5C	ld e,h	bit 3,h	
93		5D	ld e,l	bit 3,l	
94		5E	ld e,(hl)	bit 3,(hl)	im 2
95		5F	ld e,a	bit 3,a	ld a,r
96		6Ø	ld h,b	bit 4,b	in h,(c)
97	não usado	61	ld h,c	bit 4,c	out (c),h
98		62	ld h,d	bit 4,d	sbc hl,hl
99		63	ld h,e	bit 4,e	ld (NN),hl
1ØØ		64	ld h,h	bit 4,h	
1Ø1		65	ld h,l	bit 4,l	
1Ø2		66	ld h,(hl)	bit 4,(hl)	
1Ø3		67	ld h,a	bit 4,a	rrd
1Ø4		68	ld l,b	bit 5,b	in l,(c)
1Ø5		69	ld l,c	bit 5,c	out (c),l
1Ø6		6A	ld l,d	bit 5,d	adc hl,hl
1Ø7		6B	ld l,e	bit 5,e	ld de,(NN)
1Ø8		6C	ld l,h	bit 5,h	
1Ø9		6D	ld l,l	bit 5,l	
11Ø		6E	ld l,(hl)	bit 5,(hl)	
111		6F	ld l,a	bit 5,a	rld
112	cursor para cima	7Ø	ld (hl),b	bit 6,b	
113	cursor para baixo	71	ld (hl),c	bit 6,c	
114	cursor para esquerda	72	ld (hl),d	bit 6,d	sbc hl,sp
115	cursor para direita	73	ld (hl),e	bit 6,e	ld (NN),sp
116	**GRAPHICS**	74	ld (hl),h	bit 6,h	
117	**EDIT**	75	ld (hl),l	bit 6,l	
118	**ENTER**	76	halt	bit 6,(hl)	
119	**DELETE**	77	ld (hl),a	bit 6,a	
12Ø	**K**/**L** modo	78	ld a,b	bit 7,b	in a,(c)
121	**FUNCTION**	79	ld a,c	bit 7,c	out (c),a
122	não usado	7A	ld a,d	bit 7,d	adc hl,sp
123	não usado	7B	ld a,e	bit 7,e	ld sp,(NN)
124	não usado	7C	ld a,h	bit 7,h	
125	não usado	7D	ld a,l	bit 7,l	
126	número	7E	ld a,(hl)	bit 7,(hl)	
127	cursor	7F	ld a,a	bit 7,a	
128	▀	8Ø	add a,b	res Ø,b	
129	▜	81	add a,c	res Ø,c	
13Ø	▙	82	add a,d	res Ø,d	
131	▄	83	add a,e	res Ø,e	
132	▛	84	add a,h	res Ø,h	
133	▐	85	add a,l	res Ø,l	
134	▚	86	add a,(hl)	res Ø,(hl)	
135	▜	87	add a,a	res Ø,a	
136	▓	88	adc a,b	res 1,b	

Código	*Caracter*	*Hexa-decimal*	*Z8Ø assembler*	*após CB*	*após ED*
137	▓	89	adc a,c	res 1,c	
138	▓	8A	adc a,d	res 1,d	
139	inverso "	8B	adc a,e	res 1,e	
14Ø	inverso	8C	adc a,h	res 1,h	
141	inverso $	8D	adc a,l	res 1,l	
142	inverso :	8E	adc a,(hl)	res 1,(hl)	
143	inverso ?	8F	adc a,a	res 1,a	
144	inverso (	9Ø	sub b	res 2,b	
145	inverso)	91	sub c	res 2,c	
146	inverso >	92	sub d	res 2,d	
147	inverso <	93	sub e	res 2,e	
148	inverso =	94	sub h	res 2,h	
149	inverso +	95	sub l	res 2,l	
15Ø	inverso —	96	sub (hl)	res 2,(hl)	
151	inverso *	97	sub a	res 2,a	
152	inverso /	98	sbc a,b	res 3,b	
153	inverso ;	99	sbc a,c	res 3,c	
154	inverso ,	9A	sbc a,d	res 3,d	
155	inverso .	9B	sbc a,e	res 3,e	
156	inverso Ø	9C	sbc a,h	res 3,h	
157	inverso 1	9D	sbc a,l	res 3,l	
158	inverso 2	9E	sbc a,(hl)	res 3,(hl)	
159	inverso 3	9F	sbc a,a	res 3,a	
16Ø	inverso 4	AØ	and b	res 4,b	ldi
161	inverso 5	A1	and c	res 4,c	cpi
162	inverso 6	A2	and d	res 4,d	ini
163	inverso 7	A3	and e	res 4,e	outi
164	inverso 8	A4	and h	res 4,h	
165	inverso 9	A5	and l	res 4,l	
166	inverso A	A6	and (hl)	res 4,(hl)	
167	inverso B	A7	and a	res 4,a	
168	inverso C	A8	xor b	res 5,b	ldd
169	inverso D	A9	xor c	res 5,c	cpd
17Ø	inverso E	AA	xor d	res 5,d	ind
171	inverso F	AB	xor e	res 5,e	outd
172	inverso G	AC	xor h	res 5,h	
173	inverso H	AD	xor l	res 5,l	
174	inverso I	AE	xor (hl)	res 5,(hl)	
175	inverso J	AF	xor a	res 5,a	
176	inverso K	BØ	or b	res 6,b	ldir
177	inverso L	B1	or c	res 6,c	cpir
178	inverso M	B2	or d	res 6,d	inir
179	inverso N	B3	or e	res 6,e	otir
18Ø	inverso O	B4	or h	res 6,h	
181	inverso P	B5	or l	res 6,l	
182	inverso Q	B6	or (hl)	res 6,(hl)	
183	inverso R	B7	or a	res 6,a	
184	inverso S	B8	cp b	res 7,b	lddr
185	inverso T	B9	cp c	res 7,c	cpdr
186	inverso U	BA	cp d	res 7,d	indr
187	inverso V	BB	cp e	res 7,e	otdr
188	inverso W	BC	cp h	res 7,h	
189	inverso X	BD	cp l	res 7,l	
19Ø	inverso Y	BE	cp (hl)	res 7,(hl)	

Código	*Caracter*	*Hexa-decimal*	*Z8Ø assembler*	*após CB*	*após ED*
191	inverso Z	BF	cp a	res 7,a	
192	"""	CØ	ret nz	set Ø,b	
193	**AT**	C1	pop bc	set Ø,c	
194	**TAB**	C2	jp nz,NN	set Ø,d	
195	não usado	C3	jp NN	set Ø,e	
196	**CODE**	C4	call nz,NN	set Ø,h	
197	**VAL**	C5	push bc	set Ø,l	
198	**LEN**	C6	add a,N	set Ø,(hl)	
199	**SIN**	C7	rst Ø	set Ø,a	
2ØØ	**COS**	C8	ret z	set 1,b	
2Ø1	**TAN**	C9	ret	set 1,c	
2Ø2	**ASN**	CA	jp z,NN	set 1,d	
2Ø3	**ACS**	CB		set 1,e	
2Ø4	**ATN**	CC	call z,NN	set 1,h	
2Ø5	**LN**	CD	cal NN	set 1,l	
2Ø6	**EXP**	CE	adc a,N	set 1,(hl)	
2Ø7	**INT**	CF	rst 8	set 1,a	
2Ø8	**SQR**	DØ	ret nc	set 2,b	
2Ø9	**SGN**	D1	pop de	set 2,c	
21Ø	**ABS**	D2	jp nc,NN	set 2,d	
211	**PEEK**	D3	out N,a	set 2,e	
212	**USR**	D4	call nc,NN	set 2,h	
213	**STR$**	D5	push de	set 2,l	
214	**CHR$**	D6	sub N	set 2,(hl)	
215	**NOT**	D7	rst 16	set 2,a	
216	**	D8	ret c	set 3,b	
217	**OR**	D9	exx	set 3,c	
218	**AND**	DA	jp c,NN	set 3,d	
219	<=	DB	in a,N	set 3,e	
22Ø	>=	DC	call c,NN	set 3,h	
221	<>	DD	prefixo de instruções usando ix	set 3,l	
222	**THEN**	DE	sbc a,N	set 3,(hl)	
223	**TO**	DF	rst 24	set 3,a	
224	**STEP**	EØ	ret po	set 4,b	
225	**LPRINT**	E1	pop hl	set 4,c	
226	**LLIST**	E2	jp po,NN	set 4,d	
227	**STOP**	E3	ex (sp),hl	set 4,e	
228	**SLOW**	E4	call po,NN	set 4,h	
229	**FAST**	E5	push hl	set 4,l	
23Ø	**NEW**	E6	and N	set 4,(hl)	
231	**SCROLL**	E7	rst 32	set 4,a	
232	**CONT**	E8	ret pe	set 5,b	
233	**DIM**	E9	jp (hl)	set 5,c	
234	**REM**	EA	jp pe,NN	set 5,d	
235	**FOR**	EB	ex de,hl	set 5,e	
236	**GOTO**	EC	call pe,NN	set 5,h	
237	**GOSUB**	ED		set 5,l	
238	**INPUT**	EE	xor N	set 5,(hl)	
239	**LOAD**	EF	rst 4Ø	set 5,a	
24Ø	**LIST**	FØ	ret p	set 6,b	
241	**LET**	F1	pop af	set 6,c	
242	**PAUSE**	F2	jp p,NN	set 6,d	
243	**NEXT**	F3	di	set 6,e	

Código	*Caracter*	*Hexa-decimal*	*Z8Ø assembler*	*após CB*	*após ED*
244	**POKE**	F4	call p,NN	set 6,h	
245	**PRINT**	F5	push af	set 6,l	
246	**PLOT**	F6	or N	set 6,(hl)	
247	**RUN**	F7	rst 48	set 6,a	
248	**SAVE**	F8	ret m	set 7,b	
249	**RAND**	F9	ld sp,hl	set 7,c	
25Ø	**IF**	FA	jp m,NN	set 7,d	
251	**CLS**	FB	ei	set 7,e	
252	**UNPLOT**	FC	call m,NN	set 7,h	
253	**CLEAR**	FD	prefixo de instruções usando iy	set 7,l	
254	**RETURN**	FE	cp N	set 7,(hl)	
255	**COPY**	FF	rst 56	set 7,a	

Códigos Indicadores

Esta tabela provê o relatório de códigos com uma descrição geral e uma lista de instruções e funções nas quais eles podem ocorrer. No capítulo 21, sob cada instrução ou função, você encontrará uma descrição mais detalhada do que significa a indicação de erro.

Código	Significado	Situação
Ø	Conclusão bem sucedida ou salto para o número da linha maior do que qualquer outra inexistente.	qualquer uma
1	A variável de controle não existe (não foi estabelecida por uma instrução porém há uma variável ordinária com o mesmo nome.	NEXT
2	Uma variável indefinida foi usada. Isso ocorrerá com uma variável simples caso a variável seja usada antes de ser atribuída em uma instrução LET. Para uma variável subscrita isso ocorrerá caso a variável seja usada antes de ser dimensionada em uma instrução DIM. E para uma variável de controle, isso ocorrerá caso a mesma for usada antes de ter sido definida como variável de controle por uma instrução FOR e se não houver nenhuma variável simples com o mesmo nome.	qualquer uma
3	Subscrito fora de limite O subscrito fora do limite (negativo ou maior do que 65535) resultará em erro B.	variáveis subscritas.
4	Falta de espaço suficiente na memória . Observe que o número da linha na indicação (após a /) pode não estar completo na tela devido à falta de memória: Por exemplo, 4/2Ø pode aparecer 4/2 Veja o Capítulo 9.	LET, INPUT, DIM, PRINT, LIST, PLOT, UNPLOT, FOR, GOSUB. Algumas vezes durante avaliação de função.
5	Não há mais espaço na tela. CONT continuará espaço limpando a tela.	PRINT, LIST, PLOT, UNPLOT.
6	Excesso aritmético. Os cálculos levaram a um número maior do que 10^{38}.	Qualquer aritmética.
7	RETURN sem um GOSUB correspondente.	RETURN.

8	Você tentou um comando INPUT não permitido.	INPUT
9	Instrução STOP executada. CONT não tentará reexecutar a instrução STOP.	STOP
A	Argumento inválido para certas funções.	SQR, LN, ASN, ACS
B	Número inteiro fora do limite. Quando requerido um número inteiro, o argumento do ponto flutuante é arredondado para o número inteiro mais próximo. Caso isso esteja fora de um limite apropriado, ocorrerá o erro B.	RUN, RAND, POKE, DIM, GOTO, GOSUB, LIST , PAUSE, PLOT, UNPLOT , CHR$, PEEK, USR.
	Para acesso a array veja o indicador 3.	Acesso array.
C	O texto do argumento (string) de VAL não forma uma expressão numérica válida.	VAL
D	(i) Programa interrompido por BREAK.	No final de qualquer instrução ou em LOAD, SAVE, LPRINT, LLIST , ou COPY.
	(ii) A linha de INPUT começa com STOP.	
E	Não usado.	
F	O nome fornecido para o programa é uma string vazia.	SAVE.