Curso de Programação BASIC e Operação CP-200

EDITELE

Curso de Programação BASIC e Operação CP-200

2.ª Edição
1982

EDITELE
Editora Técnica Eletrônica Ltda.
São Paulo – Brasil

Editele - Editora Técnica Eletrônica Ltda.
Av. Eng.º Luís Carlos Berrini, 1168 - 5.º andar
04571 - São Paulo - SP - Brasil

Cx. Postal 30141
Impresso no Brasil - Printed in Brazil

artes gráficas guaru s/a.
Impresso nas Oficinas de Artes Gráficas Guaru S/A - Rodovia Presidente Dutra, km. 214 - Fone: 209-6311 - Bonsucesso - Guarulhos.

Apresentação

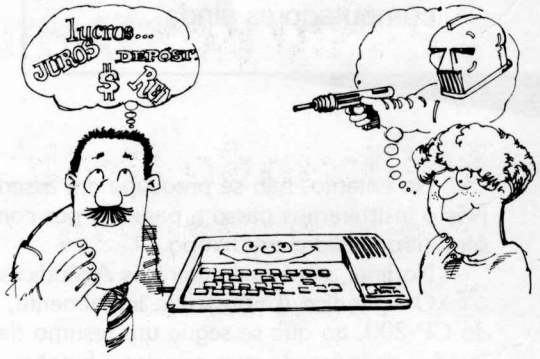
O mundo à nossa volta está sofrendo constantes avanços tecnológicos e, em conseqüência, caminhando a passos largos para uma nova era.

Uma característica peculiar dessa evolução é a possibilidade que o homem moderno dispõe de utilizar-se de uma ampla variedade de equipamentos eletrônicos, incluindo-se, entre eles, os computadores.

Estes transformaram, em poucos anos, a realidade de uma série de atividades humanas, de tal forma que a Informática, sem dúvida, tenderá a abranger, cada vez mais, a ciência manejada pelo homem da atualidade.

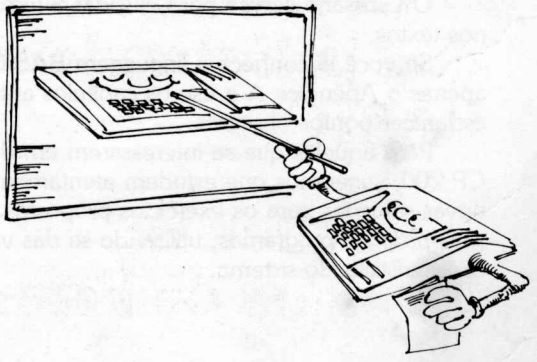
Um computador pessoal é, por assim dizer, o resultado dessa evolução tecnológica que, através da miniaturização dos circuitos, produziu um computador bastante compacto, porém com múltiplos recursos funcionais — tal como o seu CP-200.

Seu uso estende-se de pequenos programas didáticos a cálculos científicos, estando ainda capacitado para elaborar programas administrativos, controles contábeis e bancários, sem mencionar sua utilidade enquanto instrumento de lazer — onde você poderá evidenciar toda sua criatividade elaborando jogos eletrônicos.



Daí, pode-se concluir que uma infinidade de processos e atividades de seu cotidiano podem ser simplificados pelo uso do CP-200 e, a cada dia, você provavelmente irá descobrir uma nova área de aplicação para ele.

E, mesmo que você não esteja familiarizado com computadores, após algumas horas de treinamento, já estará apto a programá-lo, pois sua aprendizagem torna-se muito simples, em razão de sua operação extremamente simplificada — a qual está detalhadamente explicada neste livro.



Como utilizar este livro

Todas as informações necessárias à correta manipulação do CP-200 estão contidas neste livro.

Ele foi elaborado com o objetivo de auxiliá-lo na instalação, programação e operação do micro, ao mesmo tempo em que o ensina a explorar todos os recursos que ele apresenta — desde as operações mais simples às mais complexas.

Se você é um principiante, a maior parte deste livro foi escrita para você: auto-instrutivo, ele é dirigido aos iniciantes na área de computação e, à medida que oferece uma descrição completa dos componentes funcionais e estruturais do seu computador, coloca ao seu alcance um curso completo da linguagem BASIC.

Cada um de seus capítulos serve de informação aos que se seguem. Por isso, acompanhá-los em seqüência é essencial.

Siga com a leitura continuamente, sem ignorar os exercícios — eles levantam os pontos interessantes que são abordados no texto.

Se começar a notar que suas respostas não estão corretas, é possível que você não tenha estudado o texto atentamente. Nesse caso, reestude o texto antes de passar adiante.

Suposição

Você nada sabe sobre computadores ainda...

Método

Guia passo a passo

No entanto, não se preocupe em assimilar tudo de uma só vez. Nós o instruiremos passo a passo e, por conseguinte, cada conceito será introduzido a seu tempo.

No final, você encontrará os Apêndices e o Glossário.

O Apêndice A apresenta, inicialmente, as peculiaridades do CP-200, ao que se segue um resumo de suas características e um quadro sinóptico de suas principais funções, comandos e instruções.

O Apêndice B irá auxiliá-lo na interpretação dos Códigos de Erro que aparecem no decorrer da digitação dos dados, enquanto que o Apêndice C aborda o Código de Caracteres do micro.

O Glossário servirá para elucidar a terminologia técnica utilizada nos textos.

Se você já conhece a linguagem BASIC do computador, leia apenas o Apêndice A e use os capítulos anteriores apenas para esclarecer pontos obscuros.

Para aqueles que se interessarem em dominar completamente o CP-200, sugerimos que estudem atentamente este livro, testando novas soluções para os exercícios propostos, ou mesmo, elaborando seus próprios programas, utilizando-se das variações das características do sistema.

Introdução

Descrição do Computador

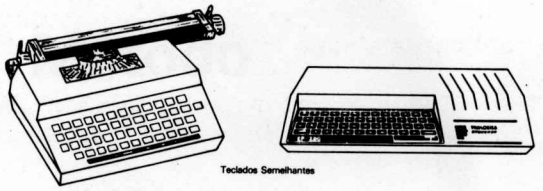
Veremos, nesse capítulo, como é constituído o seu computador e como ele faz para comunicar-se com você. Não entraremos em detalhes técnicos, mas, para aqueles que se interessarem, o capítulo XIII abordará alguns aspectos técnicos do CP-200.

Externamente, podemos ver as tomadas para ligação da antena da TV, do gravador cassete e do cabo de alimentação, além do teclado. Na parte interna está o computador propriamente dito.

Passemos primeiramente à parte externa.

O Teclado

O teclado do seu CP-200 é semelhante ao de uma máquina de escrever comum.



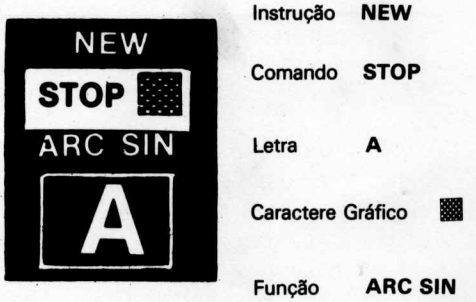
Teclados Semelhantes

Existem, porém, significativas diferenças entre ambos:

— No CP-200, quando você pressiona uma tecla qualquer, o computador emite um sinal sonoro (um "bip"). Esse sinal servirá como uma indicação de que a tecla foi pressionada corretamente;

— As teclas do seu computador são capazes de introduzir não apenas os caracteres normais de uma máquina de escrever, como também instruções, comandos, funções e símbolos gráficos.

Com apenas uma tecla você será capaz de gerar até cinco controles diferentes, como é mostrado na figura ao lado.



Observe que:

— Os caracteres em branco, no interior das teclas, representam os número de 0 a 9 e as letras, de A a Z. A esse conjunto denominamos **caracteres de texto.**

— Os sinais e palavras em vermelho são denominados **comandos.**

Para acioná-los deve-se proceder como que para as maiúsculas de uma máquina de escrever comum: pressione, simultaneamente, a tecla SHIFT e a que contiver o comando desejado.

— Existem ainda as **funções** (localizadas abaixo das faixas brancas) e os **caracteres gráficos** (no interior delas). O modo de utilizá-los será explicado a seu tempo. Assim, você não ficará sobrecarregado de informações que só serão úteis mais tarde.

A figura mostra o teclado do CP-200.

Observe-o, mas não se preocupe em tentar entender o significado de tudo que aparece nele.

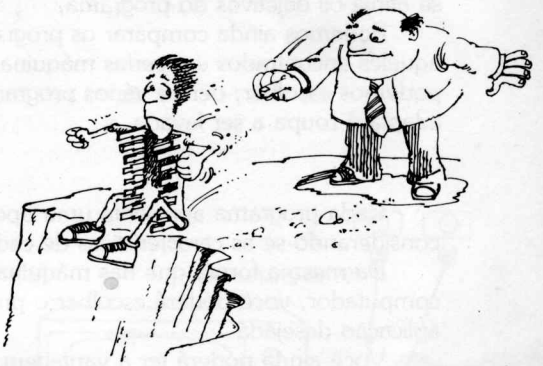
Cada item será detalhadamente esclarecido no momento oportuno.

Mais tarde, após conectar o computador ao seu TV, tente digitar qualquer coisa, sem importar-se com o teclado, para, apenas, acostumar-se a ele.

Os Componentes Internos

O computador é dividido, internamente, em três partes

principais:
— o microprocessador;
— os circuitos de entrada e saída e
— a memória de armazenamento.



O Microprocessador

O microprocessador é o componente fundamental do seu computador: é ele que executa todas as instruções com grande velocidade e confiabilidade.

Na realidade, tudo o que ele faz é obedecer as suas ordens, executando-as à risca. Por isso, as instruções devem ser claras e precisas para expressar exatamente o que você quer que ele faça.

Note, ainda, que essa comunicação entre o computador e você só é possível através dos dispositivos de Entrada e Saída.

Entrada e Saída (E/S)

Quando você digita uma mensagem qualquer no seu CP-200, está utilizando um dispositivo de Entrada de dados (informações) — no caso, o teclado.

Uma vez interpretada e executada pelo microprocessador, a informação final lhe é devolvida através de um dispositivo de Saída, o qual pode ser o vídeo de seu televisor.

Memória

As informações, dentro do computador, são guardadas em dispositivos especiais denominados Memórias.

O CP-200 é constituído de dois tipos de memória: **ROM** e **RAM.**

ROM — abreviatura de *Read Only Memory* (Memória Exclusiva de Leitura). É a parte da memória que armazena as informações, as quais instruem o microprocessador sobre como interpretar os dados introduzidos, como processá-los e devolvê-los de forma que o resultado corresponda ao esperado.

Essas informações não podem ser alteradas pelo microprocessador, mas apenas lidas (daí o nome).

RAM — abreviaturas de *Random Access Memory* (Memória de Escrita e Leitura). É a memória que permite ao microprocessador ter um acesso total à informação armazenada, quer seja para a sua leitura ou para introduzir modificações.

As informações armazenadas na RAM correspondem aos dados e programas que você introduz no computador. Mas, ao desligá-lo, essas informações são apagadas da memória.

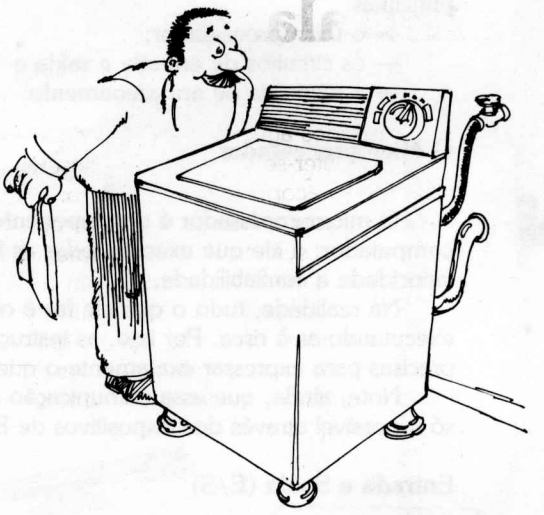
Cada informação (letra, número, sinal, instrução, etc) ocupa uma certa posição na memória. No CP-200 você terá 16 mil posições diferentes para armazenar informações. Em computação, denominamos estas posições de *byte*, portanto, dizemos que a capacidade de memória RAM do CP-200 é de 16 kilobytes ou, simplesmente, 16 kbytes.

Programa

Programa pode ser definido como uma seqüência de instruções ordenadas e inter-relacionadas que "dizem" ao computador o que deve ser realizado.

Dizemos que elas são ordenadas porque essas instruções são submetidas a uma ordem de execução; estão inter-relacionadas porque existe uma relação definida entre as instruções a fim de que se atinja os objetivos do programa.

Podemos ainda comparar os programas do computador com aqueles encontrados em certas máquinas de lavar. Nessas máquinas podemos escolher, dentre vários programas, aquele que mais se adapta à roupa a ser lavada.



Cada programa apresenta uma operação diferente considerando-se as características de cada tipo de roupa.

Da mesma forma que nas máquinas de lavar, no seu computador, você poderá escolher o programa ideal para o tipo de aplicação desejada.

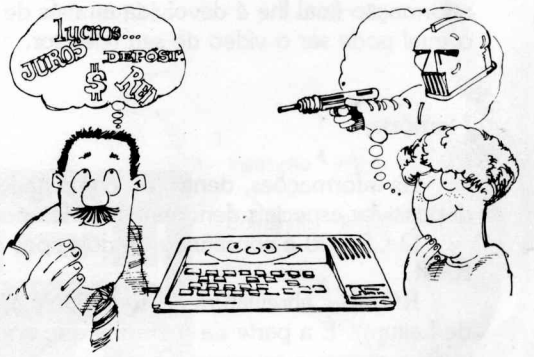
Você ainda poderá ter a vantagem de elaborar seus próprios programas. Seguindo esse livro, você irá aprender como fazer isso.

Linguagem

Para "dizer" ao computador o que ele deve fazer é necessário que seja estabelecida uma comunicação entre você e o computador. Por sua vez, essa comunicação só será possível através de uma linguagem comum aos dois.

Dentre as diversas linguagens "faladas" pelos computadores, podemos destacar a FORTRAN, ALGOL, COBOL e BASIC.

A linguagem escolhida para o CP-200 foi o BASIC, por se tratar de uma linguagem simples e eficiente, voltada justamente para os principiantes em computação.



BASIC é a abreviatura de *Beginner's All-purpose Simbolic Instruction Code* (Código Simbólico de Instrução, Universal, para Principiantes).

Fundamentando-se no vocabulário básico da língua inglesa, o Basic utiliza-se de instruções, comandos e funções de fácil compreensão, sendo, por isso, própria para principiantes — o que permitirá a você um rápido aprendizado das técnicas de programação.

Instalação

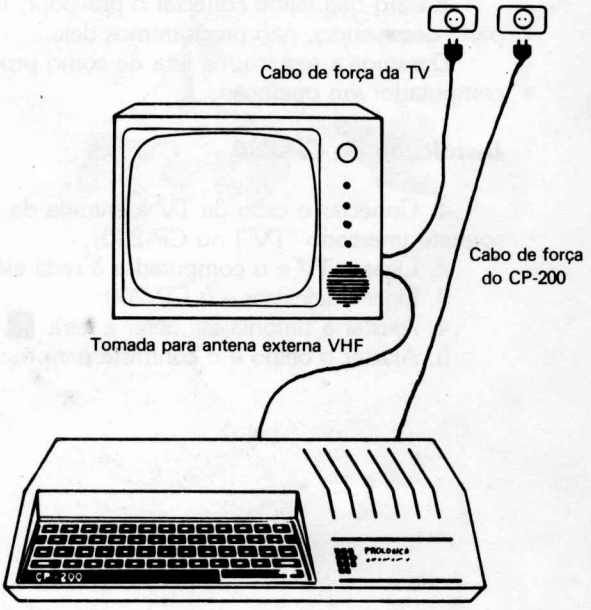
Passemos agora à instalação do seu computador.

Para obter-se o máximo do desempenho que ele pode oferecer, você deverá acoplá-lo a um televisor e a um gravador.

O televisor servirá para você acompanhar as informações digitadas, o resultado do processamento e os códigos de erro, e será imprescindível para a operação com o CP-200.

Poderá ser um aparelho comum, preto e branco, ou a cores.

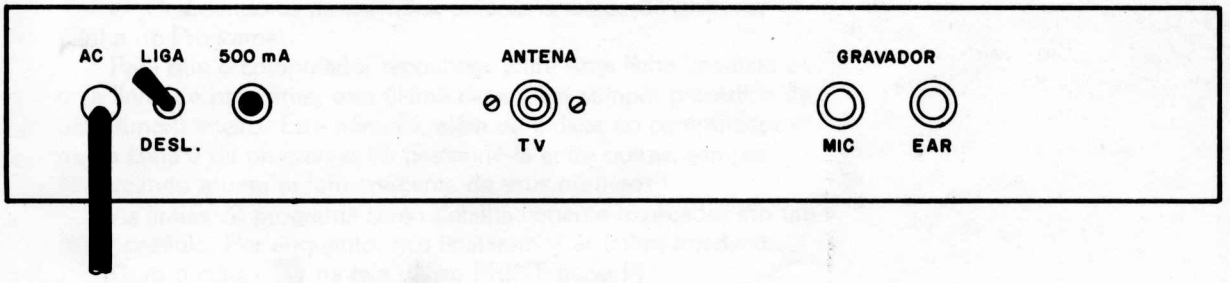
O gravador será utilizado para gravar programas do computador para a fita, ou mesmo para copiá-los de volta para a tela. Lembre-se ainda de que você poderá, também, introduzir no gravador programas de fitas pré-gravadas.



Montagem

Primeiramente, conecte o cabo para televisor na entrada da antena do seu aparelho de TV e na tomada atrás do computador (marcada "TV").

Ligue o televisor e o CP-200 à rede elétrica local, verificando antes se a tensão local corresponde à do computador.



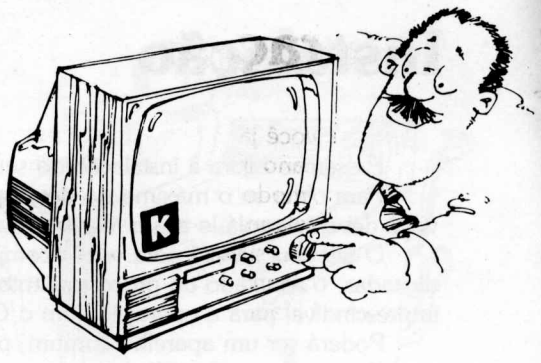
Agora o seu computador já está em funcionamento, contudo, você não terá nenhuma indicação disto até que ajuste a imagem do seu televisor.

Ajuste da Imagem

Coloque o seletor da TV no canal 2 ou 3 VHF e ajuste a sintonia até obter, no canto inferior esquerdo da tela, a letra K dentro de um retângulo.

Tente obter a melhor imagem possível através do controle da sintonia e, então, através dos controles de brilho e contraste, ajuste a letra K no melhor ponto de visibilidade. Se sentir alguma

dificuldade em ajustar a imagem com a letra K, pressione uma tecla qualquer, como P, por exemplo. Assim, uma palavra na tela facilitará mais os ajustes.



Com a imagem nítida na tela, seu computador está pronto para o uso.

É claro que faltou conectar o gravador, mas, para a primeira parte desse curso, não precisaremos dele.

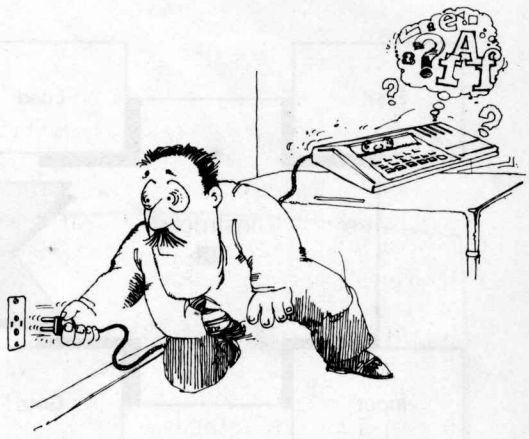
Daremos a seguir uma lista de como proceder para colocar seu computador em operação.

Instalação do CP-200

1. Conectar o cabo de TV à entrada da antena do TV e ao soquete (marcado "TV") no CP-200;
2. Ligar o TV e o computador à rede elétrica;
3. Ligar o televisor e o CP-200;
4. Ajustar a sintonia até obter a letra K na tela;
5. Ajustar o brilho e o contraste para melhorar a imagem.

Operação

Agora você já tem o seu computador instalado e pronto para operar. No cano inferior esquerdo da tela deve estar o cursor K indicando o modo de introdução de instruções. Caso você já tenha pressionado algumas teclas e digitado alguma coisa, e não saiba como apagá-la, simplesmente desligue o computador, tornando a ligá-lo após alguns segundos. Este procedimento não prejudicará os ajustes já realizados.



O cursor K indica que você pode introduzir instruções no computador para que ele as execute. Porém, como "dizer" ao computador o que ele deve fazer? Computadores não "entendem" português!

A linguagem BASIC

Entre as diversas linguagens utilizadas pelos computadores uma se destaca pela sua simplicidade e eficiência: a linguagem BASIC.

Como já foi falado alguma coisa a respeito do BASIC no capítulo I, neste capítulo daremos algumas noções de como fazer para que o computador execute instruções simples e daremos, ainda, algumas dicas de como corrigir erros de digitação.

As instruções em BASIC são introduzidas linha a linha, cada uma delas seguida pelo acionamento da tecla "ENTER" (em português, introduzir).

Ao introduzir-se uma linha, o CP-200 poderá processá-las:

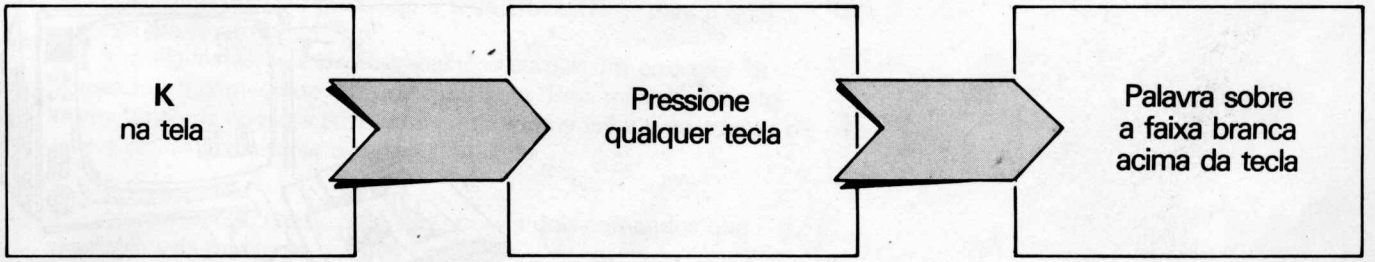
— Imediatamente, após pressionar-se a tecla ENTER (Linha Imediata), ou

— Guardando-as na memória para uma execução posterior (Linha de Programa).

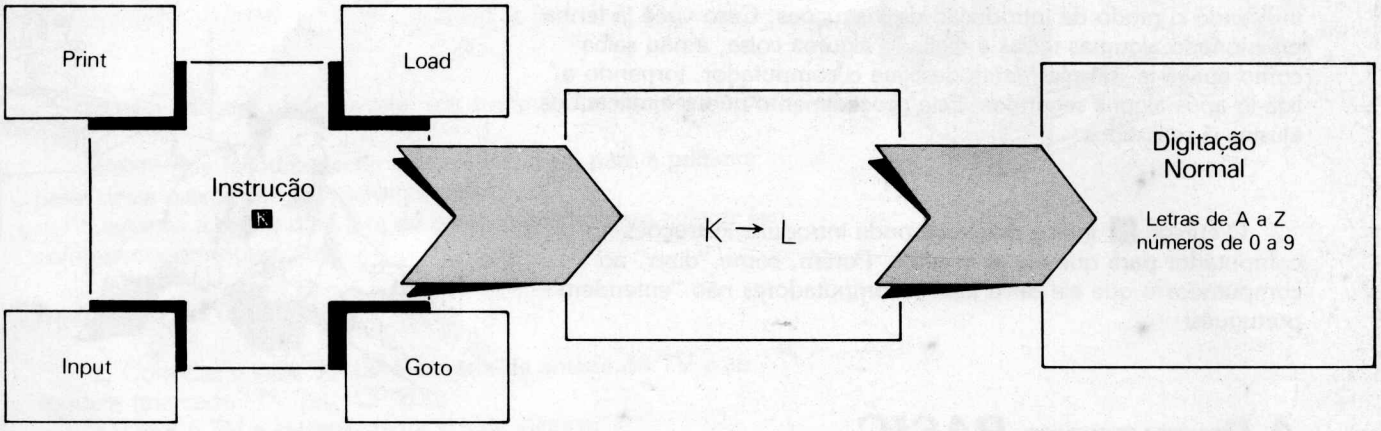
Para que o computador reconheça entre uma linha imediata e uma linha de programa, esta última deverá ser sempre precedida de um número inteiro. Este número, além de indicar ao computador que a linha é de programa, irá posicioná-la entre outras, sempre obedecendo a uma ordem crescente de seus números.

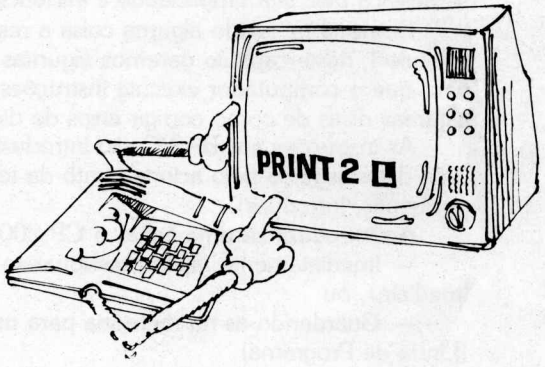
As linhas de programa serão detalhadamente explicadas em um outro capítulo. Por enquanto, nos limitaremos às linhas imediatas.

Com o cursor K na tela, digite PRINT (tecla P):

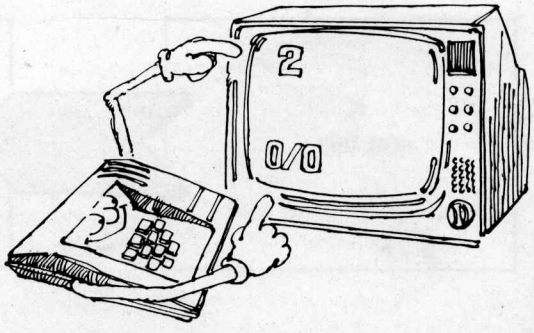


Com PRINT na tela, o cursor muda do modo K para o modo L, indicando que a próxima tecla a ser pressionada colocará na tela uma letra ou número. Pressione, agora, o número 2. Na tela estará a instrução:



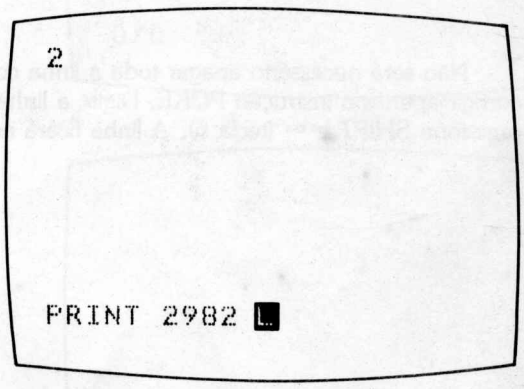


Esta linha na tela significa para o computador o mesmo que "imprima o número dois" (PRINT, em português, significa imprima). Se agora você pressionar ENTER, o computador entenderá (como a linha não começa por número) que deve executar a ordem imediatamente, e **imprimirá** no canto superior da tela, o número 2.

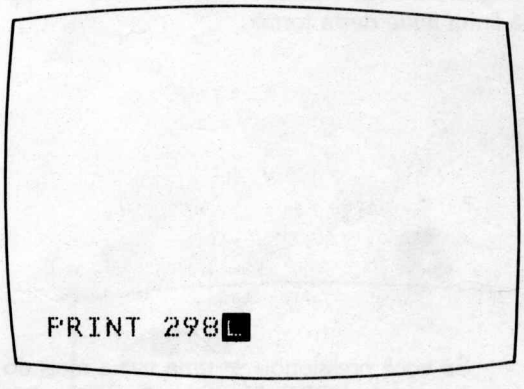


Se você não errou ao pressionar as teclas até agora, a tela ficará como é mostrado acima. Mas não se preocupe com o 0/0 na tela. Trata-se de um código que o CP-200 utiliza para informar que você introduziu a linha corretamente. Isto será explicado detalhadamente nos próximos capítulos. Por enquanto, continue operando como se o código 0/0 fosse a letra K.

Porém, todos nós, às vezes, cometemos pequenos erros. Os que surgirem durante a digitação de uma linha podem ser facilmente corrigidos. Para exemplificar, digamos que você queira mandar o computador imprimir o número 1982 e, acidentalmente, digite PRINT 2982. Na tela aparecerá:



Você pode corrigir essa linha usando o comando DELETE (em português, cancelar). Para usá-lo, basta manter pressionada a tecla SHIFT e pressionar a tecla correspondente ao DELETE (0). Feito isso, aparecerá na tela:



Repita a operação até que o cursor fique logo após a instrução PRINT. Nesse ponto você já pode introduzir o número correto e prosseguir com a execução da linha.

Se você não pressionar SHIFT e DELETE simultaneamente, aparecerá um zero (0) na tela após o número 2.

Por isso, tente repetir a correção.

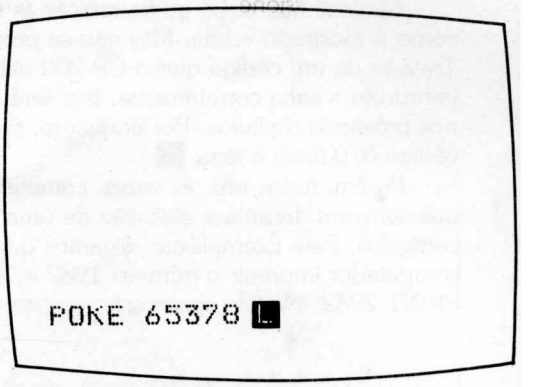
Pressione SHIFT, certificando-se de ter ouvido um "bip", ao mesmo tempo em que pressiona a tecla DELETE — para a qual corresponderá um novo "bip".

Em alguns casos é muito trabalhoso corrigir um erro que foi cometido no começo de uma extensa linha. Pelo método descrito acima teríamos que apagar praticamente toda a linha para corrigir o erro e digitar novamente o restante da linha.

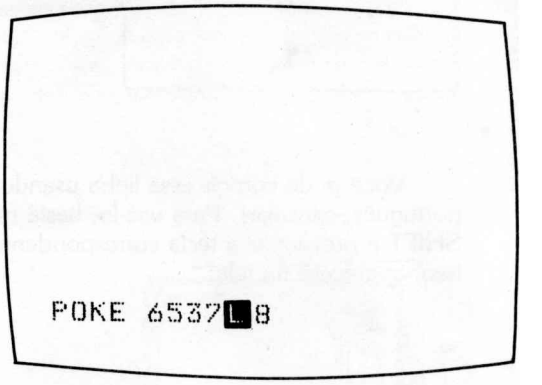
Visando esses casos, o CP-200 possui dois comandos que resolvem este problema. São os comandos ← e →

Suponhamos que você quisesse mandar o CP-200 imprimir o número 65378 mas digitasse, por engano, a instrução POKE, ao invés da instrução PRINT.

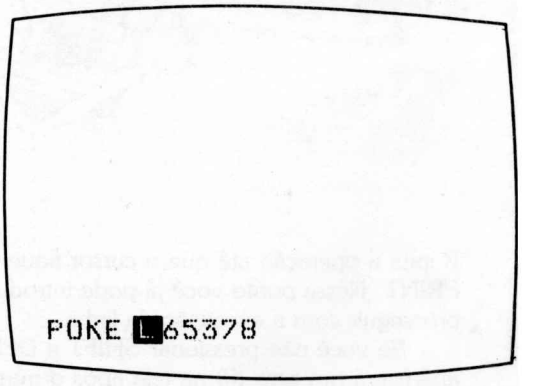
Tente digitar e corrigir a mensagem da figura abaixo:



Não será necessário apagar toda a linha com DELETE para corrigir apenas a instrução POKE. Digite a linha errada e então pressione SHIFT e ⇐ (tecla 5). A linha ficará assim:

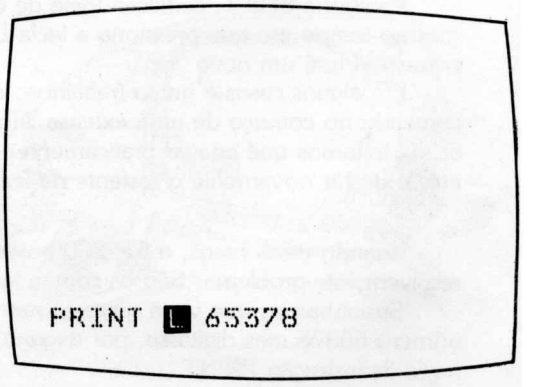


mantenha SHIFT pressionada e acione ⇐ mais quatro vezes até que a linha fique desta forma:

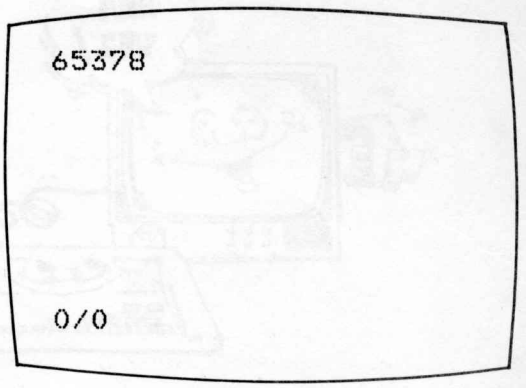


Se você pressionou ⇐ uma vez a mais do que devia, o cursor estará antes de POKE. Nesse caso, use o comando ⇒, que move o cursor para a direita, e coloque-o na posição adequada.

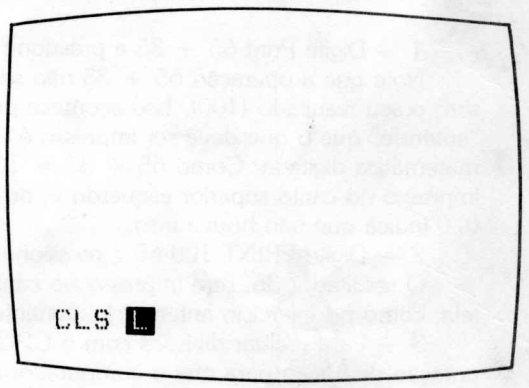
Com o cursor imediatamente após a instrução a ser corrigida, o procedimento para a correção é normal, ou seja, através do comando DELETE. Então mãos à obra! Pressione SHIFT juntamente com DELETE e, em seguida, a instrução desejada, no caso, PRINT. A linha na tela ficará assim:



Agora pressione ENTER e a linha será executada, aparecendo na tela:



Há ainda uma outra instrução que modifica o conteúdo da tela. É a instrução CLS. Esta sigla é abreviatura de *CLear Screen* que significa **Apagar a Tela.** Digite CLS e ENTER e veja que qualquer informação impressa na tela será apagada, ficando apenas o código 0/0 no canto inferior esquerdo. É importante notar que CLS é uma instrução completa, portanto não se deve digitar nada antes ou depois dessa instrução. A linha imediata fica simplesmente

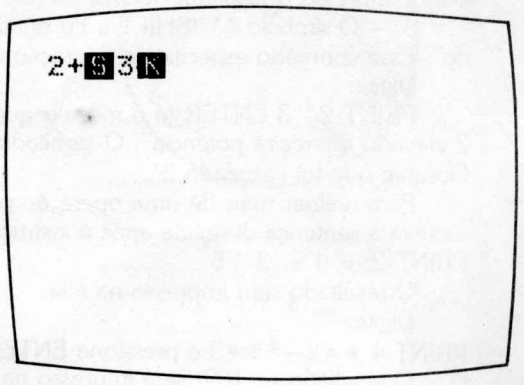


Erros de Sintaxe

Digite 2 + 3 e pressione ENTER.

O que aconteceu?

Você perceberá que não há resultado na parte superior esquerda da tela, mas, como você pode notar, o cursor S apareceu em sua parte inferior:



K indica que você omitiu a instrução (não digitou PRINT);

S indica que houve um erro de sintaxe. No caso, o computador "entendeu" o número 2 como a indicação de uma linha de programa, uma vez que foi introduzido sem uma instrução prévia.

S virá imediatamente após o elemento errado. No exemplo acima, ele aparece após o sinal + indicando que não poderá realizar a operação (soma) se, antes, você não acionar uma instrução.

1 — Digite Print 65 + 35 e pressione ENTER.

Note que a operação 65 + 35 não será impressa na tela, mas, sim, o seu resultado (100). Isso acontece porque o computador "entende" que o que deve ser impresso é o resultado da sentença matemática digitada. Como 65 + 35 = 100, o resultado, 100, foi impresso no canto superior esquerdo e, no canto inferior, o código 0/0 indica que não houve erro.

2 — Digite PRINT 100-65 e pressione ENTER.

O resultado, 35, será impresso no canto superior esquerdo da tela, como no exercício anterior, juntamente com o código 0/0.

3 — Para realizar divisões com o CP-200, utilizamos uma notação de fração para que o computador reconheça a operação. A barra / é utilizada para este fim.

Digite PRINT 3/2 e pressione ENTER.

Será impresso na tela o resultado (1.5).

Note que o computador utiliza o ponto para separar os inteiros dos decimais, ao invés de vírgula.

4 — Para indicar multiplicação, utiliza-se o símbolo * Portanto, se quisermos imprimir o resultado da multiplicação de 35 por 105 devemos digitar:

PRINT 35*105

e será impresso o resultado (3675) na tela.

5 — O símbolo ** (SHIFT e H) significa "elevado à potência de". Esse comando especifica a operação de potenciação.

Digite:

PRINT 2**3 ENTER (é o mesmo que "imprima o resultado de 2 elevado a terceira potência". O resultado (8) será impresso na tela. Observe que foi calculado 2^3.

Para realizar mais de uma operação por vez, simplesmente escreva a sentença desejada após a instrução PRINT. Por exemplo:

PRINT 2 + 4 — 3 * 5

O resultado será impresso na tela.

Digite:

PRINT 4 **2—5**3 e pressione ENTER.

O resultado (—109) será impresso na tela.

Primeiros Passos

Imprimindo o Resultado de Uma Expressão Matemática

Como você deve ter visto nos exercícios do capítulo anterior, podemos fazer com que o computador resolva equações que envolvam diversas operações matemáticas, imprimindo o resultado na tela. Porém, devemos saber elaborar essas equações, para dizermos ao computador exatamente o que queremos que ele resolva.

Em uma sentença matemática existe uma ordem correta para executarmos estas operações. Em sentenças como 1 + 2 * 3, em que existem operações matemáticas diferentes, calcula-se em primeiro lugar, o resultado da multiplicação 2*3, para, depois, somá-lo a 1, embora a ordem da sentença seja primeiro a soma e depois a multiplicação. Chamaremos a isso de grau de prioridade.

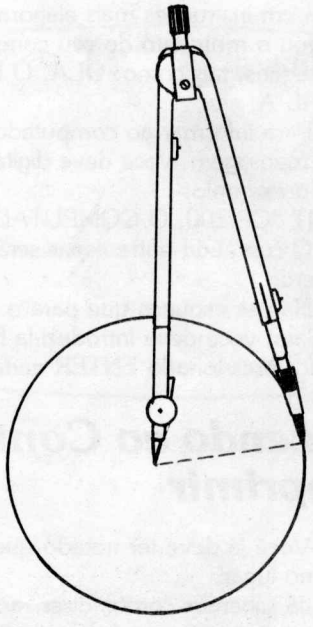
Além da prioridade da multiplicação sobre a soma, há várias outras regras que estabelecem essa relação de prevalência. Relacionamos, abaixo, as principais:

Grau de prioridade

Maior	Expressões entre parênteses
↓	Funções Matemáticas (SIN, COS, SQR, etc.)
↓	Potenciação (**)
↓	Multiplicação (*) e Divisão (/)
Menor	Soma (+) e Subtração (—)

Dentro do mesmo grau de prioridade, o computador realiza as operações na ordem em que aparecem (da esquerda para a direita).

Vejamos um exemplo típico:



$$\text{Raio} = \frac{\text{Circunferência}}{2\pi}$$

Para se calcular o raio de uma circunferência divide-se o comprimento da mesma por duas vezes π (lê-se **pi**). π é a relação entre o diâmetro e comprimento da circunferência e equivale a aproximadamente 3,14159265.

Podemos digitar essa fórmula de duas maneiras distintas:

PRINT C/2*π ou PRINT C/(2*π)

Para que se introduza o número π, o cursor deve estar indicando o modo de função [F], pois o CP-200 considera o π como uma função. Para introduzir o cursor no modo de função, deve-se pressionar as teclas SHIFT e FUNCTION simultaneamente seguidas de ENTER. Assim o cursor [F] aparecerá na tela. Após ter introduzido a função desejada (no caso, π) o computador retorna ao modo anterior ([L]), indicando que você pode digitar a mensagem.

Se você introduzir as linhas, verá que o resultado será um código de erro (2/0). Isso ocorreu porque você não informou ao computador o valor de C.

Tente novamente atribuindo valores à circunferência como, por exemplo:

1) Raio = 20/2*3.14 ou PRINT 20/2*π, que resulta em 31.4.

2) Raio = 20/(2*3.14) ou PRINT 20/(2*π), que resulta em 3.185.

Como podemos deduzir, um dos resultados está errado. O raio de uma circunferência não pode ser maior do que o seu comprimento. Portanto, o modo (1) de escrever a equação está incorreto. De fato, se tormarmos por base a tabela de prioridades, veremos que do modo L as operações têm a mesma prioridade e serão realizadas na seqüência em que aparecem. O resultado será, então, dado pela multiplicação de π pela divisão 20/2π.

No modo (2) a situação é inversa, pois a multiplicação entre parênteses tem prioridade sobre o restante da equação. Deste modo, teremos a seqüência correta de execução.

Imprimindo mensagens

Até agora você usou a instrução PRINT apenas para imprimir números, quer sejam em instruções diretas (como PRINT 2), quer sejam em instruções mais elaboradas (como PRINT 2**4/8). Chegou o momento do seu computador começar a imprimir mensagens, tais como: OLÁ; O RAIO DA CIRCUNFERÊNCIA É IGUAL A, etc.

Para informar ao computador que você quer que seja impressa uma mensagem, você deve digitá-la entre aspas (tecla SHIFT e P). Veja o exemplo:

PRINT "CP-200, O COMPUTADOR PESSOAL".

O conteúdo entre aspas será impresso no canto superior esquerdo.

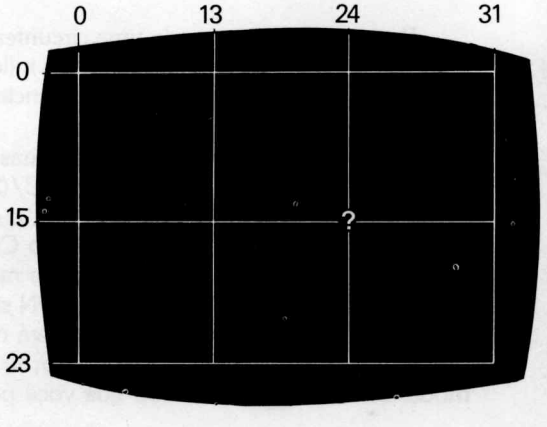
Não se esqueça que para o computador executar essa linha imediata, você deve introduzi-la finalizando com ENTER. Enquanto não for pressionado ENTER nada ocorrerá.

Dizendo ao Computador Onde Imprimir

Você já deve ter notado que estamos imprimindo sempre no mesmo lugar.

Já sabemos como "dizer" ao CP-200 o que ele deve imprimir, mas precisamos, em alguns casos, informálo também onde a mensagem ou o número deve ser impresso.

Por isso a tela do seu televisor foi dividida em linhas e colunas, a fim de que pudéssemos ter um sistema de referência para a impressão. O CP-200 divide a tela em 24 linhas contendo, cada uma, 32 colunas verticais. As linhas são nemeradas de 0 a 23 e as colunas de 0 a 31. Veja abaixo o esboço da tela. O ponto de interrogação localizado no esboço está colocado na posição 15, 24, respectivamente, linha e coluna.



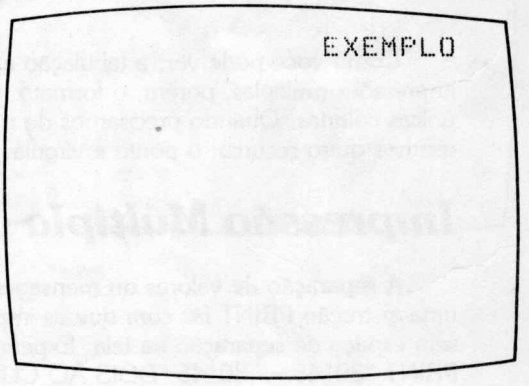
Isso significa que o nosso ponto de interrogação está posicionado na coluna de número 24, dentro da linha 15.

Usa-se a instrução PRINT AT — PRINT seguida da função AT (tecla C) — para definir o ponto onde a impressão começa. Assim, para imprimir o ponto de interrogação do exemplo acima, digite:
PRINT AT 15, 24; "?"

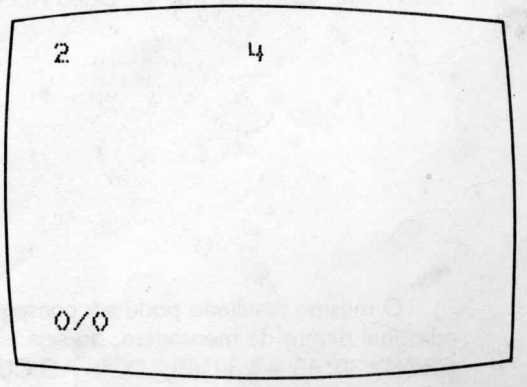
Deve-se entender essa mensagem como "imprima na linha 15, coluna 24, o ponto de interrogação". O ponto e vírgula deve sempre ser colocado após as coordenadas de impressão (linha e coluna). Essa pontuação serve para separar as coordenadas do conteúdo a ser impresso. Podemos definir uma forma geral da ilustração PRINT AT, como a que segue:
PRINT AT linha, coluna; "conteúdo"

Existem casos em que não precisamos definir a linha e a coluna onde a impressão deve começar, mas apenas a coluna. É o caso, por exemplo, da montagem de tabelas, onde precisamos imprimir os valores dispostos em colunas previamente determinadas. Nas máquinas de escrever encontramos teclas de tabulação para este fim.

O CP-200 também possui uma função de "tabulação", definida por TAB (tecla P com cursor em F). O modo de utilização é o mesmo que foi usado para AT (tecla C com cursor em F). Com a diferença de que TAB, não exige a introdução do número de linha, pois a tabulação é realizada sempre dentro da linha atual de impressão. Experimente a linha imediata abaixo:
PRINT TAB 10; "EXEMPLO"



Também com TAB precisamos usar o ponto e vírgula para separar as informações sobre **onde** imprimir (coluna) da informação sobre **o que** imprimir (mensagem ou número). Se for usada uma vírgula para fazer essa separação, a função TAB simplesmente não opera. Nesse caso, a tabulação é realizada pela vírgula.



Como a vírgula pode tabular?

Em uma instrução PRINT, quando usamos vírgulas para separar valores ou mensagens a ser impressos, ocorre uma tabulação automática para que não ocorra que sejam impressos sem um espaçamento — o que tornaria difícil a leitura.

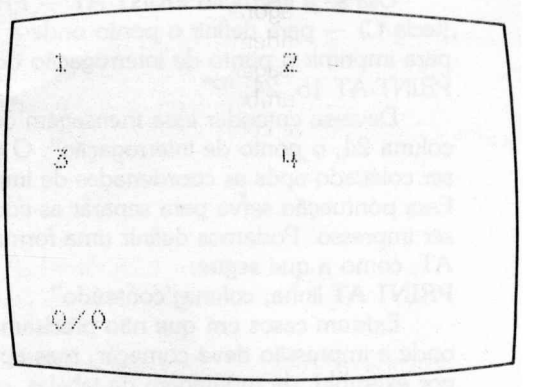
Digite a linha:
PRINT 1 + 1, 2 + 2 e pressione ENTER

Na tela será impresso:

A tabulação automática é realizada sempre nas colunas 0 e 16, o que significa que, se usarmos uma instrução PRINT com vírgulas para imprimir mais de dois valores ou mensagens, a tabulação do terceiro será na coluna 0 da linha seguinte, e assim por diante. Como ilustração, introduza a linha:

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

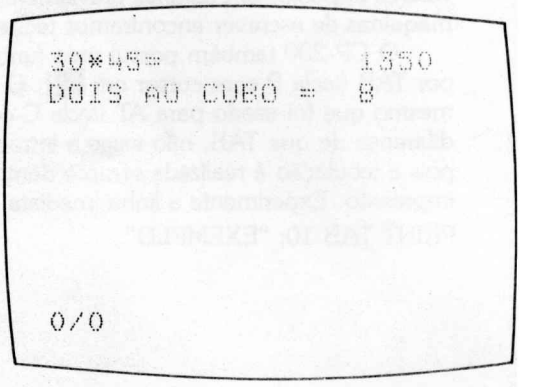
Na tela aparecerá:



Note que esta característica do CP-200 permite a impressão de vários dados com uma única linha digitada. Tente esta linha:

```
PRINT "30*45=", 30*45, "DOIS AO CUBO =",2**3
```

A tela ficará assim:

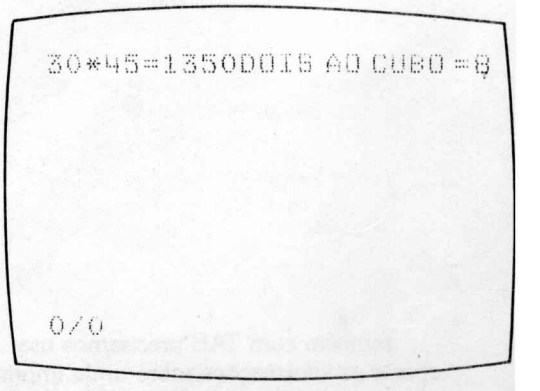


Como você pode ver, a tabulação automática é muito útil para impressões múltiplas, porém, o formato na tela fica limitado a duas únicas colunas. Quando precisamos de mais colunas por linha, usamos outro recurso: o ponto e vírgula.

Impressão Múltipla

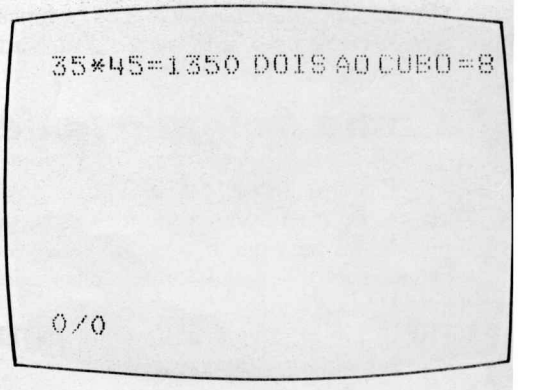
A separação de valores ou mensagens por ponto e vírgula ém uma instrução PRINT faz com que as impressões sejam consecutivas, sem espaço de separação na tela. Experimente esse exemplo:

```
PRINT "30*45=";30*45;"DOIS AO CUBO = ";2**3
```



Para melhorar o resultado anterior, introduzimos espaços entre os valores e as mensagens:

```
PRINT "30*45=";30*45;"↓";"DOIS AO CUBO=";2**3
```



O mesmo resultado pode ser conseguido colocando-se o espaço adicional dentro da mensagem, ou seja:

```
PRINT "30*45 =";30*45;" DOIS AO CUBO = ";2**3
```

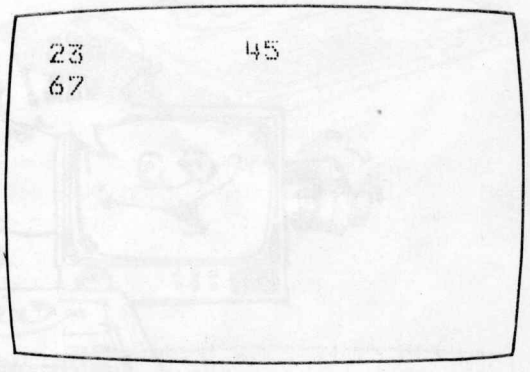
Resumo da Instrução PRINT

Faremos agora uma breve recapitulação do que vimos nesse capítulo. Aprendemos que é possível fazer o computador imprimir números, mensagens, e o resultado de equações matemáticas. Aprendemos também que podemos definir a posição onde a impressão deve começar e como proceder para tal. Completando, verificamos que o computador pode imprimir várias mensagens e/ou números com uma única instrução, combinação várias características.

Relacionamos abaixo vários exemplos para fixar melhor os conceitos.

Exemplos:

PRINT "MENSAGEM"	imprime "MENSAGEM
PRINT 2	imprime 2
PRINT 23+47	imprime 70
PRINT 23;45;67	imprime 234567
PRINT 23;"+";45;"—";67	imprime 23+45—67
PRINT 23, 45, 67	imprime 23 45



PRINT TAB 20;23	imprime 23 a partir da coluna 20
PRINT AT 17,20;23	imprime 23 a partir da coluna 20 da linha 17

Funções Matemáticas

Além das operações matemáticas, o CP-200 apresenta uma série de FUNÇÕES intrínsecas que permitirão a você realizar os mais complexos cálculos aritméticos.

Elas estão localizadas logo abaixo de cada tecla, representadas sempre por seus nomes ou símbolos respectivos.

Para imprimi-la você deverá, primeiramente, pressionar as teclas SHIFT e ENTER (FUNCTION) simultaneamente; esse procedimento mudará o cursor para o modo F de operação, permitindo a você digitar a função desejada. Uma vez introduzida a FUNÇÃO, o curso voltará para o modo anterior (L).

Daremos abaixo uma relação das mais conhecidas, mas você poderá encontrar a lista completa de todas as FUNÇÕES deste computador no Apêndice A.

SGN x — Função "sinal" fornece —1 para x negativo, 0 para x=0 +1 para x positivo.

ABS x — Fornece o valor absoluto ou o módulo de forma que o resultado será sempre um argumento positivo.

SIN x — Fornece o seno do argumento (este deve estar em radianos)

COS x — Fornece o cosseno do argumento (este deve estar em radianos)

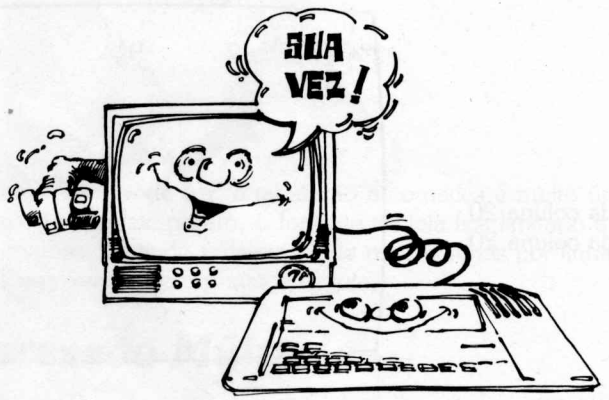
TAN x — Fornece a tangente do argumento (este deve estar em radianos)

ASN x — Fornece o arco-seno do argumento (este deve estar em radianos)

ACS x — Fornece o arco-cosseno do argumento (este deve estar em radianos)
ATN x — Fornece o arco-tangente do argumento (este deve estar em radianos)
LN x — Fornece o logaritmo natural (base e = 2,718281828)
EXP x — Fornece a função exponencial (e^x = 2,718...^{x})
SQR x — Fornece a raiz quadrada
INT x — Fornece a representação inteira do argumento. Esta função arredonda sempre para : INT 4.7 = 4
PI — π = 3,1415927 (PI não tem argumento)
RND — Fornece um número qualquer aleatório) que denominaremos RaNDômico.

Todas as FUNÇÕES do CP-200 quando em operação, devem obedecer a um critério de prioridade (veja o capítulo anterior) em que somente "perdem" para as expressões entre parênteses.

NOTA: Embora a FUNÇÃO π (PI) matematicamente não seja considerada uma função, ele assumirá essa a característica no CP-200, por uma questão de disposição do teclado.



1. Imprima "CENTRO" no centro da tela (Lembre-se de que são 32 colunas e 24 linhas).

```
PRINT AT 12,13:"CENTRO"
```

2. Imprima "LINHA UM" na primeira linha, coluna 16.

Isso pode ser feito de três formas:

```
PRINT TAB 16;"LINHA UM" ou
PRINT TAB 8, "LINHA UM" ou, ainda
PRINT, "LINHA UM"
```

3. Imprima no começo da segunda linha, a mensagem "SEGUNDA LINHA", usando vírgulas.

```
PRINT,, "SEGUNDA LINHA"
```

4. Imprima, no meio da segunda linha, a mensagem do exercício anterior.

```
PRINT,,, "SEGUNDA LINHA"
```

5. Imprima "FIM" no canto inferior direito da tela.

```
PRINT AT 21,29; "FIM"
```

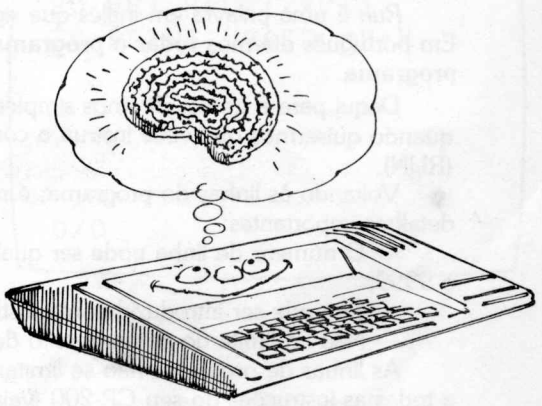
6. Imprima "NORTE" no centro da primeira linha, "SUL" no centro da última, "OESTE" no início da linha central (12), e "LESTE" no final da mesma linha.

```
PRINT AT 0,14; "NORTE"; AT 21, 15; "SUL"; AT 12,0; "OESTE";
AT 12,27; "LESTE"
```

Programação Elementar

Você já deve estar bastante familiarizado com o uso da instrução PRINT em linhas imediatas.

Como já foi visto, as linhas imediatas são executadas assim que a tecla ENTER é acionada, e não há como executá-las mais de uma vez. Para introduzirmos linhas com instruções as quais o computador possa executar diversas vezes sem "esquecê-las", precisamos de um outro tipo de linha: **a linha de programa.**



Programas Elementares

A principal característica de um computador é a sua grande velocidade na realização de cálculos ou tarefas rotineiras. Trabalhos repetitivos e demorados podem ser **programados** em um computador para que este, toda vez em que for necessário, execute, passo a passo, uma lista de tarefas pré-determinadas.

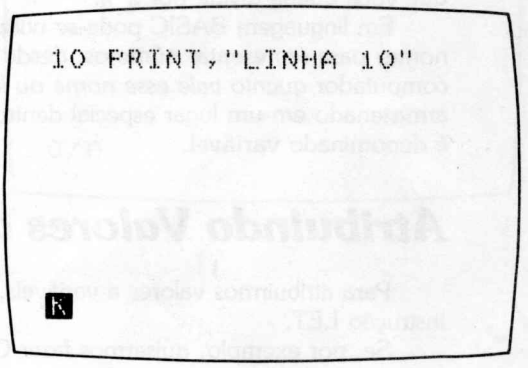
Na linguagem BASIC, estabelecemos a seqüência correta de execução quando adicionamos um número às linhas de instrução. Essas linhas numeradas são chamadas de **linhas de programa**, como foi visto no capítulo I.

Ao introduzirmos uma linha de programa, o computador irá armazená-la na memória e só a executará quando você lhe ordenar através da instrução RUN.

Faça o exemplo a seguir:

10 PRINT, "LINHA 10" ENTER

Aparecerá na tela:

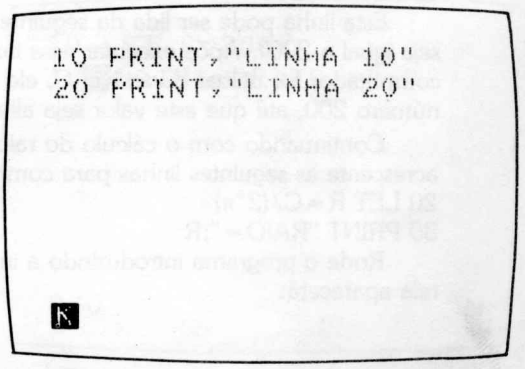


Note que a instrução não foi executada, mas apenas copiada na primeira linha da tela, enquanto que na parte inferior da tela apareceu o cursor K .

Digite agora o seguinte:

20 PRINT, "LINHA 20"

A tela mudará para:



A segunda linha também não foi executada, e agora você tem duas linhas armazenadas na memória com uma seqüência de execução definida. Digite RUN e veja o que acontece:

```
                LINHA 10
                LINHA 20

 0/20
```

As linhas foram executadas e o código de erro significa que a última linha executada foi a linha 20 e não houve erro.

Run é uma palavra em inglês que significa **correr, fazer andar.** Em português dizemos **rodar o programa** ao invés de **correr o programa.**

Daqui para a frente, diremos simplesmente **rodar o programa** quando quisermos que você instrua o computador para executá-lo (RUN).

Voltando às linhas do programa, é necessário esclarecer alguns detalhes importantes:

— O número de linha pode ser qualquer número inteiro entre 1 e 9999;

— Só pode ser introduzida uma instrução por linha;

— Não há limite do comprimento de uma linha de programa.

As linhas de programa não se limitam a instruções PRINT, mas a todas as instruções do seu CP-200 (veja no teclado). Iremos, pouco a pouco, descrevendo a aplicação de cada instrução.

Para a próxima instrução abordada usaremos novamente o exemplo do raio da circunferência. Você está lembrado da linha imediata.

PRINT C/(2*π)? Pois bem, quando você tentou introduzir esta linha, ocorreu um erro, pois o computador não sabia qual era o número que você queria dividir por 2*π.

Em linguagem BASIC pode-se utilizar letras ou até mesmo nomes para representar números, desde que seja *"dito"* ao computador *quanto vale esse nome ou letra.* Esse valor *atribuído* é armazenado em um lugar especial dentro da memória RAM, o qual é denominado **variável.**

Atribuindo Valores a Variáveis

Para atribuirmos valores a variáveis, em BASIC, usamos a instrução LET.

Se, por exemplo, quisermos fazer C igual a 200, devemos digitar:

10 LET C=200

Esta linha pode ser lida da seguinte maneira: "Faça com que C seja igual a 200". Após executar essa instrução, toda a vez em que o computador for utilizar a variável C, ele a "entenderá" como sendo o número 200, até que este valor seja alterado.

Continuando com o cálculo do raio da circunferência, acrescente as seguintes linhas para completar o programa:

20 LET R=C/(2*π)
30 PRINT "RAIO=";R

Rode o programa introduzindo a instrução RUN e ENTER. Na tela aparecerá:

```
 RAIO =31.38099


 0/30
```

Listagem e Alteração de Programas

Note que na memória do computador, o programa de impressão de LINHA 10 e LINHA 20, foi substituído pelo cálculo do raio.

Você pode obter uma listagem do programa que está na memória usando a instrução LIST. Digite a linha imediata:
LIST ENTER
e o programa aparecerá na tela da seguinte forma:

```
10 LET C=200
20 LET R=C/(2*PI)
30 PRINT "RAIO =";R


0/0
```

Esta linguagem mostra que as linhas: 10 PRINT, "LINHA 10" e 20 PRINT, "LINHA 20" foram substituídos, respectivamente, pelas linhas:
10 LET C = 200 e 20 LET R = C/(2*π)

Isso é feito automaticamente, ao se digitar uma linha de programa. O conteúdo da nova linha substitui o da anterior. Tente, por exemplo, introduzir o seguinte:
20 ENTER

A tela mudará para:

```
10 LET  C=200
30 PRINT "RAIO =";R


0/0
```

A linha 20 desapareceu, o que significa que podemos **apagar** linhas de programa apenas digitando o número da linha e ENTER. Nesse caso, você está substituindo uma linha por outra que, na realidade, não possui conteúdo algum.

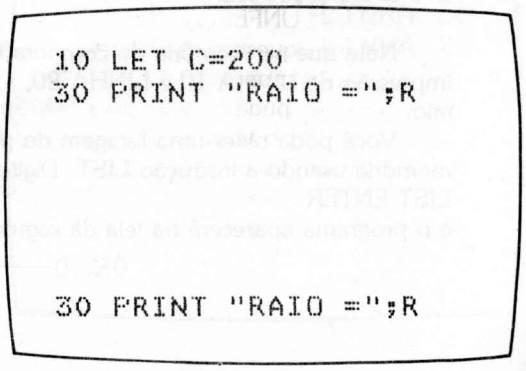
Outra forma de se alterar o conteúdo de uma linha de programa é corrigindo diretamente o texto, sem a necessidade de digitar novamente a linha.

O CP-200 possui uma função que permite essa correção de linhas já introduzidas: é a instrução EDIT. Em programação chamamos a essas correções de **edição de programa**. Como ilustração vamos alterar a linha 30 do programa que está na memória. Para tanto, utilize as teclas ↑ e ↓ até que o cursor ▒, fique localizado à frente da linha 30, como abaixo:

```
10 LET C=200
30▒PRINT "RAIO =";R


0/0
```

Pressione agora as teclas SHIFT e EDIT (tecla 1) simultaneamente e a linha 30 será copiada na parte inferior da tela, juntamente com o cursor ▌, para que sejam efetuadas as alterações necessárias:



O que queremos é que o programa imprima a mensagem RAIO = seguida do valor calculado para o raio. Para isso devemos fazer o computador imprimir o resultado da sentença $C/(2*\pi)$, então, a linha 30 deve ficar:

30 PRINT "RAIO = ";C/(2*π)

Será necessário, então, deslocar o cursor para a direita até o final da linha e deletar a variável R, pois ela não será mais necessária.

30 PRINT "RAIO = ";R▌

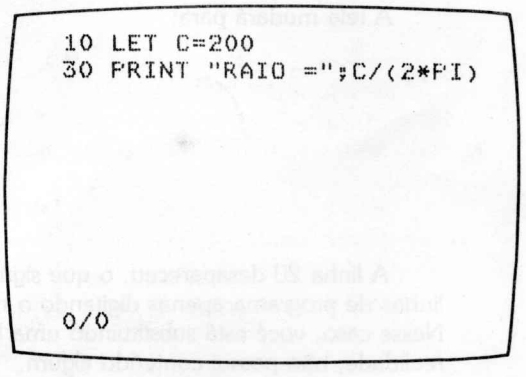
Acione DELETE (lembre-se do terceiro capítulo)

30 PRINT "RAIO = " ▌

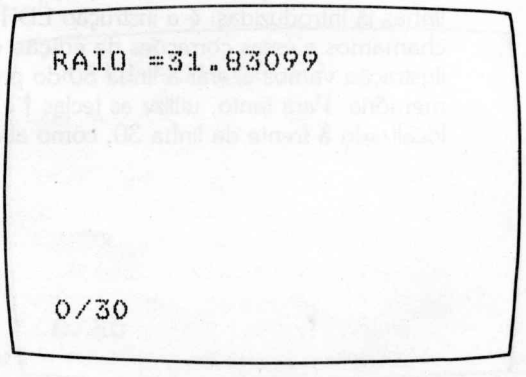
Digite agora a fórmula correta:

30 PRINT "RAIO = ";C/(2*π) ▌

Com a linha corrigida (editada), basta pressionar ENTER e ela retornará ao seu lugar, no alto da tela, alterando o conteúdo anterior.



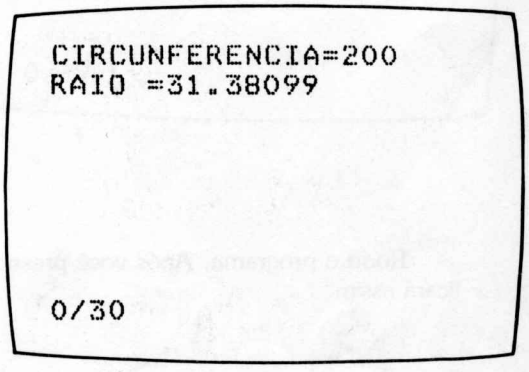
Execute o programa (RUN ENTER); a tela deve ficar assim:



Para melhorar o programa, podemos fazer com que o CP-200 forneça também o comprimento da circunferência na tela. A instrução necessária nesse caso é:
PRINT "CIRCUNFERÊNCIA = ";C

Por uma questão de lógica matemática, já que primeiro definimos o comprimento da circunferência antes de calcular o raio, seria bom que pudéssemos imprimir primeiro a variável C, para depois imprimir o valor do raio (variável R). Com esse intuito devemos escolher o número de linha de modo que a variável C seja impressa depois de receber um valor (LET C = 200) e antes da impressão do resultado final (PRINT "RAIO = ";C...). Analisando o programa, vemos que os números de linha são 10 e 30, portanto, podemos usar qualquer número inteiro entre esses dois (11, 12,...28,29). Escolhemos o número 20. Então digite:
20 PRINT "CIRCUNFERÊNCIA = ";C

Execute o programa:



Resumindo, para editar uma linha de programa você deve:

— Utilizar as setas ↑ e ↓ para posicionar o cursor na linha a ser alterada;

— Pressione SHIFT e EDIT (tecla 1) para que a linha seja copiada na parte inferior da tela;

— Utilizar as setas → e ← para posicionar o cursor à frente do texto a ser eliminado (DELETE) ou no ponto onde você pretende introduzir informação;

— Ao terminar a edição da linha, faça uma minuciosa verificação no texto definitivo. Confirmado o texto, pressione ENTER, o que introduzirá a linha na memória colocando o novo texto;

— Caso você queira que as alterações realizadas em uma linha não sejam consideradas pelo computador, pressione EDIT novamente e a linha original será copiada novamente. Esse procedimento deve ser realizado antes de se introduzir as alterações (ENTER), pois isso modificará a linha armazenada.

Mais sobre Variáveis

Como já foi dito, as variáveis podem armazenar (representar) qualquer valor, desde que associem valores a nomes. Isso permite que chamemos pelo nome da variável, toda vez em que for necessário representar um valor. Com as instruções que vemos até aqui, torna-se muito trabalhoso alterar os valores de uma dada variável, pois teríamos que modificar a instrução LET toda vez que um novo valor fosse usado. O ideal seria uma instrução que permitisse a introdução de valores sem precisar alterar o programa.

INPUT é a resposta. *Input* em português significa **introduzir** e é exatamente isso que o computador espera que seja realizado ao

executar essa instrução: ele aguarda que seja introduzido um valor e o colocará em uma variável especificada na instrução.

Como isso é feito? Veja no exemplo abaixo:

10 INPUT C

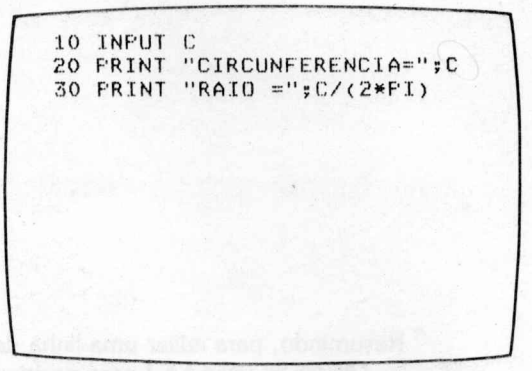
Ao executar essa linha, o CP-200 irá parar o processamento, esperando que seja introduzido um valor numérico. Na parte inferior da tela aparecerá o cursor L indicando que deve ser feita a introdução de letras ou, no caso, números. Vejamos como isso funciona na prática:

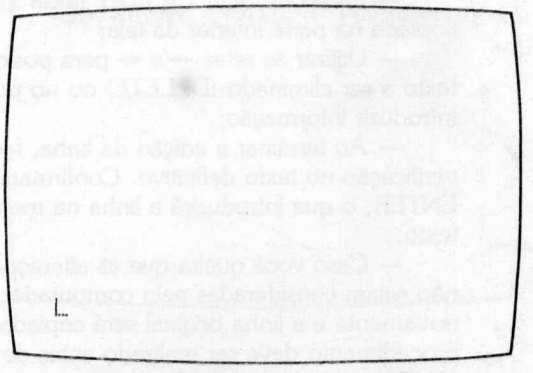
Digite:

10 INPUT 0 (ENTER)

Na tela aparecerá:

Rode o programa. Após você pressionar RUN ENTER, a tela ficará assim.

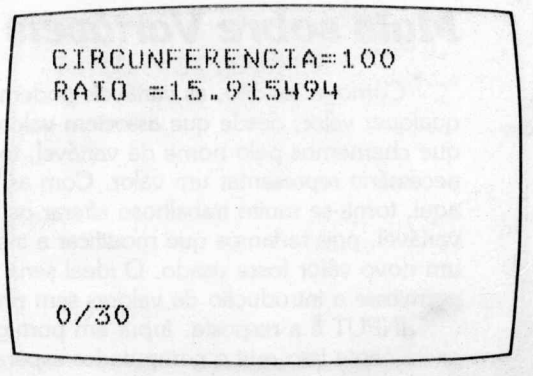




É essa indicação que permite que você digite um valor que será atribuído à variável C. Digite um valor qualquer, como por exemplo:

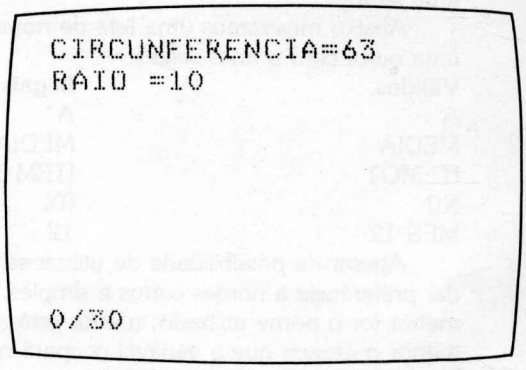
100 ENTER

Depois disso, o computador completará o programa da forma normal, imprimindo o valor introduzido e, abaixo dele, o resultado.



Note que você pode introduzir um novo número; para tanto basta rodar o programa novamente. Tente agora, introduzindo o número 63, por exemplo:

RUN
63 ENTER



Isso pode ser repetido quantas vezes for necessário. Daí o nome *variável*, pois o conteúdo de uma variável pode ser mudado a qualquer momento.

Até aqui temos usado nomes de variáveis com apenas uma letra (C e R).

Porém, os nomes de variáveis numéricas podem ter qualquer comprimento.

Para continuar, vamos introduzir um novo programa no CP-200. Desta vez vamos apagar o programa que está na memória e digitar, linha a linha, o novo programa.

Quando quisermos apagar um programa de memória, não basta apenas digitar CLS (veja no capítulo 3), para eliminá-lo. Tente este comando:

CLS ENTER

O programa desaparecerá da tela, porém, se você pressionar ENTER, o programa retornará novamente ao vídeo. Isso indica que esse comando não serve para eliminar o programa da memória.

Em BASIC, quando queremos introduzir um novo programa, ao mesmo tempo em que se apaga o antigo, usamos a instrução NEW (novo, em inglês). Digite:

NEW ENTER

Verifique agora, pressionando ENTER, se o programa ainda está na memória.

Não aparecerá nada na parte superior da tela, indicando que não há programa algum na memória. A instrução NEW serve para apagar o programa da memória, liberando espaço para novos programas. Após a execução dessa instrução, o cursor aparecerá no modo K .

Mas, vamos ao novo programa. Ei-lo abaixo:

```
10 INPUT A
20 INPUT B
30 LET MEDIA = (A + B)/2
40 PRINT"MEDIA = " ; MEDIA
```

Esse programa pede a introdução de dois números e calcula a média aritmética entre eles. Note que usamos a palavra MEDIA para definir um valor numérico. As variáveis que representam valores numéricos são denominadas variáveis numéricas. Mais à frente veremos outros tipos de variáveis.

As variáveis possuem alguns recursos especiais quando usadas na instrução LET. Veja o exemplo:

LET A = A + B

Nesse caso, a variável de nome A aparece duas vezes em uma

única instrução. Essa instrução é executada da seguinte maneira: em primeiro lugar, o CP-200 soma os valores de A e B e, então, coloca o resultado na variável A, novamente.
O nome de uma variável numérica pode ter qualquer comprimento e combinação de letras e números, porém deverá sempre começar por uma letra.

Abaixo mostramos uma lista de nomes válidos para variáveis e uma outra com nomes ilegais.

Válidos	**Ilegais**
A	A*
MEDIA	MEDIA:
ITEMO1	ITEM-1
X0	0X
MES 12	12

Apesar da possibilidade de utilizar-se nomes longos, devemos dar preferência a nomes curtos e simples. Lembre-se de que quanto menor for o nome utilizado, menor será a possibilidade de erro e menor o espaço que a variável ocupará na memória (sim, pois o nome também é armazenado).

Uma característica das variáveis é que elas não se limitam a armazenar números. Elas também podem armazenar mensagens literais, como nomes, endereços, observações, etc. Em programação, existe um termo especial para mensagens literais. Este termo é *string*. Um string é uma seqüência de caracteres sem significado matemático para o computador, mas com um significado literal para o operador.

As variáveis armazenam apenas um tipo de informação: numérica ou string.

Para diferenciar uma variável numérica de uma variável string usamos colocar o sinal de cifrão ($) após o nome das variáveis *string*.

O CP-200 aceita nomes de variáveis string com apenas um caractere de comprimento. Abaixo estão alguns nomes válidos para variáveis string.

A$ X$ C$

Vamos a um exemplo para fixar melhor estas informações. Para pagar o programa da memória, juntamente com todas as informações armazenadas, podemos simplesmente pressionar as duas teclas RESET ao mesmo tempo. Essas teclas fazem com que o computador *esqueça* tudo o que está armazenado e recomece suas operações como se estivesse sendo ligado naquele instante.

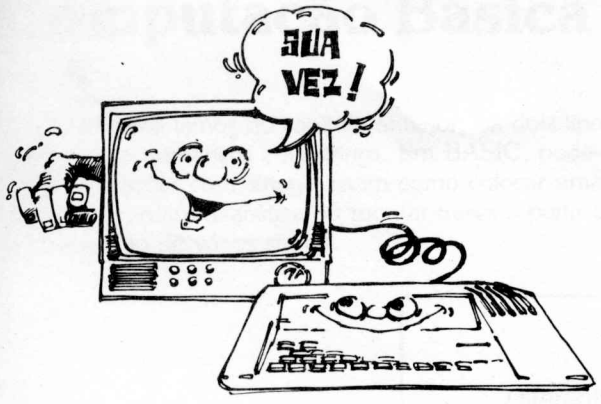
Com o computador inicializado, digite o seguinte programa:

```
10 PRINT "DIGITE UMA FRASE"
20 INPUT A$
30 PRINT "ESTA E A MENSAGEM:" ; A$
```

Rode-o e introduza uma frase qualquer, como "COMPUTADOR PESSOAL CP-200"

Depois de pressionar ENTER, o computador colocará na tela:

```
DIGITE UMA FRASE
ESTA E A FRASE: COMPUTADOR PESSOAL CP-200
```



Nota: ENTER deve ser pressionado após as digitações, para que o CP-200 reinie o processamento.

1. Elabore um programa que converta graus Fahrenheit em graus Celsius.

Solução proposta:

```
10 INPUT F
20 LET C=5*(F-32)/9
30 PRINT F
```

2. Rode o programa do primeiro exercício e calcule a temperatura equivalente a:

a. 59 graus Fahrenheit
b. 77 graus F
c. O ponto de fusão da água (32°F)
d. O ponto de ebulição da água (212°F)

As respostas devem ser, respectivamente:

a. 15°C
b. 25°C
c. 0°C
d. 100°C

Nesta questão sabemos que tipo de informação devemos digitar quando o computador pára a execução à espera de novos dados. Quando lidamos com programas mais sofisticados, não queremos que o computador apenas pare a execução, mas que indique o tipo de informação que deve ser introduzido. Isso pode ser realizado por uma instrução PRINT imediatamente antes à INPUT.

3. Inclua uma linha com INPUT no programa de conversão para indicar a introdução da temperatura correta. Para obter uma listagem do programa a partir de um resultado da tela, simplesmente pressione ENTER.

Você pode adicionar uma linha como a seguinte:

```
5 PRINT "PROXIMO VALOR EM GRAUS CENTÍGRADOS"
```

4. Qual a diferença e a semelhança entre a instrução CLS, a instrução NEW e a tecla RESET.

Semelhança: Todas limpam a tela.

Diferenças: CLS apaga apenas a tela;
NEW apaga a tela e os programas na memória;
RESET apaga todas as informações do computador, fazendo-o reiniciar sua operação.

5. Elabore um programa que faça as seguintes tarefas:

a) Permita a introdução do nome da cidade e da temperatura local em °F;

b) Calcule a temperatura em graus centígrados;

c) Imprima, com identificação, o nome da cidade e a temperatura local, em graus centígrados.

Soluções Propostas

```
10 PRINT "QUAL CIDADE?"
20 INPUT C$
30 PRINT "QUAL A TEMPERATURA (FAHR.)?"
40 INPUT TF
50 LET TC = 5* (TF-32)/9
60 PRINT "EM"; C$;",";TC;"GRAUS CELSIUS"
```

6. Escreva um programa que calcule a economia de um automóvel. Para isso, ele deve:

a) aceitar a introdução de quantos quilômetros foram percorridos;

b) pedir a informação de quantos litros foram gastos nesse percurso;

c) pedir a introdução do preço por litro de combustível;

d) pedir, por fim, o tipo de combustível usado (gasolina, álcool, diesel);

e) calcular:

— o consumo (em Km/l) e

— o custo (em Cr$/Km)

f) imprimir as seguintes informações:

— A quilometragem percorrida;

— O total gasto de combustível;

— O consumo no percurso:

— O custo por quilômetro.

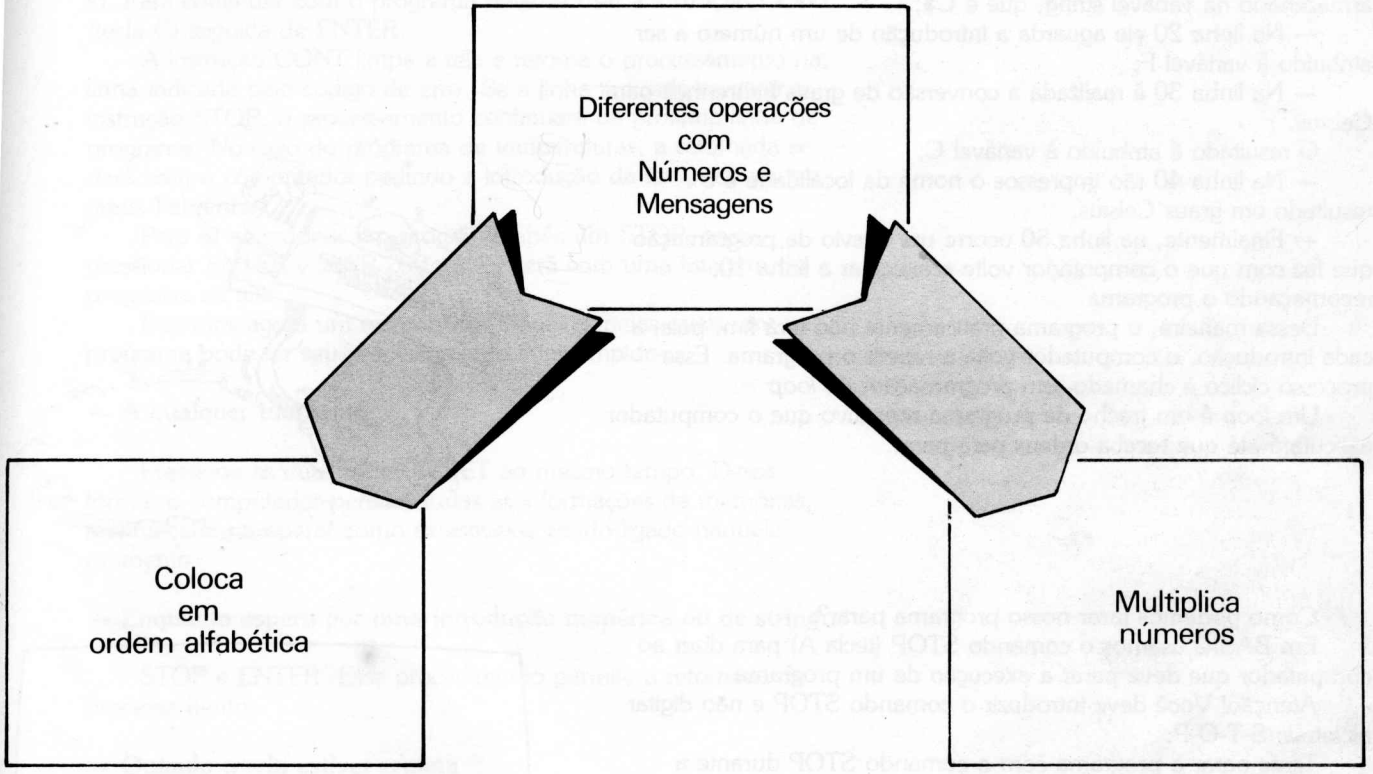
Solução sugerida:

```
10 PRINT "DIGITE A QUILOMETRAGEM"
20 INPUT KM
30 PRINT "QUANTOS LITROS?"
40 INPUT LT
50 PRINT "CUSTO POR LITRO?"
60 INPUT CUSTO
70 PRINT "QUAL COMBUSTÍVEL?"
80 INPUT C$
90 PRINT "FORAM PERCORRIDOS"; KM; "KM"
100 PRINT "FORAM GASTOS"; LT; "LITROS DE"; C$
110 PRINT "CONSUMO = "; KM/LT; "KM/L"
120 PRINT "CUSTO = CR$" ; (LT*CUSTO)/KM; "POR KM"
```

Note que a solução proposta é apenas uma das muitas variações possíveis da resposta. Você deve tentar elaborar outras soluções para os problemas propostos. Desta forma, estará exercitando os conhecimentos adquiridos.

Computação Básica

Conforme vimos no capítulo anterior, há dois tipos distintos de variáveis: as *numéricas* e as *strings*. Em BASIC, pode-se realizar certas operações com strings, assim como colocar uma lista de nomes em ordem alfabética ou montar frases a partir da concatenação de vários strings.



Foi visto ainda que as variáveis string podem ser colocadas em instruções de atribuição (LET) ou de introdução (INPUT) da mesma forma que as numéricas.

Quando num programa queremos processar diversos valores, torna-se necessário dotar esse programa de recursos de tal forma que, após apresentar os resultados, ele torne a pedir um novo valor para cálculo. Isso só será possível se o programa for repetido tantas vezes quantas forem necessárias.

Até aqui, para cada novo cálculo, usávamos digitar RUN ENTER para executar todo o programa novamente.

Para resolver esse tipo de problema, o BASIC do seu CP-200 possui uma instrução chamada GOTO (em português, "ir para"). Ao executar essa instrução, o CP-200 passa a realizar as instruções a partir da linha cujo número é indicado na instrução GOTO.
Exemplo:

```
100 GOTO 10
```

Nesse caso, o computador, após executar a linha número 100, passará a executar a linha número 10, e assim sucessivamente.

Tomemos como exemplo uma versão simplificada do programa de temperatura do exercício 5 do capítulo anterior:

```
10 INPUT C$
20 INPUT F
30 LET C=5*(F-32)/5
40 PRINT C$,C
50 GOTO 10
```

Note que usamos nesse exemplo duas variáveis com o mesmo nome: C e C$. No entanto, elas são diferenciadas pelo tipo de dado que contêm. O cifrão, assim, serve para definir uma nova variável — a qual denominamos C$.

Observe também que cada uma delas será armazenada em locais diferentes da memória.

Outro detalhe importante: quando o computador pára à espera de um string, na tela aparecerá o cursor L. entre aspas (" L. ").

O programa acima é executado da seguinte forma:

— Na linha 10 o computador espera por um nome a ser armazenado na variável string, que é C$;

— Na linha 20 ele aguarda a introdução de um número a ser atribuído à variável F;

— Na linha 30 é realizada a conversão de graus Fahrenheit para Celsius.

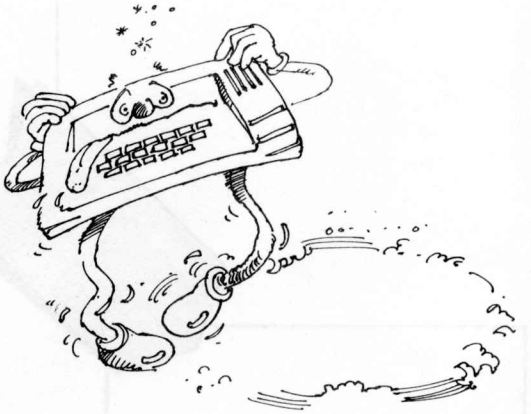
O resultado é atribuído à variável C;

— Na linha 40 são impressos o nome da localidade e o resultado em graus Celsius;

— Finalmente, na linha 50 ocorre um desvio da programação que faz com que o computador volte a executar a linha 10, recomeçando o programa.

Dessa maneira, o programa praticamente não terá fim, pois, a cada introdução, o computador volta a repetir o programa. Esse processo cíclico è chamado, em programação, de *loop*.

Um *loop* é um trecho de programa repetitivo que o computador executará até que receba ordens para parar.



Como podemos fazer nosso programa parar?

Em BASIC usamos o comando STOP (tecla A) para dizer ao computador que deve parar a execução de um programa.

Atenção! Você deve introduzir o comando STOP e não digitar as letras: S-T-O-P.

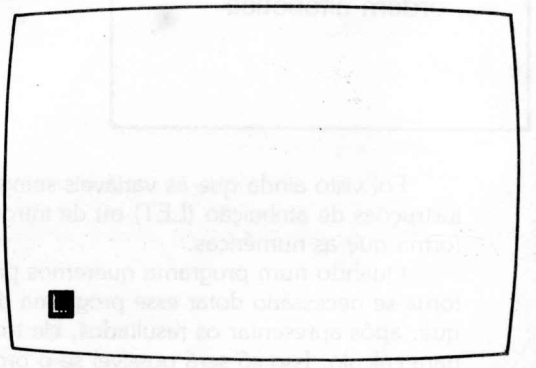
Tente parar o programa com o comando STOP durante a introdução do nome da cidade. Na tela, o cursor estará assim " L ". Se você simplesmente pressionar STOP ENTER, o computador considerará que foi introduzido o nome STOP para a cidade e imprimirá normalmente, continuando o programa.

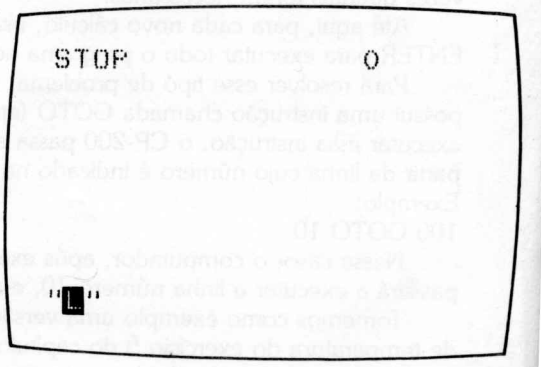
Tente fazer o programa parar da seguinte forma:

RUN ENTER

STOP ENTER (Não se esqueça de pressionar Shift e A para o comando STOP).

Na tela, com o cursor





Veja, o computador interpretou que, na cidade STOP, a temperatura estava em zero graus! Isso porque você introduziu o comando STOP entre aspas e, por isso, ele foi considerado como um string. Para evitar esses problemas, você deverá eliminar a primeira aspa com o comando DELETE, da seguinte forma:

Com o cursor "L" na tela, digite:

SHIFT DELETE

SHIFT STOP ENTER

O programa é paralisado e o computador fica esperando pelo próprio passo a ser dado: continuar com o programa ou abandoná-lo. Para continuar com o programa deve-se usar a instrução CONT (tecla C) seguida de ENTER.

A instrução CONT limpa a tela e retoma o processamento na linha indicada pelo código de erro. Se a linha indicada contiver a instrução STOP, o processamento continuará da próxima linha de programa. No caso do programa de temperaturas, a retomada se dará com o computador pedindo a introdução da temperatura em graus Fahrenheit.

Para se abandonar um programa após um STOP, basta pressionar ENTER e o CP-200 responderá com uma listagem do programa na tela.

Daremos agora um resumo das maneiras pelas quais um programa pode ter seu processamento interrompido:

— A qualquer momento

Pressione as duas teclas RESET ao mesmo tempo. Dessa forma, o computador perderá todas as informações da memórias, recomeçando a operar como se estivesse sendo ligado naquele momento.

— Enquanto espera por uma introdução numérica ou de string

STOP e ENTER. Esse procedimento permite a retomada de processamento.

— Quando a tela estiver repleta

O computador pára automaticamente quando não houver mais espaço para colocação de dados no vídeo. CONT limpará a tela e retomará o processamento.

— Durante o processamento (não é esperada uma introdução)

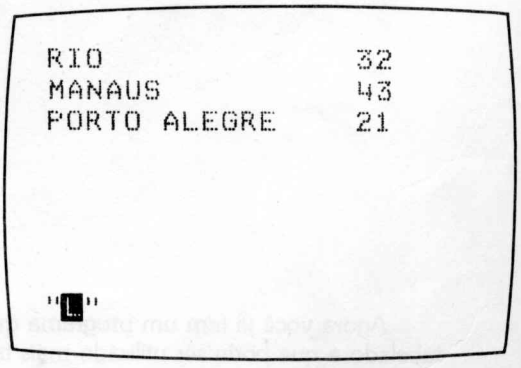
BREAK (tecla de espaço) opera de modo similar a STOP.

Mais à frente você aprenderá a fazer o programa parar por si próprio. Por enquanto, devemos nos preocupar com o modo pelo qual os dados estão sendo apresentados na tela.

Rode o programa introduzindo o nome de algumas cidades e respectivas temperaturas locais como, por exemplo:

RIO 90°F; MANAUS 110°F; PORTO ALEGRE 70°F

A tabela resultante terá algo parecida com:



Para melhorar a apresentação seria conveniente colocar um cabeçalho para definir o conteúdo de cada coluna. Sugerimos a seguinte linha:

```
5 PRINT "LOCALIDADE"; TAB 16; "GRAUS CELSIUS"
```

Para introduzir essa linha, pare o programa usando STOP, pressione ENTER e terá na tela a listagem do programa. Digite, então, a linha 5, terminando por ENTER e rode novamente o programa, repetindo os valores.

Você obterá esse resultado na tela:

```
LOCALIDADE      GRAUS CELSIUS
RIO             32
MANAUS          43
PORTO ALEGRE    21


"L"
```

A linha 5 será executada apenas no começo do programa, formando o cabeçalho. Após a impressão na linha 40, o computador retornará à linha 10, pedindo um novo nome. A linha 5, portanto, não faz parte do *loop*.

Uma outra melhoria no programa seria a indicação de que dado é esperado pelo computador quando aparece o cursor ■ na tela. Para isso devemos acrescentar duas linhas antes das duas instruções de introdução. Sugerimos:

```
 8 PRINT AT 21,0;"QUAL LOCALIDADE?"
18 PRINT AT 21,0;"QUAL A TEMP. FAHR?"
```

O programa final ficará:

```
 5 PRINT "LOCALIDADE"; TAB 16; "GRAUS CELSIUS"
 8 PRINT "QUAL A LOCALIDADE?"
10 INPUT C$
18 PRINT "QUAL TEMP. (FAHR) ?"
20 INPUT F
30 LET C=5*(F-32)/9
40 PRINT C$,C
50 GOTO 10
```

Rode o programa (RUN ENTER). O CP-200 perguntará:

```
LOCALIDADE GRAUS CELSIUS



QUAL A LOCALIDADE?
"■"
```

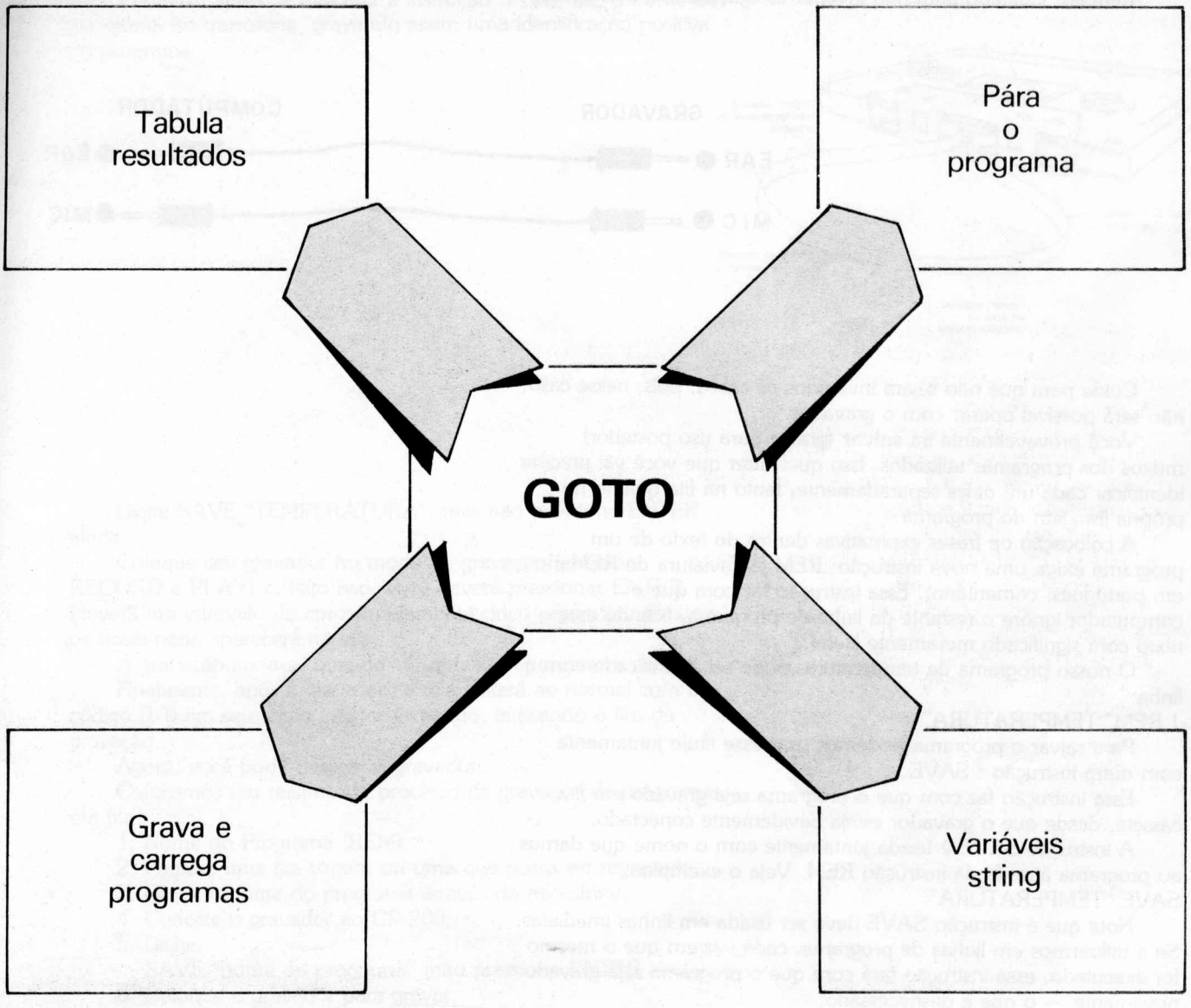
Depois de introduzido o local, aparecerá:

```


QUAL TEMP. (FAHR.)?
```

Agora você já tem um programa que fornece um resultado tabelado e que pode ser utilizado mais tarde, quando formos testar

novas características do CP-200. Depois de tanto trabalho na digitação desse exercício, seria bom que pudéssemos guardá-lo em algum lugar de onde pudéssemos tornar a copiá-lo quando necessário, sem a necessidade de uma nova digitação. Uma fita cassete é o lugar ideal.



Utilização de fita cassete. Como Arquivo de Programas e Dados

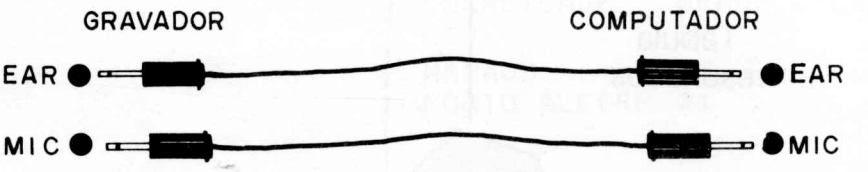
O CP-200, mediante a utilização de gravadores, permite armazenamento de informações, tais como programas, variáveis e certos parâmetros internos do microprocessador em fitas cassete.

Essas informações poderão, uma vez gravadas, serem mantidas por tempo indeterminado, formando um arquivo de programas e dados para o computador.

A conexão do gravador ao CP-200 é feita através de dois cabos de ligação que vêm juntamente com o computador.

Conecte um deles à tomada MIC do CP-200 (parte de trás) e à tomada MIC do gravador. Outro deve ser conectado à tomada EAR do CP-200 e à tomada de mesmo nome do gravador (ou à tomada MONITOR).

Atenção: Cuidado para não inverter as ligações dos dois cabos.



Cuide para que não sejam invertidos os cabos, pois, nesse caso, não será possível operar com o gravador.

Você provavelmente irá **salvar** (gravar para uso posterior) muitos dos programas utilizados. Isso quer dizer que você vai precisar identificar cada um deles separadamente, tanto na fita quanto na própria listagem do programa.

A colocação de frases explicativas dentro do texto de um programa exige uma nova instrução: REM (abreviatura de REMark; em português, comentário). Essa instrução faz com que o computador ignore o restante da linha de programa, ficando esse texto com significado meramente literal.

O nosso programa de temperaturas pode ser identificado com a linha:

```
1 REM "TEMPERATURA"
```

Para salvar o programa podemos usar esse título juntamente com outra instrução : SAVE.

Essa instrução faz com que o programa seja gravado em fita cassete, desde que o gravador esteja devidamente conectado.

A instrução SAVE é usada juntamente com o nome que damos ao programa através da instrução REM. Veja o exemplo:

```
SAVE "TEMPERATURA"
```

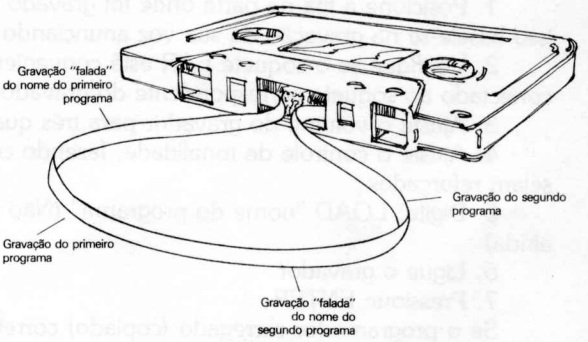
Note que a instrução SAVE deve ser usada em linhas imediatas. Se a utilizarmos em linhas de programa, cada vez em que o mesmo for executado, essa instrução fará com que o programa seja gravado novamente — o que é desnecessário.

Na prática podemos colocar uma instrução REM em qualquer ponto do programa, ou ainda, incluí-la diversas vezes no decorrer da programação; nesse caso, cada uma delas identificará uma parte distinta do programa. Veja o exemplo:

```
 1 REM "TEMPERATURA"
 5 PRINT "LOCALIDADE"; TAB 16; "GRAUS FAHR"
 8 PRINT AT 21,0; "QUAL LOCALIDADE?"
10 INPUT C$
18 PRINT "QUAL TEMP. (FAHR.)?"
20 INPUT F
25 REM***CONVERSAO***
30 LET C = 5*(F-32)/9
```

```
35 REM***SAIDA***
40 PRINT C$,C
45 REM*** RETORNO***
50 GOTO 5
```

Para a localização de um programa na fita é conveniente gravar uma mensagem falada antes de gravar o programa propriamente dito. Portanto, antes de executar a instrução SAVE, fale o nome do programa no microfone, gravando assim uma identificação positiva do programa.



Digite SAVE "TEMPERATURA", mas não pressione ENTER ainda.

Coloque seu gravador no modo de gravação (pressione RECORD e PLAY) e, feito isso, você deverá pressionar ENTER. Haverá um intervalo de aproximadamente cinco segundos, durante os quais nada aparecerá na tela.

A gravação inicia-se quando várias faixas aparecerem na tela.

Finalmente, após a gravação, a tela voltará ao normal com o código 0/0 em seu canto inferior esquerdo, indicando o fim da gravação.

Agora, você pode desligar o gravador.

Colocamos um resumo do processo de gravação de programas em fita:

1. Nome do Programa (REM);
2. Prepare uma fita virgem ou uma que possa ser regravada;
3. Grave o nome do programa através do microfone;
4. Conecte o gravador ao CP-200:
5. Digite:
   SAVE "nome do programa" (não pressione ENTER ainda);
6. Coloque o gravador para gravar;
7. Pressione ENTER;
8. Quando aparecer 0/0 na tela, pare o gravador.

Aconselhamos ouvir a fita para certificar-se da gravação. Desligue os cabos de conexão, volte a fita e coloque o gravador para tocar de maneira normal.

Você deve ouvir sua própria voz identificando o programa. Depois disso, você ouvirá um ruído referente ao movimento da fita antes de se pressionar ENTER. Em seguida, haverá cinco segundos de silêncio. Então, será ouvido um forte ruído correspondente às faixas na tela, seguido do ruído inicial. Aí termina a gravação.

Se após a gravação da sua voz não houver os ruídos descritos, provavelmente a gravação não surtiu efeito. Tente novamente, mas

antes verifique se as conexões estão corretas. Em alguns gravadores só é possível gravar com o cabo EAR desconectado, por isso retire esse cabo da tomada de seu gravador para fazer a tentativa. Outras causas podem ser sujeira na cabeça de gravação, fita ruim, etc.

Agora vamos copiar um programa da fita para a memória do CP-200.

Abaixo mostramos um resumo de como proceder para realizar essa operação.

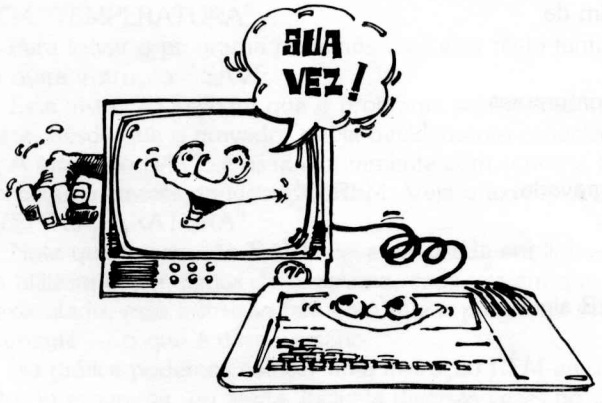
1. Posicione a fita na parte onde foi gravado o programa. Para isso baseie-se na gravação da sua voz anunciando o programa.
2. Verifique se o soquete EAR está convenientemente conectado ao soquete correspondente do gravador.
3. Ajuste o volume do gravador para três quartos do máximo.
4. Ajuste o controle de tonalidade, fazendo com que os agudos sejam reforçados.
5 Digite: LOAD "nome do programa" (Não pressione ENTER ainda)
6. Ligue o gravador.
7. Pressione ENTER

Se o programa for carregado (copiado) corretamente, você verá uma série de faixas na tela. Ao terminar o carregamento, aparecerá o código 0/0 indicando que a operação foi bem sucedida. Caso esse código não apareça após sumirem as faixas da tela é provável que você tenha digitado o nome errado, e o computador continuará lendo na fita até encontrar. Nesse caso você deve tentar novamente.

O maior problema para se carregar um programa gravado em fita é o ajuste correto do volume. O ajuste correto permite que o CP-200 conheça os sinais gravados sem distorção. O nível depende das características particulares do gravador usado e deve ser determinado por você mesmo.

Se você se esquecer do nome do programa ou quiser carregar o primeiro que estiver na fita, simplesmente digite:

LOAD "" sem colocar nada entre as aspas.



1. Repita o programa de cálculo do raio (capítulo três) adicionando:
   a) tabulação de resultados;
   b) um título;
   c) um *loop* que torne necessário apenas uma introdução da instrução RUN.
2. Rode o programa usando valores quaisquer.
3. Grave o programa em fita, utilizando o título como nome do programa. Não se esqueça de verificar se o programa foi gravado

realmente. O programa deve ficar como o listado abaixo:

```
 1 REM "RAIO"
 5 PRINT "CIRCUNFERENCIA","RAIO"
10 INPUT C
20 PRINT C, C/(2*π)
30 GOTO 10
```

4. Escreva um programa que imprima todos os números que couberem na tela. Lembre-se de que o programa pára automaticamente ao preencher a tela.

Soluções Propostas

```
 5 LET N=2
10 PRINT N,
20 LET N=N+2
30 GOTO 10
```

Ao rodar o programa acima haverá uma certa demora para aparecer algo na tela. Isso se deve ao fato de o CP-200 estar executando um *loop* que não contém nenhuma instrução de parada (INPUT, por exemplo). Ele só vai parar quando não houver mais lugar na tela onde imprimir. Nesse ponto ocorrerá um erro de código 5 (veja apêndice com os Códigos de Erro) na linha 10.

Um detalhe importante desse programa é a vírgula final na linha 10. Essa vírgula faz com que o próximo valor a ser impresso seja tabulado automaticamente. Dessa forma, a próxima instrução PRINT será executada a partir da próxima meia tela (veja "Como a vírgula pode tabular?" no Capítulo IV).

5. Adicione no programa da questão anterior a seguinte linha e tente entender seu significado rodando o programa.

```
25 PAUSE 60
```

No próximo capítulo você encontrará uma explicação detalhada dessa instrução.

6. Baseando-se no exercício 6 do capítulo anterior, tente elaborar um programa que:

— forneça o montante gasto com o combustível;
— calcule a economia de combustível em Km/l, após vários abastecimentos.

Solução proposta

```
 5 LET P=0
 6 LET L=0
10 PRINT "DIGITE A QUILOMETRAGEM"
20 INPUT KM
30 PRINT "QUANTOS LITROS?"
40 INPUT LT
50 PRINT "CUSTO POR LITRO?"
60 INPUT CUSTO
70 PRINT "QUAL COMBUSTÍVEL?"
80 INPUT C$
85 CLS
90 PRINT "FORAM PERCORRIDOS";KM;"KM"
100 PRINT "FORAM GASTOS ";LT;"LITROS DE"; C$
110 PRINT "CONSUMO= "; KM/LT;" KM/L"
120 PRINT "CUSTO=CR$";(LT*CUSTO)/KM;"POR KM"
130 LET P=P+KM
140 LET L=L+LT
```

```
150 PRINT "TOTAL GASTO = CR$";L*CUSTO
160 PRINT "CONSUMO MEDIO = ";P/L;"KM/L"
165 PRINT
170 GOTO 10
```

Note que nesta solução adotamos uma instrução CLS antes de imprimir o relatório de consumo. Isso fará com que a tela esteja limpa no momento de imprimir os resultados.

Outra particularidade dessa solução é a utilização da instrução PRINT sem item algum para ser impresso (linha 165). Essa linha força o programa a deixar uma linha da tela em branco (salta uma linha).

As instruções 130 e 140 são acumulativas: o conteúdo inicial das variáveis é zero e, a cada *loop*, são incrementados pelos valores parciais.

5. Descubra qual a função do programa abaixo:

```
10 POKE 30000,211
20 POKE 30001,240
30 POKE 30002,201
40 LET USR 30000
50 GOTO 40
```

Não se preocupe com o significado dessas instruções, mas apenas com o efeito conseguido. Você poderá usar esse efeito em outros programas, como será mostrado nos exercícios do próximo capítulo.

Desvios Condicionais

Uma característica peculiar do computador é a capacidade que ele tem de tomar decisões.

Em termos práticos, o computador, em um dado ponto do programa, pode decidir entre realizar um salto (um desvio na programação) ou prosseguir para a linha seguinte.

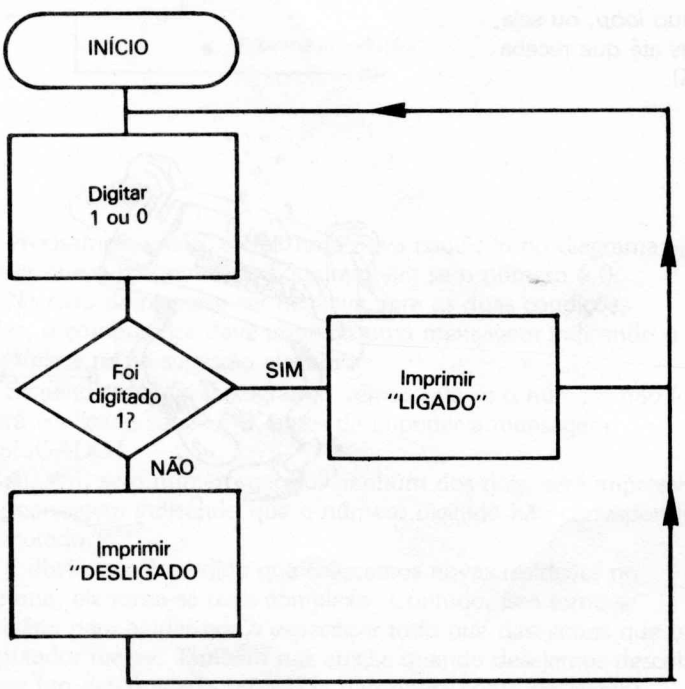
Em programação algumas decisões adotadas pelo computador são pré-estabelecidas. Vejamos um exemplo:

Suponha que você queira que o CP-200 funcione como um interruptor, de tal forma que quando você digitar 1, ele imprima "LIGADO", e quando você digitar 0, ele imprima "DESLIGADO".

Nesse caso, podemos fazer um diagrama das etapas que devem ser executadas para se chegar ao resultado que queremos.

Um diagrama representa graficamente aquilo que deve ser seguido pelo programa. Isto torna muito mais fácil ao desenvolvimento das várias etapas a serem processadas.

Veja o nosso exemplo disposto num diagrama de blocos:



No segundo bloco nos é solicitada a introdução de um número (1 ou 0).

No bloco seguinte, o computador deve tomar uma decisão: se foi digitado 1, deverá imprimir "LIGADO" e, se foi digitado 0, deverá imprimir "DESLIGADO".

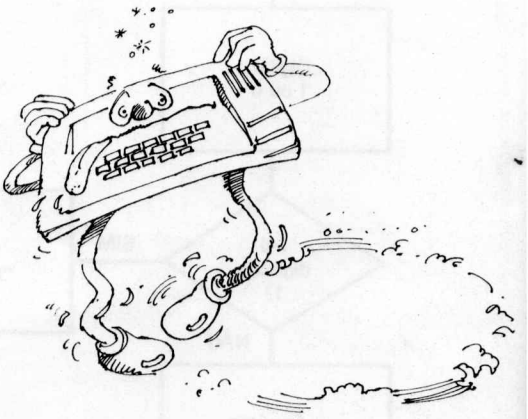
Após a impressão de qualquer uma das mensagens, o programa deve retornar para receber um novo número.

Como você pode perceber, o diagrama de blocos é apenas uma maneira organizada de se traçar um plano para um programa.

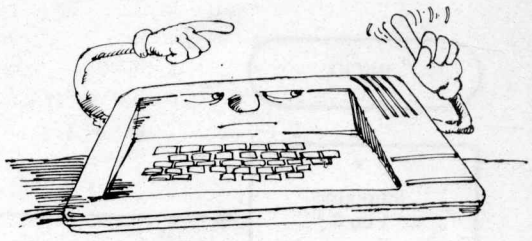
Convencionou-se usar um retângulo com os lados menores arredondados para indicar o começo ou fim de um programa. Um retângulo comum representa trechos onde ocorre uma ação, ao passo que um losango denota os pontos de tomadas de decisões. As linhas indicam o caminho a ser percorrido, estabelecendo a seqüência lógica do programa.

Comumente, essa seqüência, tecnicamente denominada de **fluxo lógico,** inicia-se de cima para baixo ou da esquerda para a direita, salvo os casos onde a orientação é estipulada mediante a indicação de uma seta sobre a linha.

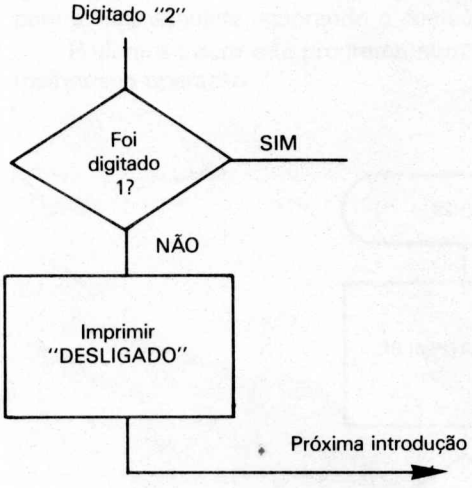
Um diagrama de blocos (fluxograma) é bastante útil para verificarmos se a lógica do programa está correta ou não. Seu emprego faz com que o programa fique num contínuo *loop*, ou seja, o computador executará continuamente as instruções até que receba um comando externo para parar (STOP ou BREAK).



Podemos notar também que a introdução de qualquer outro número diferente de 1 será interpretada como o número 0 para o computador. E não é exatamente isso o que queremos.



Segundo a lógica adotada, o computador verificará se o número digitado é 1. Se não for, o computador simplesmente imprimirá "DESLIGADO" (veja a seqüência no fluxograma).



Precisamos, então, incluir uma nova condição no diagrama: permitir que o programa verifique também se o número é 0.

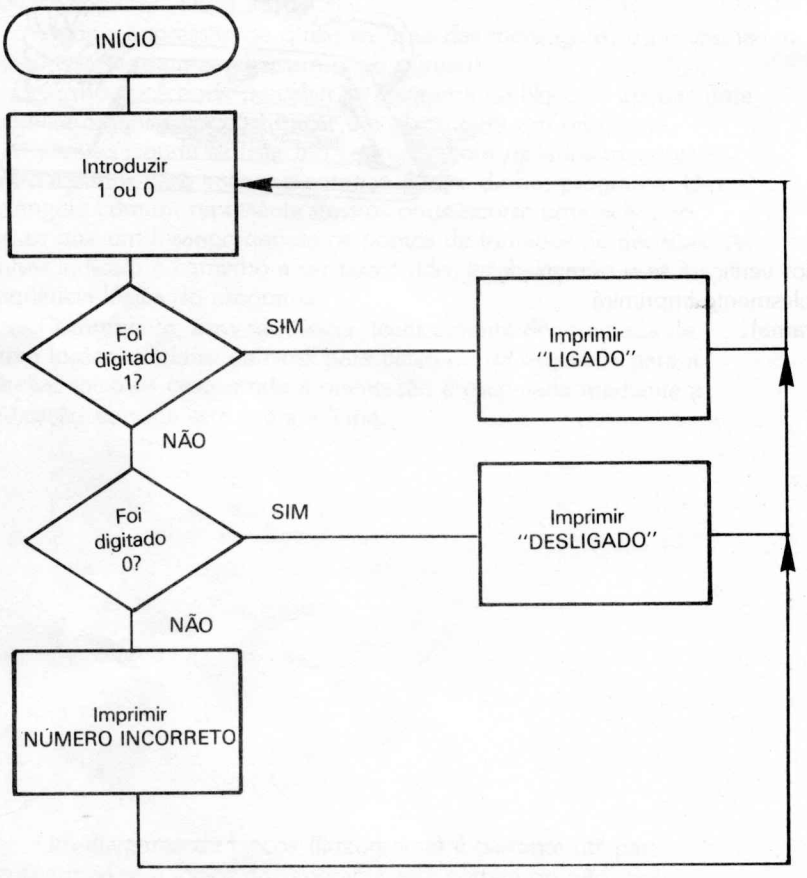
No caso da resposta ser negativa para as duas condições citadas, o computador deve imprimir uma mensagem indicando o erro. Veja a nossa sugestão abaixo:

Seguindo o novo fluxograma, vemos que se o número não for 1, será verificado se ele é 0, antes de imprimir a mensagem "DESLIGADO".

Aí, sim, se o número não for nenhum dos dois, será impressa uma mensagem indicando que o número digitado não corresponde ao esperado.

É óbvio que à medida que colocamos novas restrições no programa, ele torna-se mais complexo. Contudo, isso torna-se necessário para ajudar-nos a especificar tudo que desejamos que o computador realize. Também nos auxilia quando desejamos descobrir porque um determinado programa não opera como desejamos.

Precisamos agora transformar nosso fluxograma num programa em BASIC.



A instrução de entrada nós já a conhecemos: INPUT.

Vamos agora colocar o número digitado em uma variável chamada CHAVE, por exemplo. A primeira linha será então:

10 INPUT CHAVE

A próxima instrução deve verificar se a variável CHAVE é igual a 1. Em português, poderíamos dizer: SE a condição é verdadeira (CHAVE = 1) ENTÃO imprima LIGADO.

Como a linguagem BASIC é derivada do inglês, substituímos as palavras SE e ENTÃO por suas equivalentes: IF e THEN. Portanto, a instrução obedecerá a essa construção: IF condição verdadeira, THEN ação a ser realizada. Como a condição que procuramos é CHAVE = 1, a instrução ficará:

20 IF CHAVE = 1 THEN GOTO 100

O desvio para a linha 100 permite que separemos a instrução de impressão daquela de retorno ao início, como abaixo:

100 PRINT "LIGADO"
110 GOTO 10

Continuando com o programa, o próximo passo é colocar a instrução que verifica se o número é 0. Podemos fazer isso da seguinte forma:

30 IF CHAVE = 0 THEN GOTO 200

Dessa vez, a impressão da mensagem é colocada na linha 200

como segue:

```
200 PRINT "DESLIGADO"
210 GOTO 10
```

Finalmente, seguindo o fluxograma, devemos introduzir uma instrução para impressão da mensagem "NÚMERO INCORRETO", juntamente com um novo desvio para a linha 10.

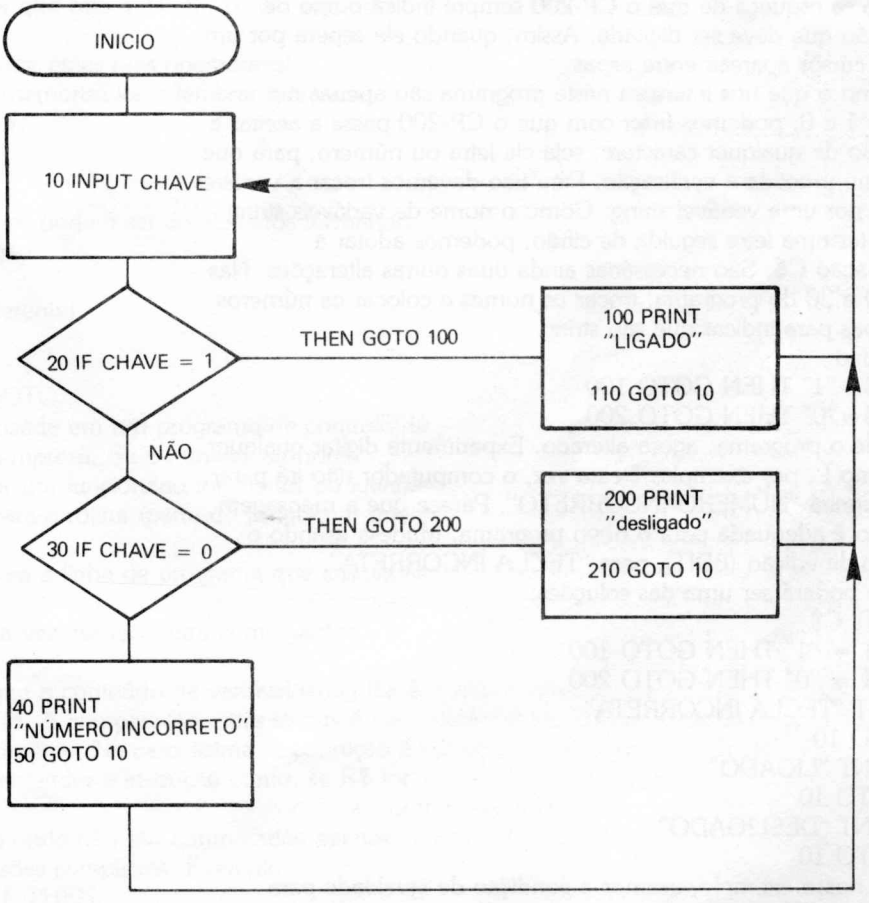
O programa ficará assim:

```
10 INPUT CHAVE
20 IF CHAVE = 1 THEN GOTO 100
30 IF CHAVE = O THEN GOTO 200
40 PRINT "NÚMERO INCORRETO"
50 GOTO 10
100 PRINT "LIGADO"
110 GOTO 10
200 PRINT "DESLIGADO"
210 GOTO 10
```

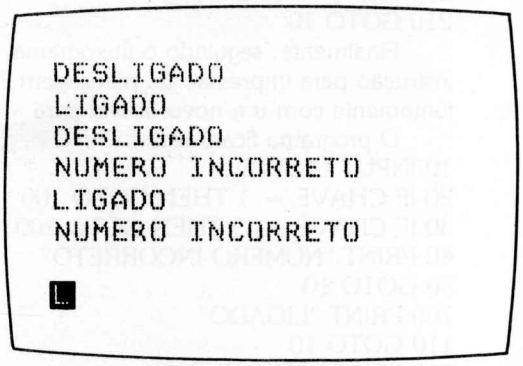
Não se preocupe com a numeração das linhas. Lembre-se de que os números apenas definem a ordem de execução.

Analisando a listagem do programa, você deve perceber que quando a condição é verdadeira, o computador realiza a instrução que segue o THEN. Caso a condição seja falsa, o computador passa para a linha seguinte, ignorando o conteúdo que segue a condição.

Podemos inserir este programa num fluxograma identificando melhor sua operação:



Rode o programa agora. Você deverá obter um resultado semelhante a este:



Se, por engano, digitarmos uma letra ao invés de um número, o computador interromperá a execução do programa e apresentará um código de erro do tipo.

2/10

O número 2 indica o tipo de erro (veja o apêndice B, que aborda os códigos de erro). Neste caso específico, significa que o computador aguardava por um número, mas foi digitada uma letra.

O número após a barra (/) indica onde ocorreu o erro (linha 10). Não se esqueça de que o CP-200 sempre indica o tipo de informação que deve ser digitada. Assim, quando ele espera por um string, o cursor aparece entre aspas.

Como o que nos interessa neste programa são apenas os números 1 e 0, podemos fazer com que o CP-200 passe a aceitar a introdução de qualquer caractere, seja ele letra ou número, para que ele mesmo proceda a verificação. Para isso devemos trocar a variável CHAVE por uma variável string. Como o nome de variáveis string só pode ter uma letra seguida de cifrão, podemos adotar a denominação C$. São necessárias ainda duas outras alterações. Nas linhas 20 e 30 do programa: trocar os nomes e colocar os números entre aspas para indicar que são string.

Veja abaixo:

```
20 IF C$ = "1" THEN GOTO 100
30 IF C$ = "0" THEN GOTO 200
```

Rode o programa, agora alterado. Experimente digitar qualquer letra, como L, por exemplo. Dessa vez, o computador não irá parar, mas imprimirá "NÚMERO INCORRETO". Parece que a mensagem ainda não é adequada para o novo programa. Mude-a usando o comando de edição (EDIT), para "TECLA INCORRETA".

Esta poderá ser uma das soluções:

```
10 INPUT C$
20 IF C$ = "1" THEN GOTO 100
30 IF C$ = "0" THEN GOTO 200
40 PRINT "TECLA INCORRETA"
50 GOTO 10
100 PRINT "LIGADO"
110 GOTO 10
200 PRINT "DESLIGADO"
210 GOTO 10
```

Em nosso exemplo, usamos a condição de igualdade para tomar a decisão. Porém você poderá estipular qualquer condição ao

computador. Ele sempre verificará se é verdadeira ou falsa, antes de prosseguir.

Operadores Relativos e Variáveis String

A condição envolve sempre a comparação de variáveis numéricas ou string.
Essa comparação é definida por um sinal que chamamos de *operador relativo.*
O sinal de igualdade é um exemplo de operador relativo, assim como outros sinais que você encontrar no teclado do CP-200 (>, <, <=, >=, <>).
Esses operadores exprimem as várias comparações que podem ser realizadas pelo CP-200. Essas comparações são:

Maior que >	IF A > B THEN...
Menor que <	IF A < B THEN...
Igual a =	IF A = B THEN...

No primeiro caso, podemos ler a instalação de exemplo como "Se A é melhor que B então..." Uma instrução desse tipo executa a ação especificada após o THEN caso a variável à esquerda do operador, A, seja maior que a que está à direita, B. No segundo caso, a condição é invertida.

Cuidado para não confundir esses dois operadores!

Uma meneira prática de memorizá-los é lembrar que eles sempre apontam para o menor:

MAIOR > MENOR
MENOR < **MAIOR**

Os três operadores relativos podem ser combinados formando novas condições de comparação:
Maior ou igual a > =
Maior ou menor que < > (diferente)
Menor ou igual a < =
Eis um exemplo prático:

```
IF HORAS <=240 THEN GOTO...
```

Essa instrução pode ser usada em um programa de controle da folha de pagamento de uma empresa. Se o número de horas trabalhadas durante o mês por um funcionário for menor ou igual a 240, o computador passará para a rotina (parte do programa) que calcula o salário normal.

Caso contrário, saltará para a linha de programa que calcula as horas extras.

Um outro exemplo, desta vez usando outra combinação:

```
IF R$ = "SIM" THEN...
```

Nesse exemplo vemos que o conteúdo da variável string R$ é comparado com a palavra SIM. A comparação entre strings é análoga a que se faz entre números. No caso acima, a condição é **igual a.** Portanto, podemos entender a instrução como: se R$ for igual a "SIM" então...

Podemos ainda ter casos onde não são comparados apenas variáveis mas também expressões complexas. Exemplo:

```
IF (A*B)/(C*D)<=C*E+F THEN...
```

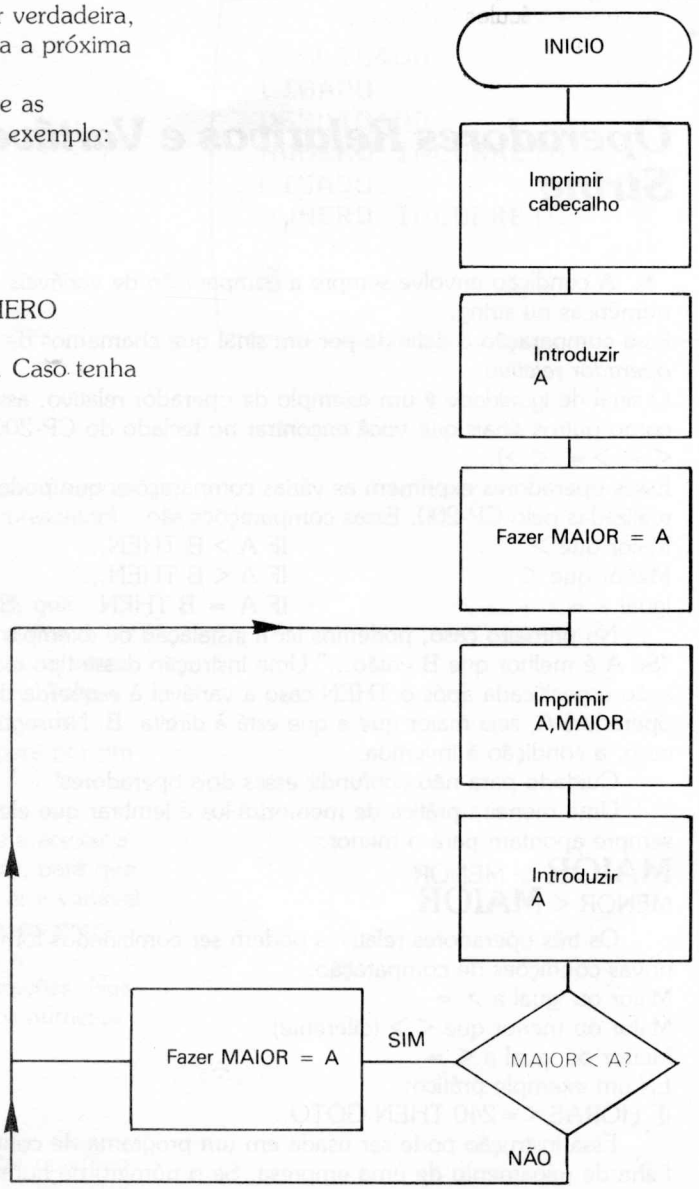
Para o caso de expressões, a comparação é efetuada entre os

resultados das mesmas. Em outras palavras, o computador calcula as expressões e verifica se a comparação entre os resultados é verdadeira. A partir daí, o procedimento é normal: se for verdadeira, a ação após THEN é realizada, caso contrário, passa para a próxima linha.

Para uma melhor compreensão do que foi dito sobre as comparações, damos aqui um programa completo como exemplo:

```
10 PRINT "NUMERO", "MAIOR"
20 INPUT NUMERO
30 LET MAIOR = NUMERO
40 PRINT NUMERO, MAIOR
50 INPUT NUMERO
60 IF MAIOR < NUMERO THEN LET MAIOR = NUMERO
70 GOTO 40
```

Tente analisar o programa e descobrir o que ele faz. Caso tenha dificuldades com a listagem, utilize o fluxograma abaixo.



O programa aceita dados numéricos e os imprime juntamente com o maior número digitado até o momento. A linha principal deste programa é, portanto a de número 60, porque é ela que permite a atualização do maior valor digitado, caso o último valor seja superior.

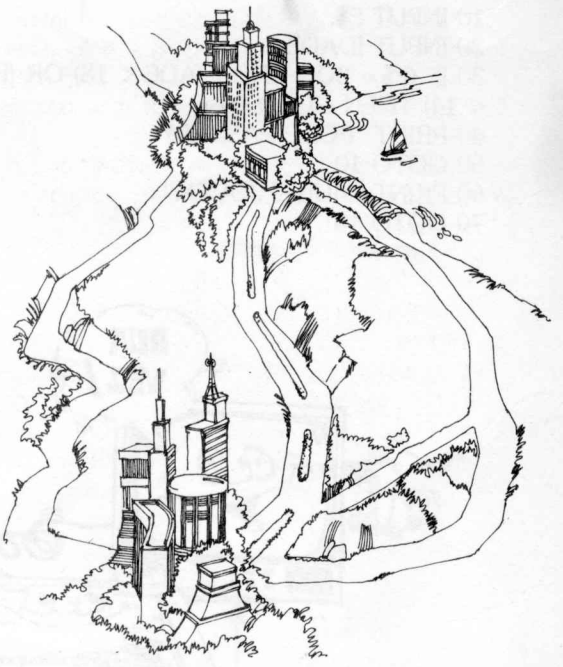
Podemos usar uma alternativa para essa linha que evite a repetição da instrução LET MAIOR = NUMERO (linha 30 e linha 60). Podemos substituir essa instrução na linha 60 por uma instrução GOTO 30. A linha ficará assim:

```
60 IF MAIOR < NUMERO THEN GOTO 30
```

Essa alteração se refere a um ponto muito importante em programação.

Podemos atingir um objetivo de várias maneiras diferentes, mas uma dentre elas será a mais recomendada. É como ir de uma cidade

para outra. Geralmente existem vários percursos possíveis; uns diretos, outros passando por outras cidades e ainda aqueles que contêm obstáculos indesejáveis.



Por isso, tenha cuidado ao elaborar um programa para não tomar o pior caminho.

Lembre-se sempre: Antes de melhorar um programa... **faça-o funcionar!**

Como já foi dito, os operadores relativos podem comparar valores string. A condição A$ = "TALVEZ" é verdadeira se o conteúdo da variável A$ for "TALVEZ".
Um string é menor que outro quando precedê-lo em ordem alfabética, assim "A"<"B", e "VOCÊ" > "EU" e, veja só, "MILHÃO" > "BILHÃO".

Resumindo, o que importa na comparação de string é a ordem alfabética. Para o CP-200, um número precede, em ordem alfabética, uma letra. Então "1" < "A". Um string pode ter números e letras, indistintamente.
Exemplos:
"AVENIDA PAULISTA, 1200" < "AVENIDA PAULISTA, 870"
"RUA A" > "RUA 2"
"ANGRA 1" < "ANGRA 2"
"SERIE AA" < "SERIE AB"

Operadores Lógicos

O CP-200 possui ainda alguns operadores que podem ampliar o uso da instrução IF... THEN. São os operadores AND, OR e NOT. São vocábulos da língua inglesa que significam, respectivamente, E, OU e NÃO.

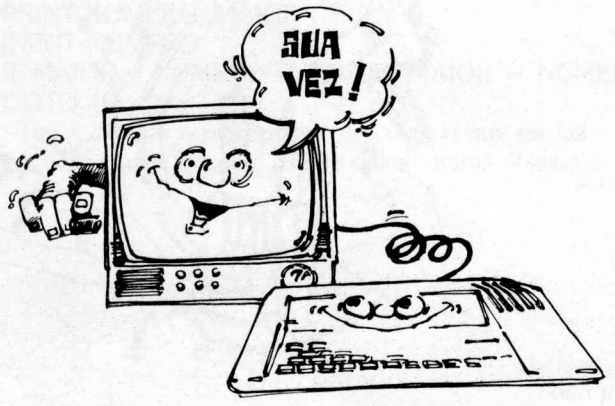
Para ilustrar o uso dos operadores lógicos, tomaremos como exemplo um programa que verifica se uma pessoa pode assistir a um filme a partir da idade e do tipo do filme.

Os filmes do tipo XX só podem ser assistidos por maiores de 18 anos.

Os do tipo AA, por maiores de 14 anos.

Você poderá sofisticar ainda mais, incluindo novas categorias e, ainda, prever o caso de crianças acompanhadas por adultos.

```
10 INPUT F$
20 INPUT IDADE
30 IF (F$ = "XX" AND IDADE < 18) OR (F$ = "AA" AND IDADE
< 14) THEN GOTO 60
40 PRINT "PODE PASSAR"
50 GOTO 10
60 PRINT "MUITO JOVEM"
70 GOTO 10
```



1. Escreva um programa que permita a introdução de vários números, imprimindo, a cada introdução, o maior e o menor número introduzidos. Comece pelo fluxograma.

2. Modifique o programa da questão anterior para obter a média aritmética dos números introduzidos.

3. Escreva um programa que permita a introdução de quatro palavras e as imprima em ordem alfabética.

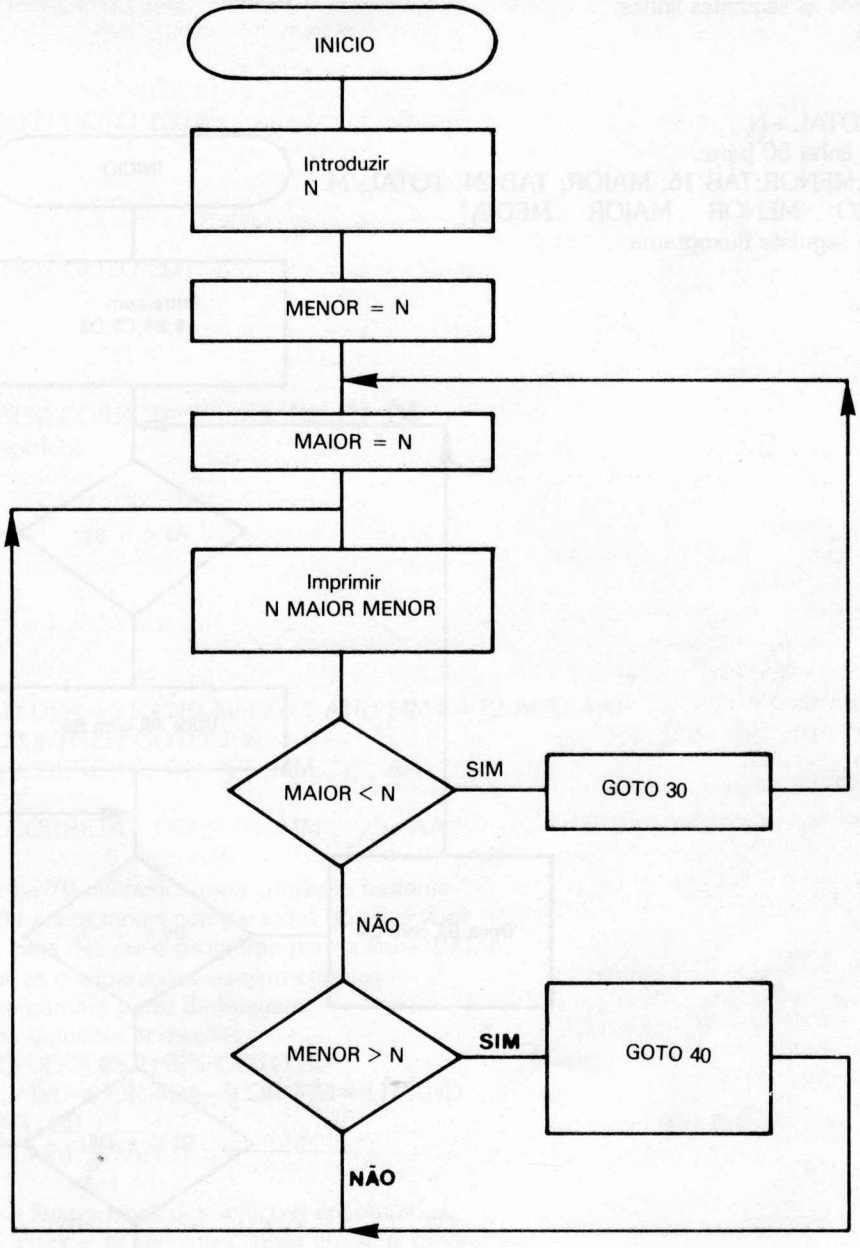
4. Escreva um programa que reconheça datas corretas na forma DD/MM/AA, onde: DD é um número entre 1 e 31, inclusive; MM é um número entre 1 e 12 e AA é um número entre 1900 e 2000.

O programa deve imprimir DATA INCORRETA ao constatar uma irregularidade.

5. Inclua no programa da questão anterior, instruções que permitam ao computador reconhecer datas incorretas inclusive para FEVEREIRO, ABRIL, JUNHO, SETEMBRO, NOVEMBRO.

Soluções Propostas

1. fluxograma



Programa

```
 1 REM CAP 7 EX 1
10 PRINT "NUMERO    MENOR    MAIOR"
20 INPUT N
30 LET MENOR = N
```

```
40 LET MAIOR = N
50 PRINT N;TAB 10; MENOR; TAB 20; MAIOR
60 INPUT N
70 IF MAIOR < N THEN GOTO 30
80 IF MENOR > N THEN LET MENOR = N
90 GOTO 50
```

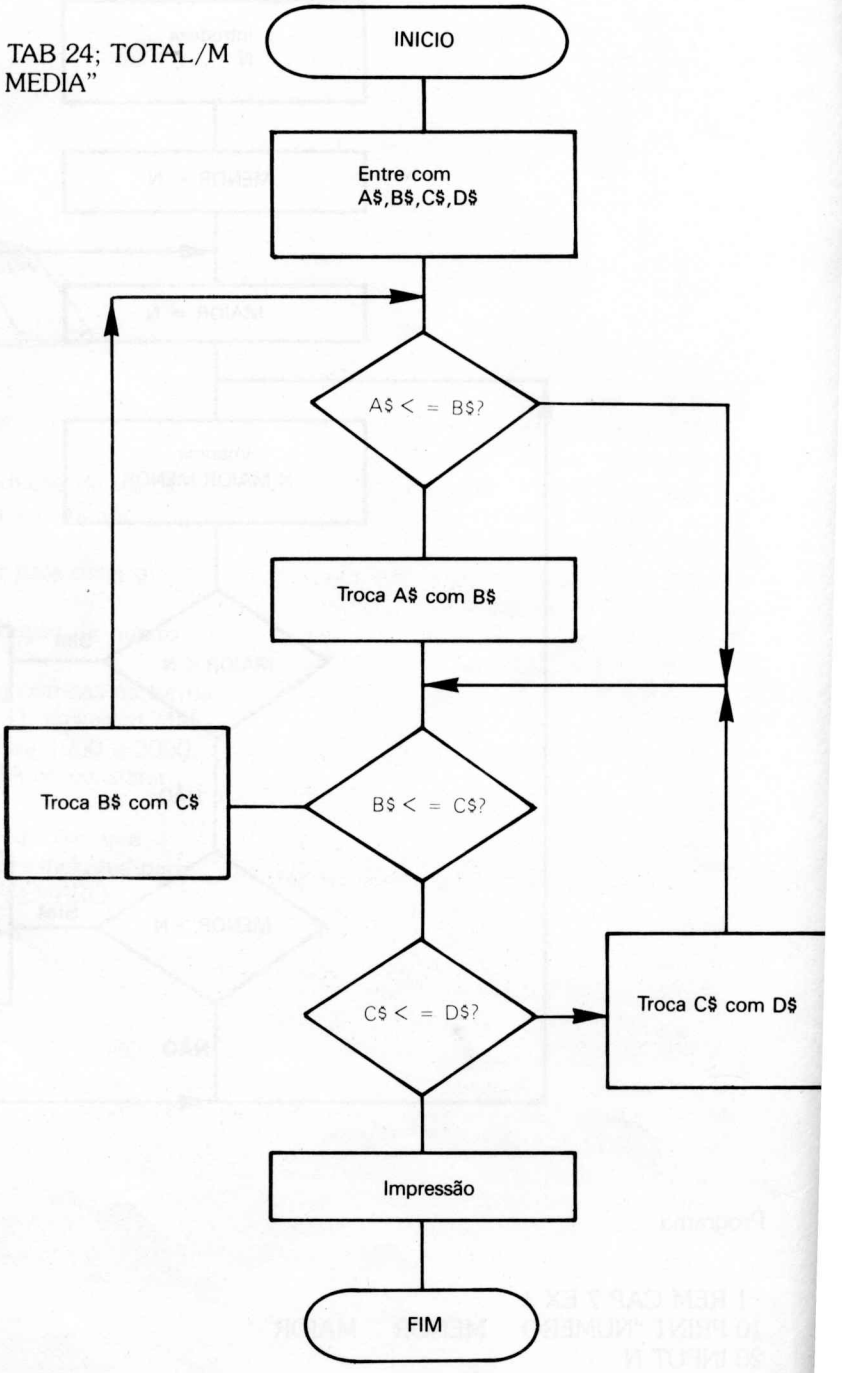
2. Acrescentamos as seguintes linhas:

```
25 LET TOTAL = N
45 LET M = 1
55 LET M = M + 1
65 LET TOTAL = TOTAL + N
```

E alterarmos a linha 50 para:

```
50 PRINT N;TAB 8;MENOR;TAB 16; MAIOR; TAB 24; TOTAL/M
10 PRINT "NUMERO    MENOR    MAIOR    MEDIA"
```

3. Sugerimos o seguinte fluxograma:



Programa:

```
10 REM ORDENAÇÃO
20 INPUT A$
30 INPUT B$
40 INPUT C$
50 INPUT D$
60 IF A$ <=B$ THEN GOTO 100
70 LET E$=A$
80 LET A$=B$
90 LET B$=E$
100 IF B$<=C$ THEN GOTO 150
110 LET E$=B$
120 LET B$=C$
130 LET C$=E$
140 GOTO 60
150 IF C$<=D$ THEN GOTO 200
160 LET E$=C$
170 LET C$=D$
180 LET D$=E$
190 GOTO 100
200 PRINT "A ORDEM CORRETA E:",,A$,,B$,,C$,,D$
```

4. Programa sugerido:

```
5 REM DATA
8 CLS
10 PRINT "DIA?"
20 INPUT DD
30 PRINT "MÊS ?"
40 INPUT MM
50 PRINT "ANO"
60 INPUT AA
70 IF (DD>=1 AND DD<=31 AND MM>=1 AND MM<=12 AND AA>
1900 AND AA< 2000) THEN GOTO 100
80 PRINT "DATA INCORRETA"; DD; "/"; MM; "/"; AA
90 STOP
100 PRINT "DATA CORRETA"; DD ; "/" ; MM ; "/" ; AA
110 GOTO 10
```

Note que na linha 70 utilizamos uma condição bastante complexa, O CP-200 aceita tantas comparações quantas você introduzir. No caso, para desviar o programa para a linha 100, é necessário que todas as comparações estejam corretas.
Tente montar o fluxograma a partir da listagem.

5. Sugerimos as seguintes alterações:

```
100 IF MM=2 AND DD > 28 THEN GOTO 80
110 IF (MM=4 OR MM=6 OR MM=9 OR MM=11) AND
DD=31 THEN GOTO 80
120 PRINT DD; "/"; MM;"/";AA
130 GOTO 10
```

Tente montar os fluxogramas das soluções encontradas.
Experimente, ainda, colocar as seguintes linhas em seus programas:

```
1 POKE 30000, 211
2 POKE 30001, 240
3 POKE 30002, 201
```

Feito isso, acrescente a instrução LETX=USR 30000 antes de cada linha com INPUT.

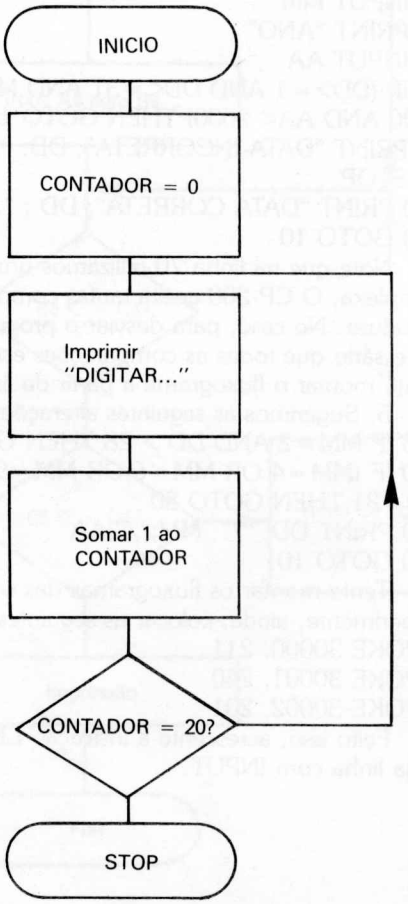
Controlando Repetições

Em programação, a parte mais trabalhosa é, geralmente, a digitação das linhas. Mesmo quando tomamos todo o cuidado para não cometer erros na introdução de um programa, o serviço ainda é cansativo.

Assim, ao elaborar um programa, devemos partir do princípio de que um texto compacto é mais conveniente, tanto para digitar quando para detectar eventuais falhas.

Imagine, por exemplo, que quiséssemos imprimir vinte vezes a mensagem "DIGITAR PODE TORNAR-SE CANSATIVO". Podemos realizar isso de uma maneira repetitiva, digitando vinte instruções PRINT. Porém, a linguagem BASIC possui recursos que tornam a elaboração de um programa repetitivo numa tarefa muito interessante e criativa.

Com os recursos conhecidos até agora, já é possível realizar essa tarefa de uma maneira mais racional. Veja o fluxograma abaixo:



A partir desse fluxograma podemos escrever o seguinte programa:

```
10 LET CONTADOR = 0
20 PRINT "DIGITAR PODE TORNAR-SE CANSATIVO"
30 LET CONTADOR = CONTADOR + 1
40 IF CONTADOR < > 20 THEN GOTO 20
50 PRINT "TAREFA REALIZADA"
```

Rode o programa e verifique se a mensagem foi impressa vinte vezes.

Não acha que o programa ficou bastante compacto em relação à sua proposição?

O CP-200 incorpora duas instruções em BASIC que permitem que a contagem dos loops seja feita de uma maneira mais objetiva. São as instruções FOR e NEXT, usadas normalmente da seguinte maneira:

```
10 FOR C = 1 TO 20
20 PRINT "DIGITAR PODE TORNAR-SE CANSATIVO"
30 NEXT C
```

Podemos entender o significado dessas instruções estabelecendo sua correspondência para o português. Assim a primeira linha pode ser lida como **"PARA C variando de 1 ATÉ 20"** — o que quer dizer que a variável C será inicialmente igualada a 1 para ser incrementada de uma unidade a cada loop, até atingir 20.

A segunda linha apenas imprime a mensagem, como no programa anterior.

A terceira linha deve ser lida como "PRÓXIMO valor de C". Quando executada, essa linha incrementa o valor da variável de controle (C) e desvia a execução para a linha que contém a instrução FOR.

As instruções FOR e NEXT devem sempre ser usadas em conjunto, pois são cmplementares. É importante notar que a palavra TO usada juntamente com FOR é um comando que deve ser digitado usando-se SHIFT e tecla 4, e não as teclas T e O.

Os loops controlados, gerados pela instrução FOR, são comumente denominados loops FOR/NEXT. A variável de controle (em nosso caso, C) é sempre uma variável numérica e não pode ter mais que uma letra no nome.

Deve-se tomar o cuidado de colocar toda a rotina (conjunto de linhas) que se quer repetir entre as linhas que contêm o FOR e o NEXT.

Podemos ver o que acontece com um loop fazendo imprimir o valor de C ao invés de uma mensagem.

Veja o programa abaixo:

```
10 FOR C=1 TO 20
20 PRINT C
30 NEXT C
```

Na tela será impresso:

```
 1
 2
 3
 4
 5
 6
 7
 8
 9
 10
 11
 12
 13
 14
 15
 16
 17
 18
 19
 20
```

Podemos fazer com que a contagem seja realizada com um incremento diferente de 1. Podemos contar, por exemplo, de quatro em quatro. Para utilizar um incremento diferente da unidade, recorremos ao comando STEP (SHIFT e E). Esse comando especifica o valor do incremento usado. Veja o exemplo:

```
10 FOR C=1 TO 20 STEP 4
20 PRINT C
30 NEXT C
```

Neste programa, a variável de controle parte de 1 e é incrementada de 4 a cada loop, até atingir 20. O resultado na tela será:

```
1
5
9
13
17
```

Esse programa imprime apenas cinco vezes, pois o incremento é maior que no caso anterior. Note que a variável de controle não foi igualada ao valor limite, pois 17+4=21. Porém, o loop foi interrompido porque o valor de C ultrapassou o estabelecido na instrução FOR.

Esse tipo de incremento é normalmente utilizado quando se deseja realizar cálculos a intervalos pré-estabelecidos dentro de uma determinada faixa.

Tomemos como exemplo uma tabela de conversão em polegadas para as medidas entre 5 e 10 centímetros, a intervalos de 5 cm. O programa abaixo monta essa tabela:

```
 5 PRINT "CENTIMETROS   POLEGADAS"
10 FOR  C=5 TO 100 STEP 5
20 PRINT C/2.54
30 NEXT C
```

Ao rodar esse programa, você obterá:

```
CENTIMETROS   POLEGADAS
5             1.968503
10            3.937007
15            5.905511
100           39.37007


0/30
```

Loops Entrelaçados

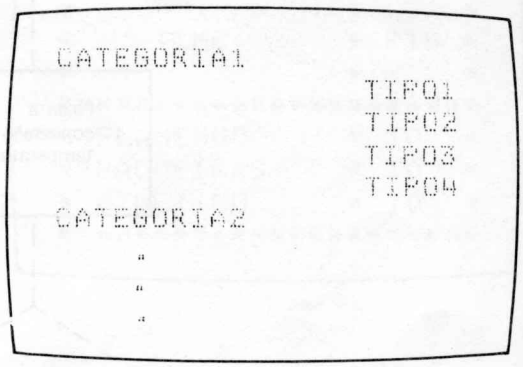
A rotina que compõe o loop FOR/NEXT pode conter um segundo loop, interno ao primeiro. Veja o exemplo:

```
10 FOR C=1 TO 3
20 PRINT "CATEGORIA" ;C
30         FOR T=1 TO 4
40         PRINT, "TIPO "; T
50         NEXT T
60 NEXT C
```

Neste caso, para cada repetição do loop externo será executado

o loop interno por completo, ou seja, será repetido até se atingir o valor limite.

Rodando o exemplo você obterá na tela:



Esse tipo de disposição é chamado de entrelaçamento de loops. É fundamental que dois loops estejam totalmente separados ou, então, no caso de loops entrelaçados, um totalmente contido no outro. Abaixo mostramos uma esquematização de loops entrelaçados:

```
FOR A = ...
  FOR B = ...
  NEXT B
NEXT A
```

SIM

A outra forma permitida para os loops é a sua total disjunção.

```
FOR A = ...
NEXT A
FOR B = ...
NEXT B
```

SIM

Nunca permita que os loops em seus programas fiquem sobrepostos, como islustrado abaixo:

```
FOR A = ...
  FOR B = ...
NEXT A
NEXT B
```

NÃO

Outro erro que deve ser evitado é um desvio da programação de fora para dentro de um loop. Haverá um erro se o computador encontrar uma instrução NEXT sem ter passado antes por uma FOR.

```
10 GOTO 50
40 FOR A = ...
50 PRINT...
60 NEXT A
```

NÃO

Apenas como exercício, tente rodar alguns programas com os erros mencionados nesse capítulo.

O loop FOR/NEXT é ideal quando queremos executar uma parte do programa várias vezes consecutivas. Porém, como fazer quando quisermos que a mesma parte do programa seja repetida em pontos diferentes durante a execução?

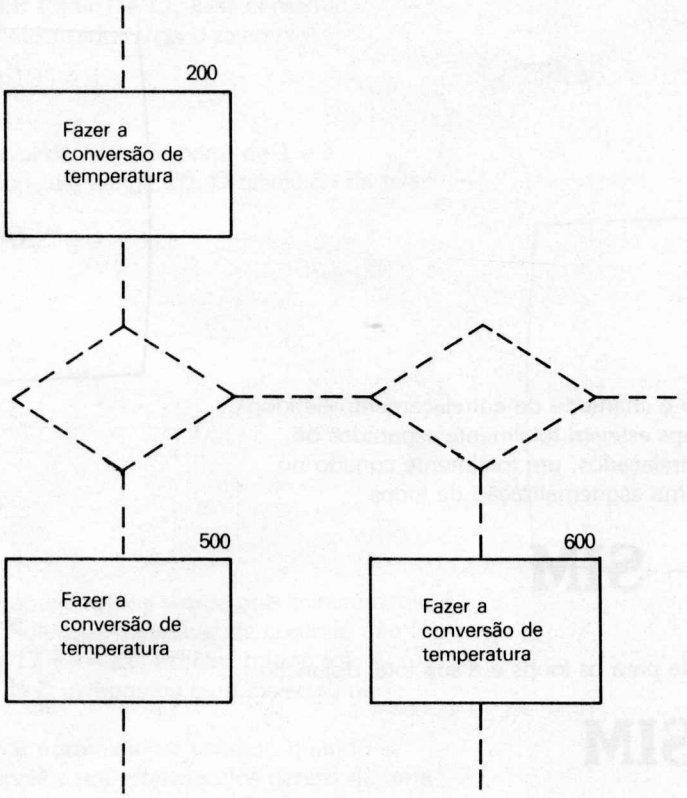
Sub-Rotinas

A um conjunto de instruções que executam uma tarefa específica dentro deste programa chamamos normalmente de rotina. Quando esse conjunto é requisitado em vários pontos do programa ele recebe o nome de sub-rotina.

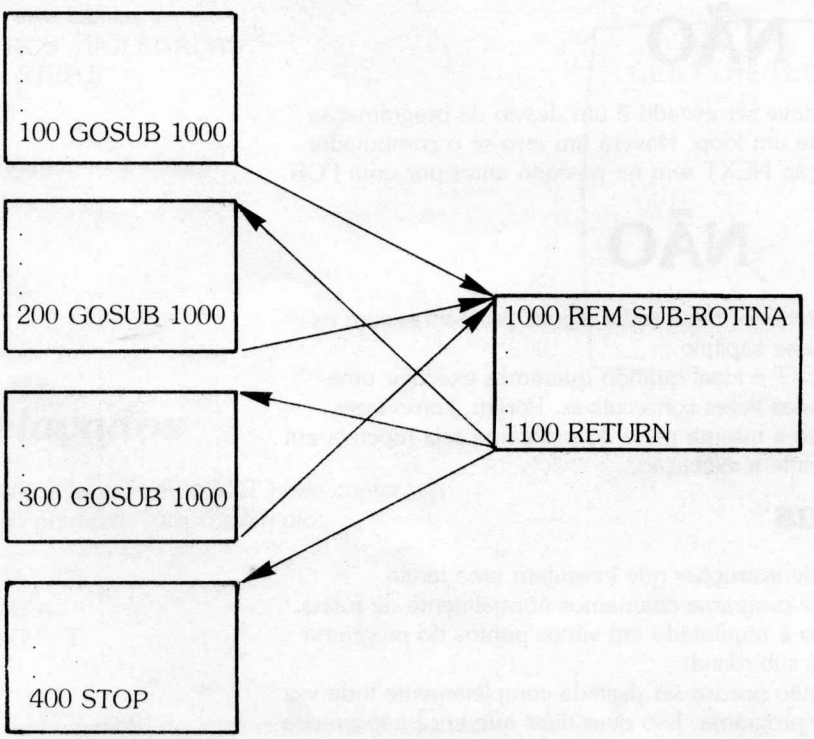
Uma sub-rotina não precisa ser digitada completamente toda vez que for necessária ao programa. Isso quer dizer que você não precisa

digitar as instruções referentes a um determinado cálculo, toda vez que ele se fizer necessário.

Eis um exemplo de um cálculo que se repete várias vezes:



Será possível escrever essa rotina apenas uma vez e apenas referir-se a ela quando necessário. Veja o exemplo abaixo:



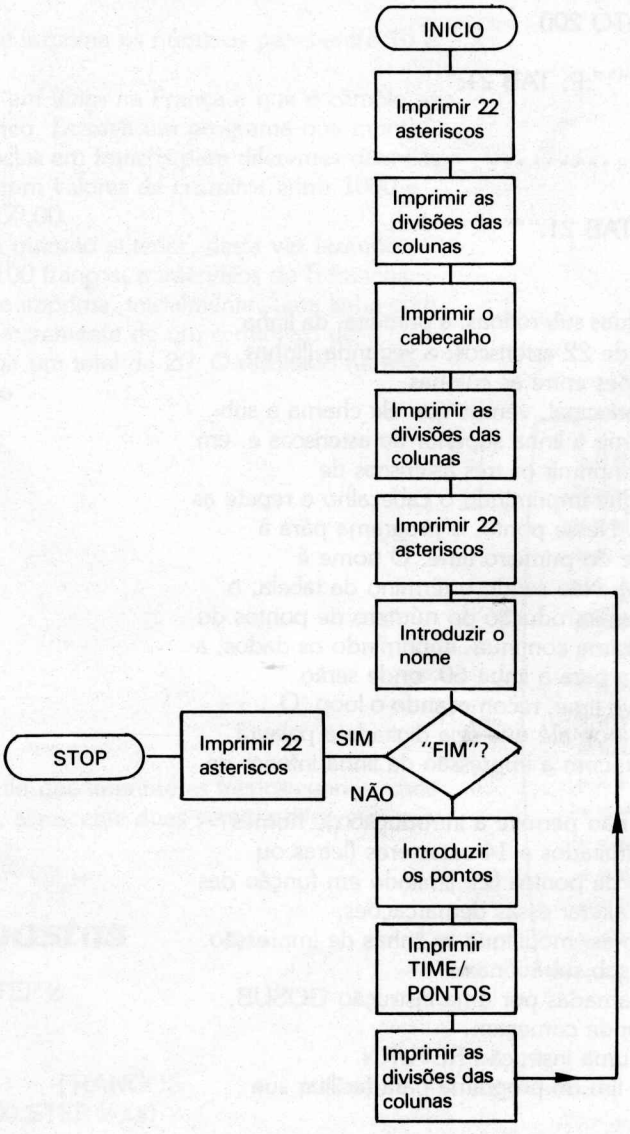
Para desviar para uma sub-rotina usamos a instrução GOSUB (GO + SUB) seguido do número de linha onde começa a sub-rotina desejada. Esta deve ser um pequeno programa, totalmente auto-suficiente, e deve terminar com a instrução RETURN. Quando o computador encontra a instrução RETURN, ele retorna para a linha subseqüente à que provocou o último desvio para a sub-rotina.

Exemplificaremos este tópico com um programa que monta uma tabela do do campeonato nacional, com o nome do time e seus pontos ganhos.

Em determinados pontos do programa precisamos imprimir uma linha de asteriscos e, em outros, apenas três asteriscos, para formar a tabela abaixo:

```
**************************
*                *       *
*     TIME       * PTS   *
*                *       *
**************************
* FLAMENGO       * 10    *
* PALMEIRAS      * 10    *
* ATLETICO       * 10    *
**************************
```

Iniciaremos fazendo o diagrama de blocos (fluxograma) do programa:



Observando o fluxograma você pode notar que o programa aceita a introdução de vários times e ainda finaliza a tabela sem a necessidade de pressionar BREAK ou introduzir STOP.

Para o computador parar o programa verificamos, após cada introdução, se foi digitada a palavra FIM. Poderíamos usar qualquer palavra ou mesmo letra para indicar o término da tabela. Logicamente, o código deve ser compatível com o programa para que não surjam erros na sua interpretação.

Na realidade, a impressão de uma única linha em um programa tão curto não justificaria, em um caso prático, a utilização de sub-rotinas. Porém, nossa intenção é estritamente didática e esse exemplo servirá para dar uma boa idéia do uso de sub-rotinas.

Aqui está o programa:

```
  1 REM CAMPEONATO NACIONAL
 10 GOSUB 120
 20 GOSUB 140
 30 PRINT "*"; TAB 5; "TIME"; TAB 15; "* PTS *"
 40 GOSUB 140
 50 GOSUB 120
 60 INPUT T$
 70 IF T$ = "FIM" THEN GOTO 200
 80 INPUT P
 90 PRINT "*"; T$;TAB 15; "*";P; TAB 21;"*"
100 GOSUB 140
110 GOTO 60
120 PRINT "**********************"
130 RETURN
140 PRINT "*"; TAB 15;"*";TAB 21;"*"
150 RETURN
200 GOSUB 120
```

Esse programa emprega duas sub-rotinas: a primeira, da linha 120, imprime uma seqüência de 22 asteriscos; a segunda (linhas 140/150) imprime as separações entre as colunas.

Analisando o programa principal, vemos que ele chama a sub-rotina de linha 120 para imprimir a linha superior de asteriscos e, em seguida, a da linha 140 para imprimir os três asteriscos de colunamento. O programa segue imprimindo o cabeçalho e repete as sub-rotinas em ordem inversa. Nesse ponto, o programa pára à espera da introdução do nome do primeiro time. O nome é comparado com a palavra FIM. Não sendo o término da tabela, o programa prossegue, pedindo a introdução do número de pontos do time citado. Feito isso, o programa continua, imprimindo os dados, a separação de colunas e retorna para a linha 60, onde serão solicitados os dados de um novo time, recomeçando o loop. O programa permanecerá nesse loop até que seja digitada a palavra FIM, terminando, dessa forma, com a impressão da linha inferior de asteriscos.

Note que esse programa não permite a introdução de nomes muito longos, os quais ficam limitados a 14 caracteres (letras ou números). Também o número de pontos fica limitado em função das demarcações da coluna. Para alterar essas demarcações, aumentando-as ou diminuindo-as, modifique as linhas de impressão.

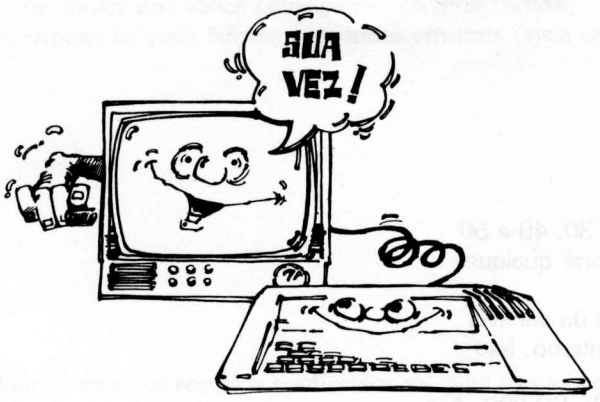
Resumindo o que vimos sob sub-rotinas:

— Devem sempre ser chamadas por uma instrução GOSUB, seguida do número da linha onde começam.

— Devem terminar com uma instrução RETURN.

— Devem vir sempre no fim do programa para facilitar sua interpretação.

— Como norma geral, as sub-rotinas devem ser utilizadas apenas nos casos em que realmente diminuem o número de instruções de um programa.



1. Escreva um programa que imprima os números pares entre 10 e 40.
2. Suponha que você esteja em férias na França e que o câmbio seja de Cr$ 75,00 para cada franco. Escreva um programa que monte uma tabela com as importâncias em francos para diferentes quantias em cruzeiros. Faça a tabela com valores de cruzeiros entre 1000 e 5000, a intervalos de Cr$ 200,00.
3. Reescreva o programa da questão anterior, desta vez fazendo variar a quantia entre 10 e 100 francos, a intervalos de 5 francos.
4. Escreva um programa que imprima, inicialmente, uma linha com 5 asteriscos e, a cada linha, incremente de um o número de asteriscos impressos até atingir um total de 20. O resultado na tela deve ser como mostra abaixo.

```
*****
******
*******
********
*********
**********
***********
************
*************
**************
***************
****************
*****************
******************
*******************
********************
```

5. No programa desse capítulo que imprime as três categorias, cada uma delas com quatro tipos, acrescente duas versões diferentes para cada tipo.

Soluções Propostas

1. 10 FOR P = 10 TO 40 STEP 2
   20 PRINT P
   30 NEXT P
2. 10 PRINT "CRUZEIROS FRANCOS"
   20 FOR C = 100 TO 5000 STEP 2000

```
   30 PRINT C,C/75
   40 NEXT C
3. 10 PRINT "FRANCOS          CRUZEIROS"
   20 FOR F=10 TO 100 STEP 5
   30 PRINT F, F*75
   40 NEXT F
4. 10 FOR N=5 TO 20
   20     FOR A=1 TO N
   30     PRINT "*";
   40     NEXT A
   50 PRINT
   60 NEXT N
```

Colocamos esse programa com espaços nas linhas 30, 40 e 50 apenas para ressaltar o loop interno. O CP-200 vai ignorar qualquer espaço digitado antes de uma instrução.

Um detalhe importante nessa solução é a utilização da variável contadora N do loop externo como limite para o loop interno. Isso mostra a versatilidade da linguagem BASIC do CP-200.

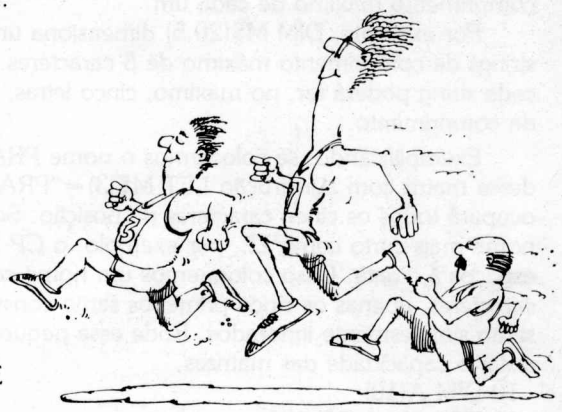
A linha 50 contém outra característica importante do CP-200. A instrução PRINT isolada serve apenas para finalizar cada linha impressa. Tente eliminá-la para ver o que acontece.

```
5. 10 FOR C=1 TO 3
   20 PRINT "CATEGORIA"; C
   30 FOR T=1 TO 4
   40 PRINT TAB 10;"TIPO";T
   50 FOR V=1 TO 2
   60 PRINT TAB 20; "VERSÃO";V
   70 NEXT V
   80 NEXT T
   90 NEXT C
```

Tente colocar a rotina mostrada nos capítulos anteriores para que o CP-200 emita um **bip** a cada impressão (Use, por exemplo, a instrução LET X=USR 30000 nas linhas 15,35 e 55).

Capítulo IX – Matrizes

Quando temos uma série de dados inter-relacionados como, por exemplo, os nomes dos vários participantes de uma corrida, podemos armazenar essas informações todas em uma única variável.

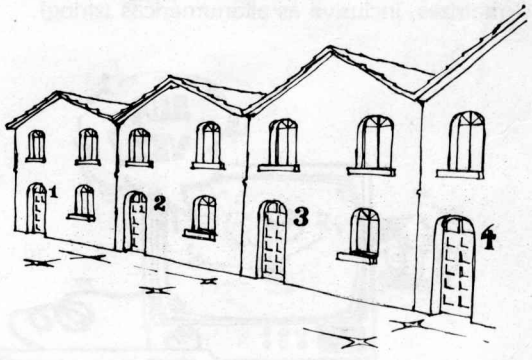


Utilizando-nos dos recursos conhecidos até aqui, para isso, teríamos que colocar cada nome em uma variável string diferente. É claro que isso limitaria o número de corredores, pois o CP-200 só permite nome de variáveis string com uma letra e, portanto, só poderiam concorrer 26 atletas.

No entanto, o BASIC do CP-200 permite que todos esses dados possam ser colocados em uma única variável, que chamamos de **matriz.**

Numa matriz os dados são localizados individualmente através de sua posição. Assim, com um mesmo nome podemos chamar qualquer um dos dados da matriz, diferenciando-o através de um número que será correspondente à sua posição dentro da matriz.

Podemos comparar uma matriz com uma rua qualquer, onde uma casa será indentificada pelo nome da rua e pelo seu número correspondente nessa rua.



Nessa analogia, a rua corresponde à matriz, e as casas, aos dados armazenados.

Para se localizar especificamente uma casa, valemo-nos do nome da rua juntamente com o número da casa em questão; assim, para localizar-se um elemento da matriz, baseamo-nos no nome desta matriz, assim como no número que identifica a posição deste elemento na matriz.

A matriz pode conter dados numéricos ou strings e deve ser definida por uma instrução especial antes de ser usada. Essa instrução é a DIM (tecla D), que é a abreviatura de dimensão ou tamanho da matriz.

Dessa forma, DIM M(20) dimensiona uma matriz numérica que contém 20 números distintos, identificados individualmente como M(1),M(2),...M(20).

Uma matriz string é dimensionada de uma maneira diferente.

Já dissemos anteriormente que uma variável string pode ter qualquer comprimento. Seria muito difícil dimensionar uma sem saber exatamente qual o comprimento máximo de cada string separadamente. Por isso, no CP-200, informamos não apenas o número de strings que a matriz acolherá como também o comprimento máximo de cada um.

Por exemplo, DIM M$(20,5) dimensiona uma matriz para 20 strings de comprimento máximo de 5 caracteres. Isso quer dizer que cada string poderá ter, no máximo, cinco letras, números ou sinais de comprimento.

Exemplificando: se colocarmos o nome PRADO na posição 3 dessa matriz com a instrução LET M$(3) = "PRADO", esse nome ocupará todos os cinco caracteres da posição. Se colocarmos um nome mais curto como ZE, por exemplo, o CP-200 completará com espaços à direita. Caso coloquemos um nome com mais de cinco caracteres, apenas os cinco primeiros serão considerados; os restantes serão simplesmente ignorados. Rode esse pequeno programa para testar a capacidade das matrizes.

```
 10 DIM A(10)
 20 FOR N = 1 TO 10
 30 INPUT A (N)
 40 NEXT N
 50 PRINT "VALOR","RAIZ QUADRADA"
 60 FOR N = 1 TO 10
 70 IF A(N)<0 THEN GOTO 100
 80 PRINT A (N), SQR A(N)
 90 GOTO 110
100 PRINT A(N),"RAIZ IMAGINARIA"
110 NEXT N
```

Nesse programa, usamos um loop FOR/NEXT para pegar todos os valores dentro da matriz. Um outro detalhe importante é a utilização de uma função matemática, a raiz quadrada. Como sabemos, é possível calcular apenas raízes quadradas de valores maiores ou iguais a zero. Esse programa verifica se o número satisfaz essa condição antes de prosseguir com os cálculos.

No exercícios desse capítulo veremos outros usos para as matrizes, inclusive as alfanuméricas (string).

1. Elabore um programa similar a este da raiz quadrada. Porém, faça o seu programa colocar os números em ordem crescente antes de imprimí-los, juntamente com suas respectivas raízes.
2. Elabore um programa que tenha as seguintes características:

a) permita a introdução de até 10 palavras ou frases com um comprimento máximo de 20 caracteres.

b) coloque as frases ou palavras introduzidas em ordem alfabética.

3. Modifique o programa anterior para que o número de palavras introduzidas seja estabelecido no momento em que o programa começa a ser executado.
4. Escreva um novo programa que:

a) Aceite 20 palavras de 5 letras cada uma e as ordene.

b) Imprima 5 palavras, já ordenadas, por linha.

c) Aceite uma nova palavra de 3 linhas.

d) Procure, entre as 20, a palavra de 3 letras.

e) Imprima esta palavra, caso a encontre, ou informe que tal palavra não se encontra entre as introduzidas.

f) Repita os ítens c até e.

Soluções Propostas

1.

```
10 DIM A(10)
20 FOR N=1 TO 10
30 INPUT A(N)
40 IF N=1 THEN NEXT N
50 FOR K=N TO Z STEP -1
60 IF A(K)>= A(K-1)THEN GOTO 100
70 LET Y=A(K)
80 LET A(K)=A(K-1)
90 LET A(K-1)=Y
100 NEXT K
140 NEXT N
150 PRINT "VALOR","RAIZ QUADRADA"
160 FOR N=1 TO 10
170 IF A(N)<O THEN GOTO 200
180 PRINT A(N), SQR A(N)
190 GOTO 210
200 PRINT A(N), "RAIZ IMAGINÁRIA"
210 NEXT N
```

2. Programa:

```
10 DIM F$
20 FOR C=1 TO 10
30 INPUT F$(C)
40 IF C=1 THEN NEXT C
50 FOR K=C TO 2 STEP -1
60 IF F$(K)>=F$(K-1) THEN GOTO 100
70 LET Y$=F$(K)
80 LET F$(K)=F$(K-1)
90 LET F$(K-1)=Y$
100 NEXT K
110 NEXT C
120 FOR C=1 TO 10
130 PRINT F$(C)
140 NEXT C
```

3. Para alterar o programa anterior, digite as seguintes linhas:

```
5 INPUT N
10 DIM F$(N,20)
120 FOR C=1 TO N
```

4. Programa sugerido:

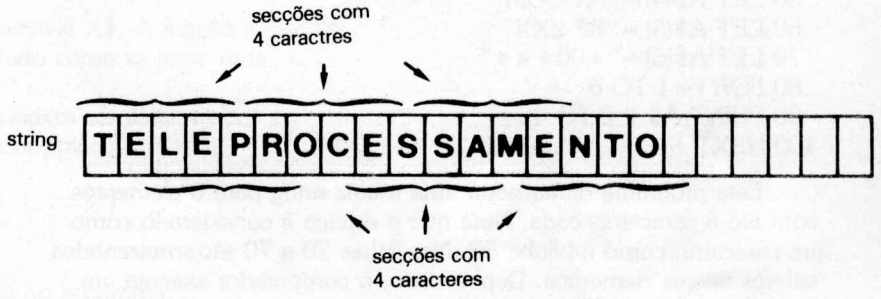
```
 10 DIM F$(20,5)
 20 FOR C=1 TO 20
 30 INPUT F$(C)
 40 NEXT C
 50 FOR N=1 TO 20
 60 PRINT F$(N);" ";
 70 IF N/5=INT(N/5) THEN PRINT
 80 INPUT P$
 90 LET P=0
100 FOR C=1 TO 20
110 IF F$(C,1 TO 3)=P$ THEN GOTO 150
120 IF F$(C,2 TO 4)=P$ THEN GOTO 150
130 IF F$(C,3 TO 5)=P$ THEN GOTO 150
140 GOTO 170
150 REM PALAVRA ENCONTRADA
155 LET P=1
160 PRINT P$,F$(C)
170 NEXT C
180 ROTINA FINAL
190 SCROLL
200 IF P=0 THEN PRINT P$, "NÃO ESTA AQUI"
210 GOTO 80
```

Capítulo X - Manipulação Dados

O CP-200 permite várias operações com os dados introduzidos, mesmo que sejam strings. Uma das mais úteis é a capacidade de se extrair parte de uma string para formar uma nova variável, ou simplesmente para uma comparação.

Essa operação é denominada **secionamento,** pois secionamos uma string maior para obter uma outra, menor. Essa segunda string será parte da outra.

Suponha que a variável string A$ tenha o seguinte conteúdo:



Nessa variável podemos ter uma série de sub-strings que compõem, juntas, a variável original. Podemos ter palavras como: TELE AMEN, etc. Podem ocorrer, ainda, certas combinações que não fazem sentido em português tais como: PROC, CESS, MENT.

Você pode tentar, como exercício, escrever uma longa string de tal forma que toda seqüência de quatro caracteres forme uma palavra conhecida. Depois de elaborada a string, faça um programa que imprima seqüências na tela.

O seccionamento de uma string é realizado indicando-se quais os caracteres que nos interessam, conforme suas posições dentro da string. Para isso, especificamos em quais posições se encontram o primeiro e o último caracteres da seção desejada, da seguinte maneira:

EXPRESSÃO STRING *(início* TO *fim)*

Os valores numéricos início e fim delimitam a seção de string.

Veja esses exemplos:

início fim

"SECIONAMENTO" (2 TO 4) = "ECI"

do 2.º ao 4.º caractere

"SECIONAMENTO" (4 TO 10) = "IONAMEN"

do 4.º ao 10.º caractere

Em caso de omissão dos valores *início* e *fim* serão considerados os próprios início e fim da string original, por exemplo:
"SECIONAMENTO" (TO) = "SECIONAMENTO"

Finalmente, se for omitido apenas um dos delimitadores, a seção se estenderá até o extremo da palavra, por exemplo:

"SECIONAMENTO" (TO 7) = "SECIONA"

até o 7º

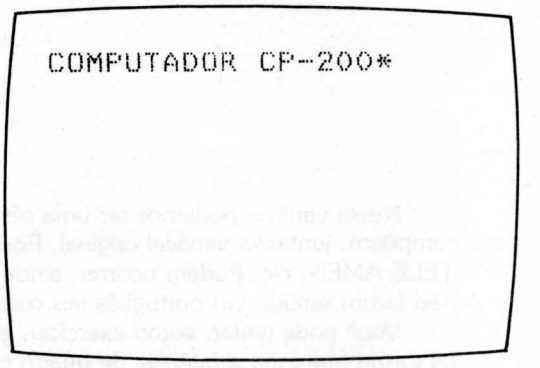
"SECIONAMENTO" (10 TO) = "NTO"

do 10º em diante

Podemos combinar matrizes string com secionamento, como foi feito no último exercício do capítulo anterior. Vejamos um novo exemplo:

```
 10 DIM A$(6,6)
 20 LET A$(1) = "DCOMLD"
 30 LET A$(2) = "7PUTX2"
 40 LET A$(3) = "RADO??"
 50 LET A$(4) = "8R COM"
 60 LET A$(5) = "BP-2XX"
 70 LET A$(6) = "*00***"
 80 FOR I=1 TO 6
 90 PRINT A$ (I,2 TO 4);
100 NEXT 1
```

Esse programa dimensiona uma matriz string para 6 elementos com até 6 caracteres cada. Note que o espaço é considerado como um caractere, como na linha 50. Nas linhas 20 a 70 são armazenados valores nesses elementos. Depois disso, o computador executa um loop (linhas 80 a 100) que imprime consecutivamente as seções de cada elemento da matriz. O resultado, na tela, será:



Além de secionar strings, podemos realizar outras operações de grande interesse para o programador. Podemos **medir** o comprimento das strings através da função LEN (abreviatura de *length*, que quer dizer comprimento). Veja o exemplo:

```
10 INPUT A$
20 PRINT A$, LEN A$
```

O programa pede a introdução de uma string e imprime seu comprimento. Se você, ao rodar o programa, digitar:

EXEMPLO ENTER

O computador imprimirá o número 7. Note que o resultado da função LEN não é uma string, mas, sim, um número, e deve ser tratado como tal. Por isso, nunca use instruções do tipo LET C$=LEN A$, pois estará errado. A forma correta é LET C=LEN A$

Essa função pode ser muito útil quando, durante o programa, precisamos saber o comprimento de determinada string. O programa abaixo pede a introdução de uma string e verifica se existe um ponto de interrogação na mesma.

```
10 INPUT F$
20 LET C=LEN F$
30 FOR P=1 TO C
40 IF F$(C TO C) ="?" THEN PRINT "CONTEM
INTERROGAÇÃO"
50 NEXT P
60 PRINT "BUSCA TERMINADA"
```

Uma outra operação muito útil em programação é efetuada pela função VAL. Essa função transforma uma string em uma expressão numérica.

Por exemplo:

LET X= VAL "SQR 25 + 5"

É o mesmo que

LET X=SQR 25 + 5

Dessa maneira, transformamos a string "SQR 25 + 5" na expressão SQR25 + 5, que, resolvida, resultará em 10.

A função VAL, quando for parte de uma expressão numérica, deve ser o primeiro item da expressão, como mostrado abaixo:

LET X = VAL "SQR 16 + 2"+10

A última operação possível com strings é a transformação de um dado numérico em uma variável string. É o inverso de VAL. Veja o exemplo:

LET X$=STR$ 35

Essa instrução coloca a string "35" na variável X$. A função STR$ permite que um número seja manipulado como se fosse uma string.

Todas as operações com dados strings podem ser usadas em conjunto, sem que haja problemas. Eis um exemplo:

```
10 INPUT A$
20 FOR C=1 TO VAL(STR$ LEN A$ + "—2")
30 PRINT A$(C TO)
40 NEXT C
```

Vamos tentar analisar a expressão usada no programa:

Suponha que A$="ABCDEF", então: LEN A$=6 e

VAL (STR$ 7 + "—2"), continuando:

VAL ("7" + "—2") = 7-2=5

Portanto, a linha 20 ficará: FOR C=1 TO 5.

Armazenamento de Dados

Todos os dados introduzidos no CP-200 precisam ser armazenados de alguma forma na memória do computador. Para isso, o computador os associa a códigos especiais. Assim, cada letra, número, instrução, função ou comando tem seu código específico dentro do CP-200.

Já dissemos, anteriormente, que os dados são armazenados em *bytes*, na memória, e que cada byte pode representar até 256 códigos diferentes. No apêndice C você encontrará a listagem de todos os códigos e caracteres do CP-200.

Há apenas um código para cada caractere do CP-200.

Note que a definição que damos aqui para caractere abrange não apenas letras e números, como também instruções, funções, símbolos gráficos, enfim, toda e qualquer informação que o computador possa entender.

Podemos descobrir qual o código de um determinado caractere com a função CODE seguida do caractere em questão entre aspas.

Essa função fornece um resultado numérico que corresponde ao código, segundo a tabela mostrada no apêndice C. Veja, por exemplo:

PRINT CODE "C" fornece o código 40
PRINT CODE "CP-200" fornece também o código 40

Podemos concluir por esses exemplos que apenas o primeiro caractere é considerado. Se não for colocado nada entre as aspas, o resultado será zero.

A operação inversa (chegar-se a um caractere a partir de código) é realizada pela função CHR$. Esta deve ser seguida do código, entre 0 e 255, e fornece como resultado um valor string correspondente ao caractere relativo ao código. Por exemplo:

PRINT CHR$ 40

Imprime na tela a letra C.

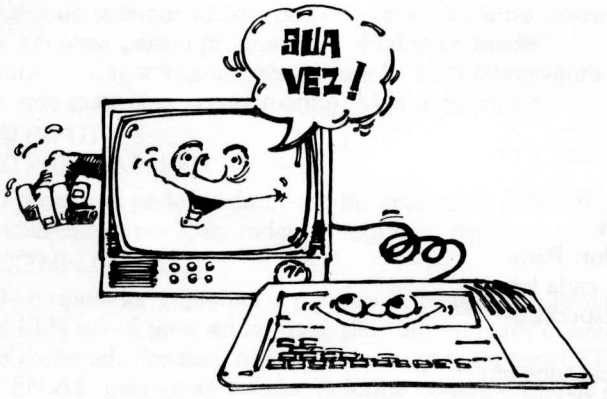
Aqui está um pequeno programa que imprime todos os códigos usados pelo CP-200 e seus caracteres correspondentes:

```
10 FOR C=O TO 255
20 PRINT C;"=";CHR$ C,
30 IF C > 22 THEN SCROLL
```

Não se preocupe com a instrução SCROLL na linha 30.

Ela serve apenas para deslocar o conteúdo da tela para cima, a fim de que nela se abra um espaço para a impressão de novos dados. Isto será explicado em detalhes no próximo capítulo. Portanto, por ora, apenas observe o efeito conseguido rodando o programa.

Esse programa imprimirá uma listagem com cada código associado ao caractere correspondente. Em algumas ocasiões, o computador imprimirá um ponto de interrogação, indicando que esse caractere não tem correspondência. Esse é o caso das setas (teclas 5 a 8), do comando EDIT e de várias outras informações que o CP-200 não coloca na tela.



1. Coloque uma seqüência qualquer dentro de uma string como, por exemplo:

LET Z$="123456789ABCDEF"

A partir daí, elabore uma série de instruções PRINT, tentando obter várias seções da string original. Por exemplo:

PRINT Z$ (3 TO 13)

Você pode tentar também combinações como: (TO 3), (3 TO), (TO), ( ), (4 TO 4), (4 TO 3), (—3 TO 4), (0 TO 8), (1 TO 345), (6 TO —99).

Faça uma análise prévia dos secionamentos, tentando descobrir em quais casos haverá erro de impressão.

2. Tente colocar na tela os símbolos gráficos que aparecem na listagem dos caracteres (Para colocar o cursor no modo gráfico, pressione SHIFT e GRAPHICS — tecla 9. O modo permanece até ser acionado novamente o comando GRAPHICS).

3. Elabore um programa que inverta a ordem das letras de uma palavra introduzida. A palavra poderá ter qualquer comprimento.

4. Escreva um programa que inverta os caracteres de uma string, fazendo com que os caracteres brancos em fundo preto passem a ser pretos em fundo branco, e vice-versa.

No exercício 2, você deve ter descoberto como digitar caracteres invertidos, o que normalmente é realizado pressionando-se as teclas no modo GRAPHICS. Usando SHIFT, podemos digitar os caracteres gráficos. Para esse programa utilize a tabela do apêndice C.

Soluções Propostas

Os exercícios 1 e 2 não oferecem maiores dificuldades, pois você descobrirá as soluções tentando-as no próprio computador. Se tiver dificuldade em sair de alguma situação, pressione as duas teclas RESET simultaneamente, o que fará com que o computador recomece a operar normalmente.

3. Sugerimos o seguinte programa:

```
 10 INPUT P$
 20 PRINT P$
 30 LET M = LEN P$
 40 DIM I$(M,1)
 50 FOR N=0 TO M—1
 60 LET I$(N+1)=P$(M—N TO)
 70 PRINT I$(N+1);
 80 NEXT N
 90 PRINT
100 GOTO 10
```

4. Esta é uma das formas de solução:

```
 10 INPUT P$
 20 PRINT P$
 30 LET M=LEN P$
 40 FOR N = 1 TO M
 50 LET C = CODE P$ (N TO)
 60 IF C>127 THEN LET C=C—256
 70 PRINT   CHR$(C+128);
 80 NEXT N
 90 PRINT
100 GOTO 10
```

Construindo Gráficos com PRINT

Uma das características mais interessantes do CP-200 é sua capacidade de elaborar desenhos na tela, utilizando os caracteres gráficos. Ele permite, inclusive, que um determinado desenho se mova para qualquer direção, aumente de proporções lentamente, ou ainda apareça de repente.

Neste capítulo, veremos como utilizar a instrução PRINT para construir desenhos na tela e, pouco a pouco, iremos descobrindo toda a capacidade do computador em manipulá-los.

Tais desenhos são construídos com caracteres gráficos e são utilizados na representação de funções matemáticas, montagem de tabelas, bem como na elaboração de figuras ilustrativas.

Por uma conveniência didática, denominaremos **gráfico** a qualquer desenho que possa ser feito na tela.

Aconselhamos que você execute cada passo desse capítulo, verificando o resultado na tela para não se perder na seqüência do curso.

Para deixar o cursor no modo gráfico, pressione simultaneamente as teclas SHIFT e GRAPHICS (tecla 9). Dessa forma, ele mudará para [G] Se pressionar qualquer tecla nesse modo, aparecerá na tela o caractere inverso, ou seja, preto em fundo branco.

Você conseguirá um caractere gráfico pressionando SHIFT juntamente com a tela correspondente ao caractere gráfico desejado.

Abaixo temos um programa que constrói uma figura nas linhas 10 a 13, por estarem estas no centro da tela. A variável M, inicialmente, vale zero — o que coloca o nosso trem no canto esquerdo. (Mais adiante você aprenderá a colocá-lo em movimento.).

Esse programa usa: espaços; o número 0 par desenhar as rodas, e alguns dos caracteres gráficos para o desenho do trem:

```
 1 REM **FERROVIA**
10 LET M=O
100 PRINT AT 10,M;"  ▚▛▟▗▙  "
110 PRINT AT 11,M;"  ▜▜▜▜▜▌  "
120 PRINT AT 12,M;"  ▜▜▜▜▜▖  "
130 PRINT AT 13,M;"    00   0  "
```

Vejamos agora como desenhar nossa figura:

— Na linha 100 colocamos o computador no modo gráfico depois de digitarmos o espaço inicial da string. Para tanto, coloque o cursor em [G] e, em seguida, pressione as teclas

E(▛), R(▜), 1(▘), 3(▗), 8(▌) para os caracteres gráficos.

Ainda com SHIFT pressionada, digite GRAPHICS novamente para sair do modo gráfico, ficando o cursor em . Solte a tecla SHIFT e digite o espaço e as aspas para terminar a linha. Nas linhas 110 e 120 os quadros são conseguidos com a tecla de espaço no modo gráfico (cursor).

Na linha 110 o string entre aspas é formado por um espaço, cinco quadrados, o caractere gráfico da tecla 5 () e um espaço final.

Na linha 120 o string é formado por um espaço, cinco quadrados, o caractere gráfico da tecla W () e um espaço final.

Na linha 130 são dois espaços, dois zeros, um espaço, um zero e um espaço final.

Resumindo, as teclas que devem ser pressionadas entre aspas são:

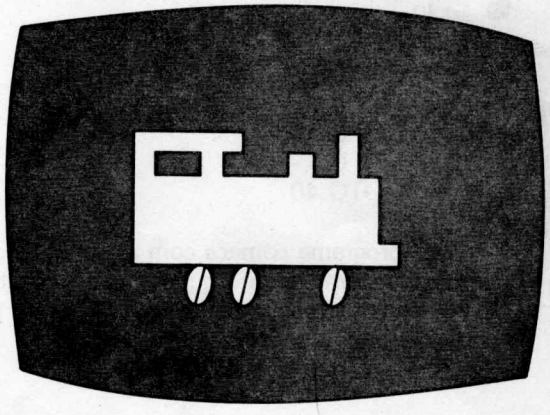
linha 100: SHIFT
" espaço 9 E R 1 3 8 9 espaço "

linha 110: SHIFT SHIFT
" espaço 9 **5espaços** tecla 5 9 espaço "

linha 120: SHIFT SHIFT
" espaço 9 **5espaços** W 9 espaço "

linha 130:
" 2 espaços 00 espaço 0 espaço "

Os espaços são importantes para a correta apresentação da figura. O resultado na tela deve ser semelhante ao que segue:



Faremos agora com que o trem se mova. As duas instruções a seguir fazem com que ele percorra toda a tela.

```
10 FOR M=0 TO 24
140 NEXT M
```

Ao chegar no canto oposto da tela, ele pára.

Nesse ponto poderíamos fazer com que ele andasse em sentido contrário, retornando ao canto esquerdo da tela. Para isso necessitamos de um novo loop FOR/NEXT que realize o movimento nesse sentido. Dessa forma teríamos dois loops, um realizando o movimento em um sentido, e o outro no sentido inverso.

FOR M=0 TO 22 STEP 1 →

FOR N=22 TO 0 STEP −1 ←

Poderíamos escrever duas vezes as rotinas de impressão, uma vez para cada loop; no entranto, já sabemos como contornar esse

problema e, por isso, usaremos as linhas de impressão como sub-rotina.

Por exemplo:

```
 10 FOR M=0 TO 22
 20 GOSUB 100
 30 NEXT M
 40 FOR M=22 TO 0 STEP -1
 50 GOSUB 100
 60 NEXT M
 70 GOTO 10
 99 REM **IMPRIME O TREM**
100     .
        .
        .
140 RETURN
```

Uma outra solução seria manter a rotina de impressão dentro de um único loop FOR/NEXT controlado por variáveis, as quais seriam alteradas após cada impressão de forma a inverter o movimento. Veja o programa:

```
 10 LET A=0
 20 LET B=22
 30 LET C=1
 40 FOR M=A TO B STEP C
 99 REM IMPRIME O TREM
100     .
110     .
120     .
130     .
140 NEXT N
150 LET D=A
160 LET A=B
170 LET B=D
180 LET C=-C
190 GOTO 40
```

Este programa começa com as variáveis definidas de tal maneira que o trem se move da esquerda para a direita.

Se substituirmos as variáveis na linha 40 pelos seus valores iniciais, teremos a seguinte instrução:

```
FOR M=0 TO 22 STEP 1
```

Quando o trem estiver impresso, as variáveis A e B terão seus valores trocados, utilizando D como variável de armazenamento temporário (linhas 150 a 170). O conteúdo de C é multiplicado por -1 (LET C = -C) para inverter o sentido da contagem do loop. Feito isso, o programa retorna à linha 40, que com as alterações realizadas equivalerá a:

```
FOR M=22 TO 0 STEP -1
```

O programa continua, realizando o loop FOR/NEXT e, quando sair do loop, novamente as variáveis serão trocadas, e assim por diante.

Qualquer método adotado fará com que o trem se mova continuamente na tela, de um lado para outro.

Os programas acima mostram apenas algumas alternativas para se conseguir a animação de figuras com o seu CP-200.

Podemos limpar a tela com a inscrição CLS antes de cada impressão. Isso fará com que o trem desapareça e torne a aparecer continuamente. O momento em que ele aparece na tela é quando o imprimimos. Podemos diminuir a intensidade dessa cintilação na tela fazendo com que o CP-200 permaneça um determinado tempo sem movimentar a imagem. Isso é realizado com a instrução PAUSE, a qual faz o computador mostrar o conteúdo da tela por um certo tempo.

Esse intervalo de tempo é especificado indicando-se o número de quadros que o computador deve colocar na tela antes de continuar com o programa.

O CP-200 coloca 60 quadros por segundo na tela do seu televisor, portanto:

PAUSE 12	indica uma pausa de 0,2 segundos
PAUSE 30	indica uma pausa de 0,5 segundos
PAUSE 60	indica uma pausa de 1,0 segundo
PAUSE 300	indica uma pausa de 5 segundos

Você ainda poderá fazer uso de um recurso para modificar o modo como o trem aparece na tela.

Até aqui, temos visto o computador imprimir nosso trem gradualmente, mas podemos fazer com que ele apareça de uma só vez na tela.

Isso torna-se possível graças a duas modalidades de processamento do CP-200: FAST e SLOW.

Quando você liga o CP-200, ele começa a operar na velocidade SLOW, processando os dados e controlando a tela ao mesmo tempo.

O comando FAST (tecla F) coloca o CP-200 na velocidade rápida de processamento (quatro vezes maior que SLOW), em que ele processa os dados sem "preocupar-se" com a tela.

O comando SLOW (tecla D) faz com que o computador retorne à velocidade normal.

Para facilitar a memorização destes comandos, lembre-se que SLOW e FAST singificam, respectivamente, lento e rápido.

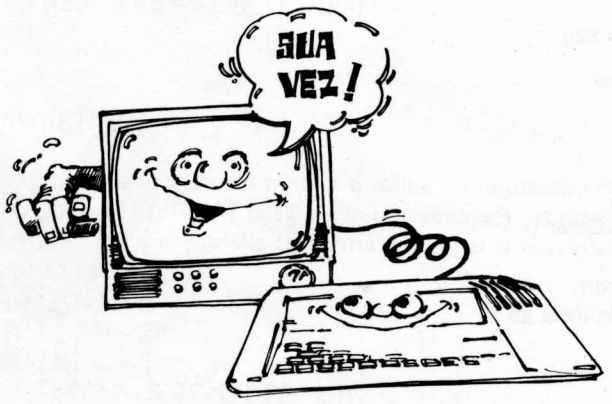
Os comandos SLOW e FAST podem ser utilizados num mesmo programa, variando-se a velocidade de processamento de acordo com as necessidades.

Veja o exemplo abaixo:

```
 10 FOR M=0 TO 22
 15 FAST
 20 GOSUB 100
 30 NEXT M
 40 FOR M=22 TO 0 STEP -1
 45 FAST
 50 GOSUB 100
 60 NEXT M
 70 GOTO 10
 99 REM **IMPRIME O TREM**
100     .
110     .
120     .
130     .
135 SLOW
138 PAUSE 6
140 RETURN
```

Quando estiver usando a velocidade FAST e utilizar a instrução PAUSE, não se esqueça de, logo após essa instrução, colocar a seguinte linha: POKE 16437,255. Sem ela, seu programa poderá ser apagado da memória.

Não é aconselhável mudar frequentemente a velocidade de operação do computador em um mesmo programa, pois isso provocará uma vibração indesejável da imagem. Recomendamos que você opte por uma das velocidades e trabalhe unicamente com ela.



1. Utilizando a instrução SCROLL que faz com que o conteúdo da tela seja deslocado para cima, acrescente algumas linhas ao programa do trem para que ele solte fumaça ao se deslocar.

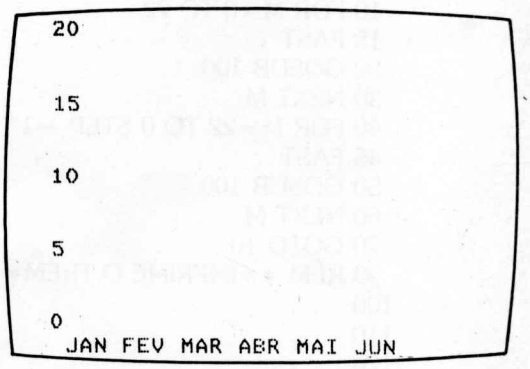
2. Escolha um veículo qualquer (avião, navio, carro, etc.) e faça-o movimentar-se na tela de cima para baixo.

3. Com o veículo do problema anterior, faça-o mover-se de um lado para outro da tela.

4. Escreva um programa que combine as características dos dois exercícios anteriores.

5. Escreva um programa que gere um gráfico de barras para valores entre 1 e 20, referentes ao faturamento de 6 períodos diferentes como, por exemplo, os meses de janeiro a junho.

O programa, antes da introdução dos dados, desenha o perfil do gráfico na tela, conforme o modelo abaixo:



6. Modifique o programa acima de modo a fazer barras mais largas. Tente vários caracteres gráficos e observe os efeitos conseguidos.

7. A crescente ao programa do Trem o pequeno programa que gera o *"bip"* sonoro, fazendo com que o trem *apite*.

Soluções Propostas

1. Programa sugerido:

```
 10 FOR M=0 TO 22
 22 GOSUB 90
 30 NEXT M
 40 FOR M=22 TO 0 STEP -1
 50 GOSUB 90
 60 NEXT M
 70 GOTO 10
 90 SCROLL
 95 PRINT AT 9,M;" * "      Obs: 5 espaços, um asterisco e
                                 dois espaços
100 PRINT AT 10,M; "  "     Obs: finalizar com dois espaços
.....
.....
.....
140 RETURN
```

Os exercícios de número 2, 3 e 4 ficam por conta de sua imaginação. Baseie sua solução na questão anterior.

5. Programa sugerido:

```
 10 REM GRÁFICO DE BARRAS
 20 PRINT AT 0,0;20;AT 4,0;15;AT 9,0;AT 14,0;5;AT 19,0;0
 30 PRINT AT 20,4;"JAN FEV MAR ABR MAI JUN"
 40 FOR N=1 TO 6
 50 INPUT X
 60 IF X > 20 OR X < O THEN GOTO 120
 70 FOR K=1 TO X
 80 PRINT AT 20—k, 4*N+1; "■ "  Obs: caractere gráfico
                                    da tecla 5.
 90 NEXT K
100 NEXT N
110 STOP
120 PRINT AT 0,5,"***ERRO—";X;"IGNORADO***"
```

6. Você pode utilizar o caractere gráfico da tecla de espaço para conseguir barras mais largas.

7. Acrescente as linhas

```
  1 POKE 30000 211
  2 POKE 30001, 240
  3 POKE 30002, 201
135 LET X=USR 30000
```

Plotando Gráficos

A este estágio do nosso curso, você já deve estar dominando o seu CP-200!

Você já sabe elaborar programas sofisticados, usando loops controlados ou utilizando-se de sub-rotinas; sabe construir os mais diversos gráficos na tela, e ainda pode fazer com que se movam.

Contudo, existe ainda uma limitação para o número de cacarcteres gráficos que você pode imprimir na tela com a instrução PRINT.

Como você já sabe, a tela dispõe de 22 linhas com 32 caracteres cada uma delas. Veja isso no exemplo abaixo que imprime na tela a curva representativa da função seno (SIN), normalmente conhecida por senóide.

```
10 FOR N=0 TO 31
20 PRINT AT 11-10*SIN (N*2*π/31), N;"*"
30 NEXT
```

Esse programa imprime uma série de pontos, formando uma curva na tela. Observe que os pontos estão relativamente distantes entre si, quando o ideal seria que formassem uma linha contínua.

O CP-200 possui funções que permitem aumentar o número de pontos colocados na tela, melhorando, assim, o aspecto da curva.

Veja o programa abaixo:

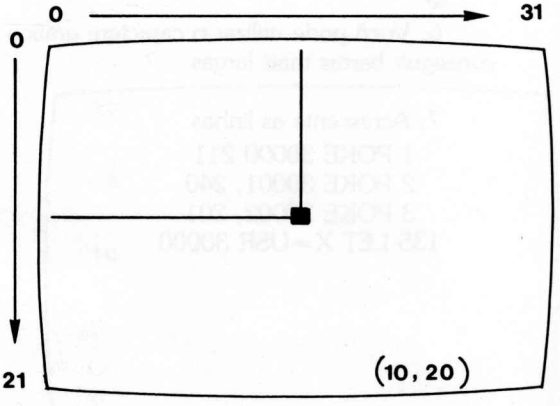
```
10 FOR X=0 TO 63
20 PLOT X,22+20*SIN(X/32*π)
30 NEXT X
```

A instrução PLOT (em inglês, plotar) ativa um ponto gráfico em uma determinada posição da tela. Tente a linha imediata:

```
PLOT 32,22
```

Essa instrução fará aparecer um pequeno caractere gráfico no centro da tela.

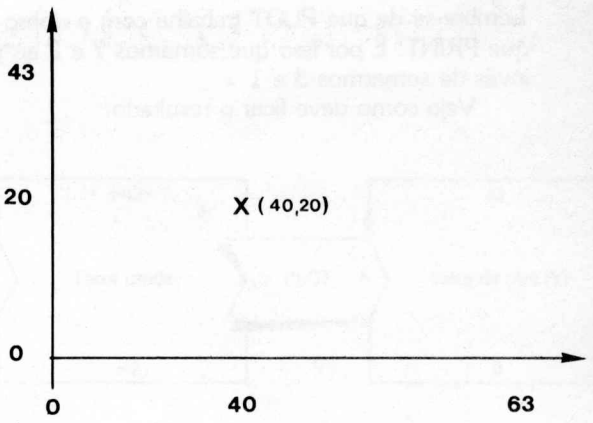
Esse caractere equivale a um quarto dos caracteres normais. A instrução PLOT considera a tela dividida em 44 linhas, cada uma comportando até 64 **pontos gráficos**. Em uma impressão normal, a tela é dividida da seguinte maneira.



Para especificar-se uma determinada posição, contamos as linhas de cima para baixo e as colunas da esquerda para a direita.

Quando queremos **plotar** um determinado ponto na tela, usamos como base para o seu posicionamento um plano cartesiano; isto é feito definindo-se os eixos X (horizontal) e Y (vertical). A

divisão considerada pela instrução PLOT ficará, então:



No exemplo mostrado na figura acima, o primeiro número 40, representa a posição em relação ao eixo X; o segundo número 20, representa a posição em relação ao eixo Y. A esses dois números denominamos par ordenado. Nós o usaremos na instrução PLOT.

Observe que o primeiro número, ao contrário do que ocorre com PRINT AT, indica a coluna e o segundo indica a linha onde aparecerá o ponto.

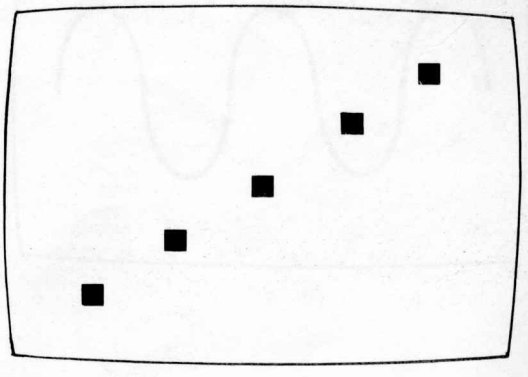
Para se apagar um determinado ponto gráfico da tela, usamos a instrução UNPLOT, que significa **apagar um ponto plotado.**

Vamos utilizar este programa de conversão de temperatura para traçar uma curva que relaciona graus Celsius a Fahrenheit.

Eis o programa:

```
100 REM "X=FAHRENHEIT, Y=CELSIUS"
119 FOR Y=0 TO 30 STEP 5
120 LET X=Y*9/5
125 REM ZERO GRAUS CELSIUS ESTA NO CANTO INFERIOR
ESQUERDO
130 PLOT X,Y
140 NEXT Y
```

Rode o programa e veja o resultado na tela:



Esse resultado, tal como aparece na tela, está incompleto.

Acrescente as linhas abaixo para melhorar o gráfico:

```
10 PRINT AT 5,0;30;AT 10,0;20;AT 15,0;10;AT  20,0;0
20 PRINT AT 21,3;"32 40 50 60 70 80"
```

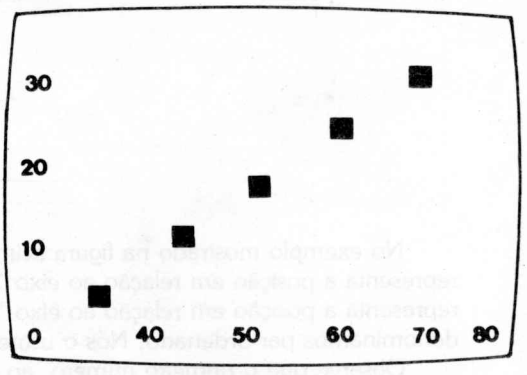
Obs: Deixe sete espaços entre 32 e 40 e nove espaços entre os demais números. Altere também a linha 120 para:

```
120 PLOT X+7,Y+2
```

Essa alteração é importante, pois evita que o gráfico seja plotado sobre a legenda (que é impressa pelas linhas 10 e 20).

Lembre-se de que PLOT trabalha com o dobro de linhas e colunas que PRINT. É por isso que somamos 7 e 2 ao par ordenado, ao invés de somarmos 3 e 1.

Veja como deve ficar o resultado:



Altere agora a linha 100 para que sejam plotados todos os valores de Y entre O e 30.

```
100 FOR Y=0 TO 30
```

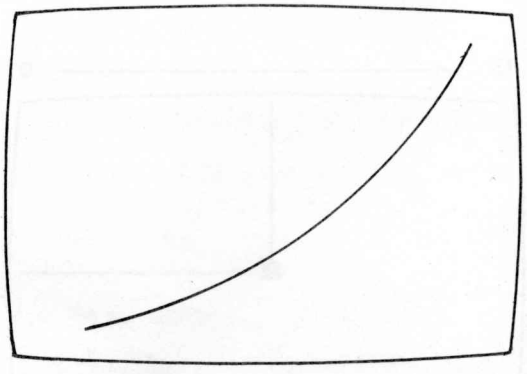
Você notará, ao rodar o programa, que a linha resultante na tela não é retilínea. Isto se deve ao fato dos pontos plotados serem todos arredondados para o número inteiro mais próximo. Assim, se Y for 9, X será igual a 16.2, e o valor plotado será 16. Caso Y seja 11, X será igual a 19.8, e o valor plotado será 20.

Vejamos agora como plotar a curva de $Y = X^2$. O primeiro problema é que $7^2 = 49$ e não será possível plotar esse ponto na tela.

Como solução, podemos dividir todos os Y por 100. Veja o progrma que sugerimos:

```
10 FOR X=0 TO 63
20 PLOT X,X**2/100
30 NEXT X
```

Rode o programa e veja se o resultado é parecido com o da figura abaixo:

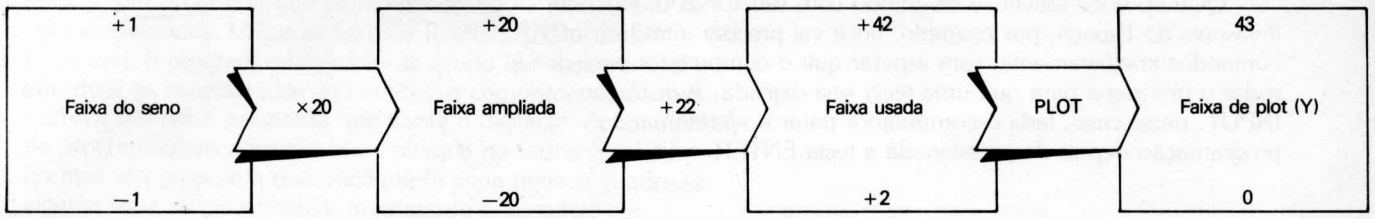


As funções como seno e co-seno produzem curvas características, porém elas mantêm os resultados entre 1 e −1. Portanto, precisamos adaptar os resultados à nossa tela. Reveja o programa do início deste capítulo.

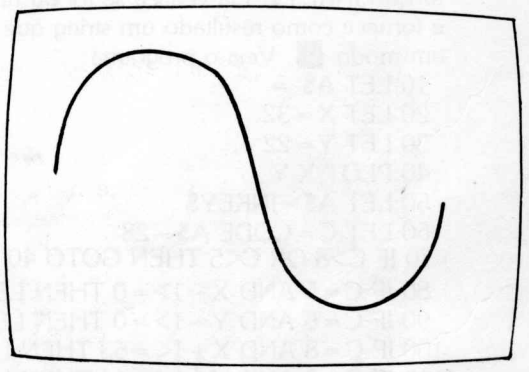
```
10 FOR X=0 TO 63
20 PLOT X,22+20*SIN(X/32*π)
30 NEXT X
```

A expressão usada na linha 20 multiplica os resultados de SIN por vinte, resultando em uma nova faixa, 20 a −20. Essa faixa é

alterada através da soma com 20 para 2 a 42. Como os números podem ser plotados na vertical entre 0 e 43, essa faixa é perfeitamente aceitável.

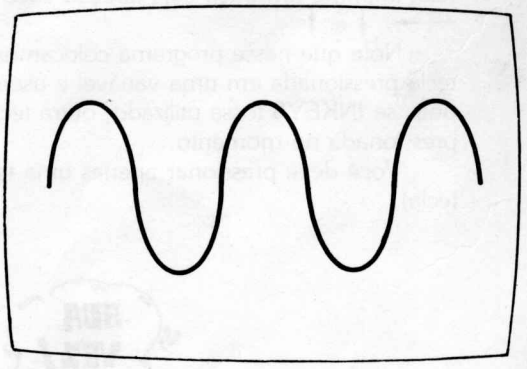


Abaixo mostramos algumas figuras que podem ser desenhadas com as funções seno e co-seno.



Mudando a linha 20, podemos fazer com que apareçam mais que uma senóide na tela:

20 PLOT X,22 + 20*SIN(X/32*π)



Resumindo o que vimos sobre as instruções PLOT e UNPLOT

— A instrução PLOT coloca um ponto gráfico na tela;

— A instrução UNPLOT apaga o ponto definido pelo par coordenado que a segue. Por exemplo: PLOT 20,40 ativa o ponto colocado na posição 20 (horizontal) e 40 (vertical); UNPLOT 20,40 desativa esse mesmo ponto.

— Essas instruções dividem a tela em 44 linhas (numeradas de 0 a 43) e 64 colunas (numeradas de 0 a 63).

Digitando Sem Interromper o Programa

Quando você estiver se divertindo com seu CP-200, jogando invasores do Espaço, por exemplo, você vai precisar introduzir os comandos imediatamente, sem esperar que o computador pare de rodar o programa para que uma tecla seja digitada. A instrução INPUT, nesse caso, faria o computador parar e só retomaria a programação depois de pressionada a tecla ENTER.

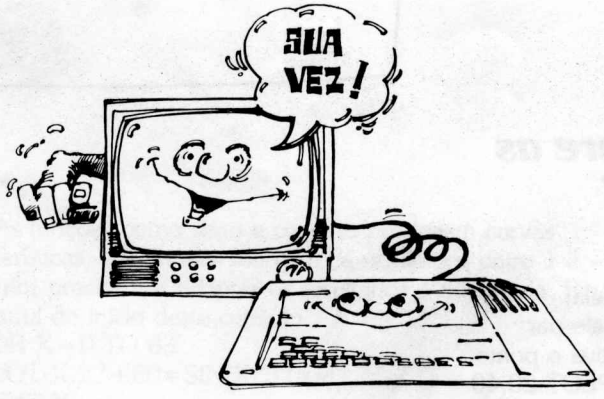
Para suprir essa necessidade, existe no BASIC do CP-200 a função INKEY$. Ela verifica se foi ou não pressionada alguma tecla, e fornece como resultado um string que equivale à tecla pressionada em modo L. Veja o programa:

```
 10 LET A$ = ""
 20 LET X=32
 30 LET Y=22
 40 PLOT X,Y
 50 LET A$=INKEY$
 60 LET C=CODE A$-28
 70 IF C>8 OR C<5 THEN GOTO 40
 80 IF C=5 AND X-1>=0 THEN LET X=X-1
 90 IF C=6 AND Y-1>=0 THEN LET Y=Y-1
100 IF C=8 AND X+1<=63 THEN LET X=X+1
110 IF C=7 AND Y+1<=44 THEN LET Y=Y+1
120 GOTO 40
```

Este programa, partindo de um ponto localizado no centro da tela, imprime um traço cuja direção você orientará através das teclas ← → ↓ e ↑

Note que nesse programa colocamos o código correspondente à tecla pressionada em uma variável e usamos essa mesma variável, pois, se INKEY$ fosse utilizado, outra tecla poderia estar sendo pressionada no momento.

Você deve pressionar apenas uma tecla por vez (ou SHIFT e a tecla).



1. Escreva um programa que desenhe um círculo de raio igual a 20

e centro em (30,20). A fórmula para o círculo é:
$(x-n)^2 + (y-k)^2 = r^2$, onde:
x e y são as coordenadas dos pontos pertencentes ao círculo;
n e k são as coordenadas do centro da circunferência;
r é o comprimento do raio.

2. Elabore um programa que plote uma figura na tela a partir de valores aleatórios. Utilize as funções RAND, RND e PLOT.
3. Reescreva o programa do círculo de modo que seja possível introduzir as coordenadas do centro e o comprimento do raio.
4. Escreva um novo programa que aceite a digitação de uma linha de programa com a função a ser plotada na forma Y=f(X).
5. Escreva um programa que verifique, a cada meio segundo, se alguma tecla foi pressionada, imprimindo o caractere correspondente ou "NADA PRESSIONADO".

Soluções Propostas

1. Programa sugerido:

```
 50 FOR X=10 TO 50
 60 LET Y=20 + SQR (400 - (X-30)*(X-30))
 70 LET Z=20 - SQR (400 - (X-30)*(X-30))
 80 PLOT X, Y
 90 PLOT X, Z
100 NEXT X
```

2. Programa sugerido:

```
 10 RAND
 20 LET X=RND
 30 LET Y=RND
 40 PLOT INT(X*64),INT(Y*44)
 50 GOTO 10
```

3. Alterações sugeridas:

```
 10 PRINT "CENTRO EM (";
 15 INPUT N
 20 PRINT N;",";
 25 INPUT K
 30 PRINT K;")-RAIO="
 40 INPUT R
 45 PRINT R
 50 FOR X=0 TO 63
 60 LET D=R*R-(X-N)*(X-N)
 70 IF D<0 THEN NEXT X
 80 LET Y=K + SQR D
 90 LET Z=K + SQR D
100 IF Y>43 OR Y<0 THEN GOTO 120
110 PLOT X,Y
120 IF Z>43 OR Z<0 THEN NEXT X
130 PLOT X,Z
140 NEXT X
```

4. Programa sugerido:

```
 10 PRINT "ESCREVA A FORMULA NA LINHA 30 COM LET"
 20 FOR X=0 TO 63
 30 LET Y=.....
 40 IF Y>43 OR Y<0 THEN NEXT X
 50 PLOT X,Y
 60 NEXT X
```

Digite esse programa com a função cuja curva você quer plotar na linha 30, usando para isso uma instrução do tipo LET Y = ...

Exemplo:

Função $y = x^2/100$

```
30 LET Y = X * X/100
RUN ENTER
```

5. Programa sugerido:

```
10 PAUSE 20
20 LET A$ = INKEY$
30 IF A$ = "" THEN GOTO 60
40 PRINT A$
50 GOTO 10
60 PRINT "NADA PRESSIONADO"
70 GOTO 10
```

Como Funciona o CP-200 por Dentro

Não é objetivo deste manual descrever em detalhes o funcionamento interno do CP-200. No entanto, neste capítulo, tentaremos dar uma boa idéia de sua operação.

Os componentes retangulares com terminais de metal são os circuitos integrados, chamados CI's — na realidade, pode-se ver apenas o encapsulamento externo do CI; o circuito propriamente dito, é muito menor.

NOTA — Não desmonte seu CP-200, pois isso incorrerá em perda de garantia.

O principal CI do CP-200 é a UCP (Unidade Central de Processamento) que é um microprocessador Z-80A. O processador é responsável por todas as operações aritméticas e lógicas do computador, além de controlar todas as outras partes do CP-200. Essas operações e controles seguem uma programação definida, gravada na memória ROM.

Esta programação é chamada **Sistema Operacional**. O sistema operacional do CP-200 está gravado em EPROM, que é um dispositivo eletrônico de armazenamento similar a ROM.

Este sistema operacional é codificado dentro da EPROM como uma seqüência de **byte** (veja definição de byte no glossário). Cada byte é localizado dentro da EPROM através de **endereços**. O primeiro byte tem endereço 0, o segundo, 1, e assim por diante até 8191. Isso indica que o CP-200 tem um sistema de operação de 8000 bytes ou, simplesmente, 8 KB.

O sistema de operação do CP-200 comporta o programa **monitor** e o **interpretador** do BASIC, e é ele o responsável pelo funcionamento do CP-200.

Você pode saber que o valor está armazenado em um dado byte através de função PEEK. Veja o programa abaixo que imprime os valores dos primeiros 40 bytes do sistema operacional.

```
10 PRINT "ENDEREÇO"; TAB 11; "BYTE"; TAB 16;
"ENDEREÇOS"; TAB 27, "BYTE"
20 FOR A=0 TO 20
30 PRINT A; TAB 21: PEEK A; TAB 16; A+20; PEEK (A+20)
40 NEXT A
```

Sob o controle desse sistema de operação, o computador recebe os dados, verifica sua validade, controla o conteúdo da tela, controla os sinais do gravador, processa as informações e fornece os resultados. Para fazer tudo isso, é necessário um **rascunho**, um lugar onde o computador possa colocar os resultados intermediários, e mesmo um local onde armazenar todas as informações processadas, sejam programas ou dados. Esta área onde é possível colocar toda essa informação é chamada memória RAM.

Como na EPROM, as informações são armazenadas em códigos nos bytes; cada byte com seu endereço. Esses endereços variam de 16384 e 32767. Como na EPROM, pode-se verificar o conteúdo da RAM com a função PEEK.

Além disso, podemos ainda alterar o conteúdo de um dado endereço da RAM.

Para colocar um dado valor em um byte de memória usamos a

instrução POKE.

Note a diferença entre função e instrução. A função fornece um resultado, e, por isso, deve ser utilizado dentro de uma instrução, como a PRINT, por exemplo.

Uma instrução pode ser introduzida diretamente em uma linha. Portanto, podemos digitar simplesmente:

POKE 20000,38

Essa instrução coloca o código 38 no byte de endereço 20000 da RAM.

É importante salientar que os primeiros 16 mil bytes não podem ter seus conteúdos alterados. Os endereços até 8191 são da EPROM e, portanto, são apenas de leitura. Os 8 mil seguintes não são utilizados no seu CP-200.

Para verificar o funcionamento da instrução POKE, digite:

PRINT PEEK 20000

O resultado será o número 38 impresso no alto da tela. Você pode repetir a experiência para outros endereços, porém deve levar em consideração que certos endereços possuem informações importantes para o bom funcionamento do CP-200. Mais à frente daremos os endereços dos bytes que contêm estas informações. Esses bytes são conhecidos como variáveis de sistema.

Para colocar um valor através de POKE existem certas restrições. Em primeiro lugar, o endereço deve ser um valor entre 0 e 32767. O valor a ser colocado deve estar entre —256 e 255. Para valores negativos, o computador adiciona 256.

A instrução POKE abre um imenso campo na utilização das capacidades do CP-200. Entretanto, o conhecimento para uma completa utilização dessa capacidade é muito extenso para ser abordado em um livro introdutório como este.

Aconselhamos àqueles que se interessarem em dominar completamente o CP-200 que estudem atentamente este livro, testando novas soluções para os exercícios propostos e utilizando variações das características do computador, através da alteração dos conteúdos das variáveis do sistema.

Organização da Memória

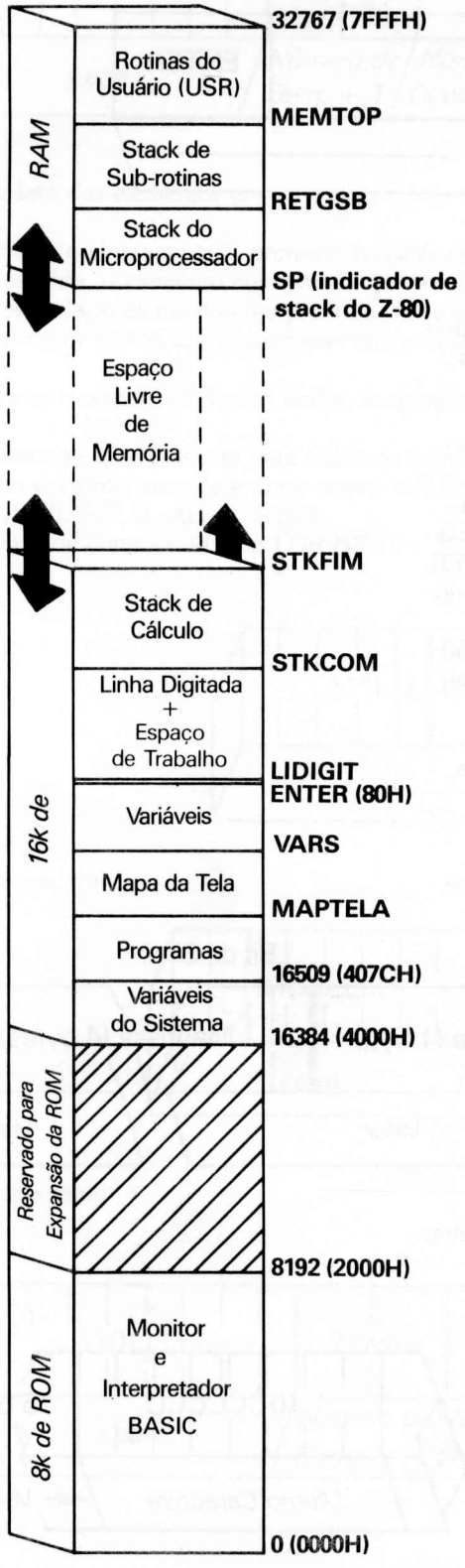
A memória é dividida em diferentes áreas que armazenam diferentes tipos de informação. Cada área é grande o bastante para conter apenas a informação essencial; a cada nova informação introduzida, a área é aumentada ou, em caso contrário, diminuída. O espaço é criado ou suprimido deslocando-se todas as áreas simultaneamente. A figura abaixo mostra uma representação de toda a memória do CP-200, incluindo RAM e ROM. O bloco da ROM e da expansão é fixo, começando em 0 e terminando em 16383.

No bloco da RAM é que ocorre a variação dos endereços, com conseqüente deslocamento das várias áreas. As extremidades são fixas, começando em 16364 e terminando em 32767.

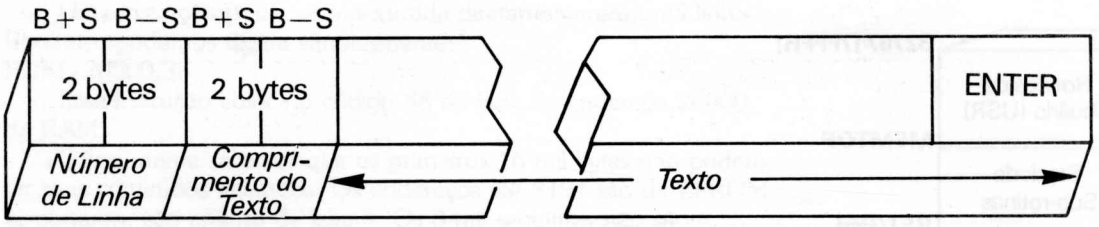
Digamos que você digite uma linha. Isso fará com que a área reservada para o texto que está sendo introduzido (delimitada pelas variáveis LIDIGIT e STKCOM) aumente e, com ela, a área do **stack** do microprocessador se deslocará, diminuindo a área livre de memória.

As variáveis do sistema contêm as informações que dizem ao computador como ele se encontra em um dado momento. Serão todas listadas e descritas em detalhes mais à frente. No momento, é mais importante entender como são armazenadas as informações

dentro das áreas principais de armazenamento. É importante, ainda, dizer que os nomes das variáveis do sistema não são reconhecidos pelo computador. Para se verificar qualquer dos valores, devemos nos valer da função PEEK.



O armazenamento de uma linha de programa dentro da área que começa no endereço 16509 e termina no endereço indicado pela variável TELA, é feito da seguinte forma:



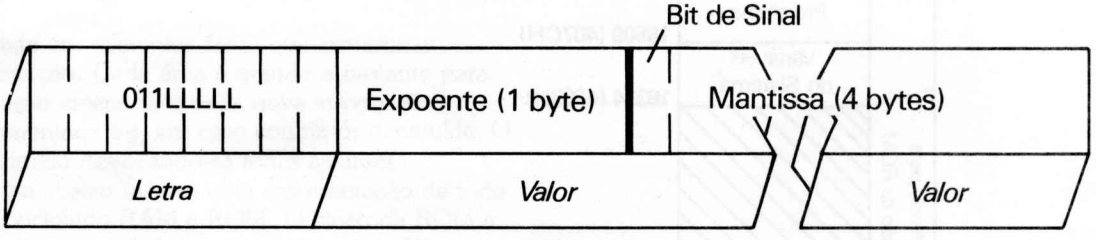
Note que, contrário a todos os outros casos com valores de dois bytes, o número da linha é armazenado no Z-80 com o byte mais significativo (B + S) em primeiro lugar.

Uma constante numérica dentro da linha é identificada pelo código 126 e é representada por cinco bytes para o número, como uma variável numérica (veja adiante).

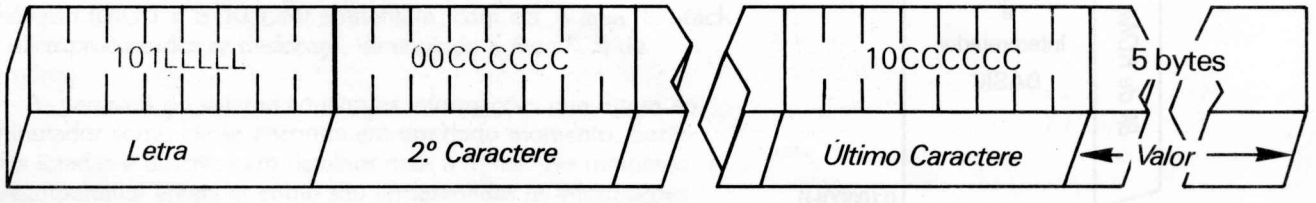
O arquivo da tela é a cópia, na memória, da imagem na tela. Começa com um ENTER, seguida de 24 linhas de texto, cada uma com 32 caracteres de comprimento, terminadas também por ENTER. Esse arquivo pode ser diminuído substancialmente, pois depende do total de memória disponível, definida pela variável MEMTOP. Quando essa variável indica que estão disponíveis menos que 3250 bytes de memória, a limpeza da tela (CLS) é realizada deixando-se apenas os caracteres ENTER no arquivo. Em caso contrário, a limpeza é realizada com 24 e 32 espaços (768 bytes). A instrução SCROLL e certas condições que alteram as duas últimas linhas da tela podem produzir linhas curtas na parte inferior da tela.

As variáveis têm formatos diferentes, dependendo da sua natureza, como é demonstrado abaixo:

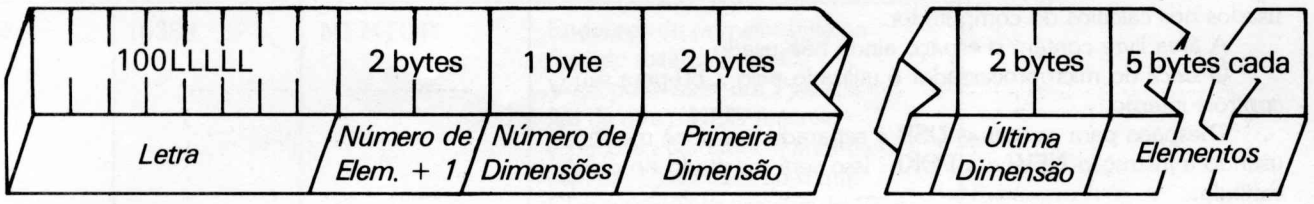
A variável numérica com nome de apenas uma letra, tem este formato:



Caso o nome da variável numérica contenha mais de uma letra, o formato será o seguinte:



Matrizes numéricas têm o seguinte formato:



A ordem dos elementos em uma matriz, seja numérica ou string, é:

Primeiro, o elemento cujo primeiro índice é um;

Em seguida, o elemento cujo primeiro índice é dois;

Em seguida, o elemento cujo primeiro índice é três;

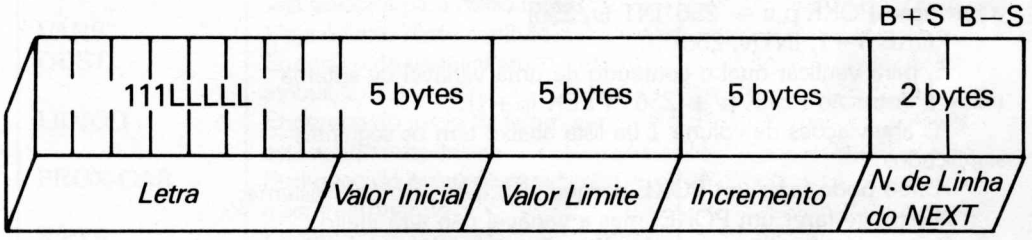
E assim por diante, até se percorrer todos os valores do primeiro índice.

Altera-se o segundo índice e, então, torna-se a variar o primeiro índice.

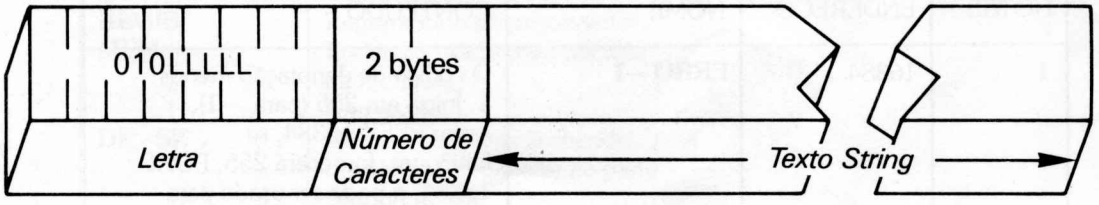
Continua-se esse processo para todos os índices.

Como exemplo, consideremos a matriz A(2,3), teremos: A(1,1), A(2,1), A(1,2), A(2,2), A(1,3), A(2,3).

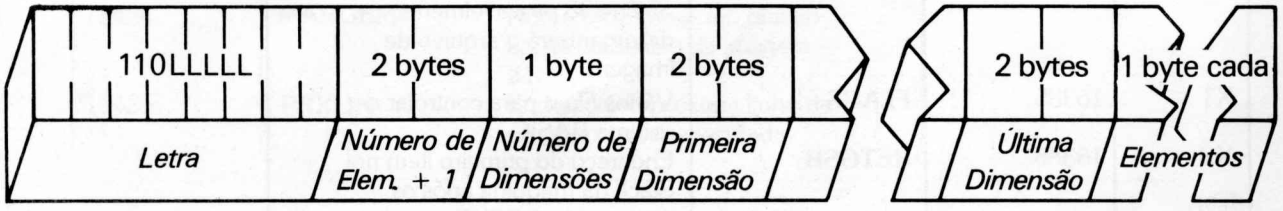
Variável de controle de loop FOR-NEXT



Variáveis string



Matrizes string



A área iniciada pelo endereço em LIDIGIT contém a linha que está sendo digitada, além de algum espaço de trabalho.

O stack de cálculo contém todas as informações e os valores usados nos cálculos do computador.

A área livre contém o espaço ainda não usado.

O *stack* do microprocessador é utilizado pelo Z-80 para seu controle interno.

O espaço para as rotinas USR é separado por você mesmo, usando a instrução NEW e a POKE. Isso será mostrado no próximo capítulo.

Variáveis do Sistema

As variáveis do sistema estão armazenadas na memória RAM entre os bytes de número 16384 e 16508.

Mas, cuidado! Não as confunda com as variáveis usadas pelo BASIC: as variáveis de sistema são apenas mneumônicos, ou seja, nomes de referência que tornam mais fácil a identificação das informações relativas ao sistema.

Para alterar uma variável de sistema cujo endereço é *p*, utilize a fórmula: POKE p,v — 256*INT(v/256)

POKE p + 1, INT (v/256), onde

p é o endereço (localização da variável) e *v* é o valor que se deseja introduzir na variável.

Para verificar qual o conteúdo de uma variável de sistema utilize a instrução POKE p,v — 256*INT (v/256)

POKE n + 1, INT(v/256)

E, para verificar qual o conteúdo de uma variável de sistema utilize a instrução PEEK p + 256* PEEK (p+.1).

As abreviações da coluna 1 da lista abaixo têm os seguintes significados:

X não pode fazer um POKE porque será destruído pelo sistema;

N permite fazer um POKE, mas a variável não terá efeito;

S as variáveis são preservadas pelo SAVE.

NOTAS	ENDEREÇO	NOME	CONTEÚDO
1	16384	**ERRO—1**	O código de denotação menos 1. Inicia em 255 (para —1), assim PEEK 16384, se funcionar, fornecerá 255. POKE 16384, p pode ser usado para forçar um erro (HALT): 0≤n≤14 dá uma das denotações comuns; 15≤n≤34 ou 99≤n≤127 dá um código não-padrão; e 35≤n≤98 provavelmente desorganizará o arquivo de imagem.
X1	16385	**FLAGS**	Vários *Flags* para controlar o sistema BASIC.
X2	16386	**RETGSB**	Endereço do primeiro item no Stack da máquina após os retornos de GOSUB.

NOTA	ENDEREÇO	NOME	CONTEÚDO
2	16388	**MEMTOP**	Endereço do primeiro byte na área do sistema BASIC. Você pode fazer um POKE a fim de que o NEW reserve espaço acima daquela área (veja capítulo V) ou fazer com que CLS reserve um arquivo de imagem menor.
N1	16390	**MODO**	Especifica cursor K, L, F ou G
			Linha da instrução que está sendo executada. Um POKE não tem efeito, exceto na última linha do programa.
S1	16393	**VERSÃO**	0 identifica ROM de 8K em programas salvos.
S2	16394	**NUMLI**	Número da linha atual (com o cursor do programa).
SX2	16396	**MAPTELA**	Veja o texto deste capítulo.
S2	16398		Endereço da posição do PRINT no arquivo de imagem. Pode ser feito um POKE de tal maneira que uma saída PRINT seja enviada para outro lugar.
SX2	16400	**VARS**	Veja o texto deste capítulo.
SN2	16402	**DEST**	Endereço da variável em atribuição.
SX2	16404	**LIDIGIT**	Endereço do início da linha que está sendo introduzida.
SX2	16406	**PROX-CAR**	Endereço do próximo caractere a ser interpretado.
S2	16408	**X PTR**	Endereço do caractere precedendo o indicador.
SX2	16410	**STKCOM**	Veja o texto deste capítulo.
SX2	16412	**STK FIM**	Veja o texto deste capítulo.
SN1	16414	**REGIB**	Registrador *b* do processador.
SN2	16415	**MEM**	Endereço usado para cálculos na memória.
S1	16417		Não usado.
SX1	16418	**DF - SZ**	Número de linhas (incluindo uma linha em branco) na parte inferior da tela.
S2	16419	**PROGTOP**	O número da primeira tela do programa, em listagem automática.
SN2	16421	**ULTIMAT**	Última tecla pressionada.
SN1	16423		Estado de *debounce* do teclado.
SN1	16424	**MARGEM**	Número de linhas em branco acima ou abaixo da imagem — 31.
SX2	16425	**PRXLIN**	Endereço da próxima linha de programa a ser executada.

NOTAS	ENDEREÇO	NOME	CONTEÚDO
S2	16427	**LINCONT**	A linha para a qual CONT salta.
SN1	16429	**FLAGX**	Flags variados.
SN2	16430	**COMSTR**	Tamanho da designação do tipo de string em atribuição.
SN2	16432	**ENDSTX**	Endereço do próximo item na tabela de sintaxe.
S2	16434	**SEED**	Valor básico para o RND. Esta é a variável inicializada pelo RAND.
S2	16436	**FRAMES**	Conta os quadros apresentados na televisão. BIT 15 é 1. BITS de 0 a 14 são decrementados para cada quadro enviado à televisão. Isso pode ser usado para temporização, mas PAUSE também a usa. PAUSE reajusta o bit 15 para 0 e coloca nos bits 0 a 14 o tamanho da pausa. Então quando tiverem chegado a zero, a pausa termina. Se a pausa (PAUSE) parar devido a uma tecla ter sido pressionada, o bit 15 é ajustado para 1 novamente.
S1	16438	**COORDX**	Coordenada X do último ponto plotado.
S1	16439	**COORDY**	Coordenada Y do último ponto plotado.
S1	16440	**LPBYTE**	O Byte menos significativo do endereço da próxima posição para LPRINT imprimir.
SX1	16441	**COLUNA**	Número da coluna para posição do PRINT.
SX1	16442	**LINHA**	Número de linha para posição do PRINT.
S1	16443	**FLAGCD**	Flags variados. O bit 7 está ativo (1) durante a modalidade SLOW.
S33	16444	**BUFIMP**	BUFFER da impressora (33º caractere é ENTER).
SN30	16477	**MEMBOT**	Área da memória para cálculo, usada para armazenar os números que não podem ser colocados convenientemente no Stack do processador.
S2	16507		Não usado.

Utilizando Rotinas em Linguagem de Máquina

A linguagem BASIC do CP-200 permite que se utilize, durante um programa normal, uma rotina em linguagem de máquina para um determinado fim. Uma rotina em linguagem de máquina é muito mais veloz que uma em BASIC, pois não necessita ser interpretada pelo programa residente na ROM; ela opera diretamente sobre o microprocessador.

Para se implementar essa rotina é necessário conhecer o conjunto de instruções do Z-80 (o microprocessador usado no CP-200). Sugerimos ao usuário interessado em conhecer o funcionamento do Z-80 a leitura de livros especializados na programação desse microprocessador.

A função que permite a execução dessas rotinas é a USR. O argumento de USR indicará o endereço na memória onde o programa em linguagem de máquina começa. Para o CP-200 sem expansão, o último endereço válido é 32767. O retorno ao programa em BASIC é realizado como o de uma sub-rotina normal. O processamento recomeça da linha seguinte à linha que gerou a chamada USR.

Como se trata de uma função, a USR deve ser feito de uma linha de programa iniciada por uma instrução.
Qualquer instrução pode ser utilizada para gerar uma chamada USR, porém deve-se levar em consideração a sintaxe da instrução e, naturalmente, o bom senso. Eis alguns exemplos de instruções válidas para uma chamada USR.

```
LET X=USR 30000
RAND USR 30000
PRINT USR 17757
```

O primeiro exemplo é o formato adotado neste livro para o acionamento do sinal sonoro do CP-200. A varíavel tem como única finalidade satisfazer sintaticamente a instrução LET, e não deve ser utilizada no restante do programa, pois seu conteúdo dependerá da rotina USR executada.

Um modo de se evitar esse desperdício de memória é utilizar a instrução RAND — o que não afetará a operação da rotina USR. Também a instrução RAND funcionará normalmente.

O terceiro exemplo permite, em alguns casos, além da execução de rotina USR, a impressão do endereço de partida da mesma.

Definindo rotinas em Linguagem de Máquina

Agora que já sabemos como executar as rotinas em linguagem de máquina, precisamos saber como defini-las, ou melhor, como introduzi-las na memória.

A colocação das instruções da rotina USR deve ser feita byte por byte, através da instrução POKE. Podemos ver isso no caso do acionamento do "Bip" do CP-200. Usamos as instruções:

```
POKE 30000,211
POKE 30001,240
POKE 30002,201
```

Estas colocam uma pequena rotina que começa no byte 30000 da memória RAM. Isso feito, cada vez que se quiser um "Bip", usa-se uma instrução do tipo: LET X=USR 30000.

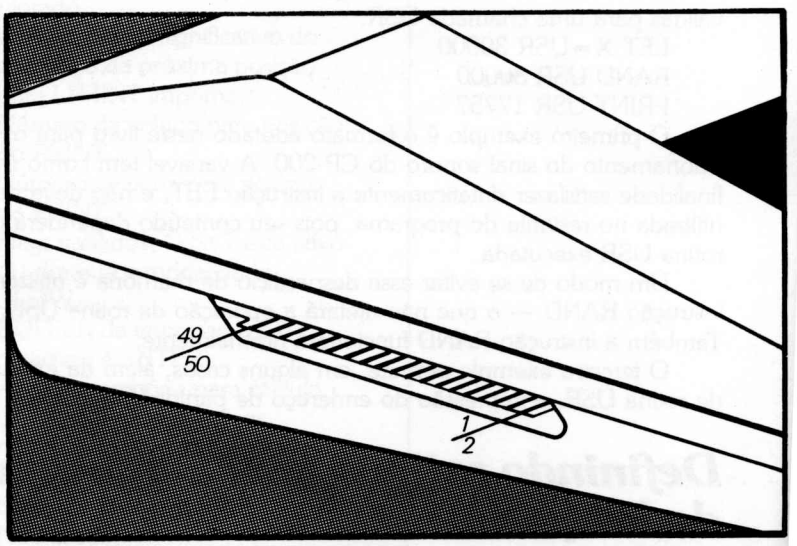
Existem algumas restrições que devem ser observadas quando elabora-se rotinas USR:

1) Evite instruções RET (201 decimal) durante a rotina pois essa instrução finalizará a chamada USR, retornando ao programa em BASIC.
2) Quando retornar, os registradores IY e I devem conter os valores 4000h e IEh (valores hexadecimais), respectivamente.
3) A operação com a tela no CP-200 utiliza os registradores A', F', IX, IY e R do Z-80. As rotinas USR não devem usá-los caso operem com a tela ou mesmo com cálculos matemáticos. Nem mesmo a leitura do par AF' é aconselhável.

Expansão

O CP-200 possui um conector lateral que fornece todos os sinais necessários para se ter acesso direto ao microprocessador. Esse conector permite utilizar o CP-200 como um microprocessador Z-80 comum, e ainda com capacidade de vídeo, gravação e sinalização sonora. A partir daí, pode-se implementar qualquer sistema à base de microprocessador imaginável, tornando ilimitado o campo de aplicação do CP-200.

Abaixo mostramos um esquema do conector, identificando os sinais correspondentes a cada contato:



INFERIOR		SUPERIOR	
N.C.	50	A3	49
N.C.	48	D7	47
N.C.	46	D6	45
N.C.	44	D5	43
N.C.	42	D4	41
N.C.	4Ø	D3	39
$\overline{\text{WAIT}}$	38	D2	37
A14	36	D1	35
$\overline{\text{BUSRQ}}$	34	DØ	33
A8	32	A7	31
A2	3Ø	A15	29
A12	28	AØ	27
A1Ø	26	A1	25
A9	24	A6	23
A11	22	A4	21
$\overline{\text{BUSAK}}$	2Ø	$\overline{\text{RFSH}}$	19
$\overline{\text{M1}}$	18	GND	17
A13	16	GND	15
$\overline{\text{WR}}$	14	GND	13
$\overline{\text{HALT}}$	12	GND	11
RD	1Ø	CLOCK	9
MRQ	8	A5	7
IOR	6	C.S. RAM	5
N.C.	4	N.C.	1
N.C.	2	N.C.	1

Proteção de Rotina USR dentro da Memória

Voltando ao diagrama mostrado no capítulo anterior, que ilustrava a distribuição das informações dentro da memória, você deve notar que existe uma área da memória, acima de MEMTOP, destinada às rotinas USR. Quando ligamos o CP-200, o conteúdo da MEMTOP (uma das variáveis de sistema) equivale ao endereço do último byte de memória disponível no CP-200 que, sem expansão, é 32767 (para 16 kbytes).

Podemos alterar o conteúdo de MEMTOP através de um POKE, abrindo um espaço para nossas rotinas USR. Veja que não abrimos esse espaço para usar a rotina do "bip" neste livro. Por isso, esse procedimento para o "bip" pode gerar problemas quando a memória estiver com um grande volume de dados, acarretando em perda da rotina do "Bip". O endereço 30000 está localizado, inicialmente, no **espaço livre de memória**, que vai sendo utilizado, à medida que se coloca informações no CP-200.

Para abrir um espaço no topo da memória para a rotina do bip, como exemplo, devemos usar o POKE da seguinte forma:

```
POKE 16388,252
POKE 16389,127
```

Isso fará com que o endereço 252 + 256*127 = 32763 seja armazenado em MEMTOP (veja **Variáveis do Sistema,** no capítulo XIII).

Desse modo, temos os quatro últimos bytes disponíveis para nossa rotina USR. Então, podemos armazenar a rotina usando:

```
POKE 32765,211
POKE 32766,240
POKE 32767,201
```

E a execução realizada por LET X=USR 32765

Esse procedimento proteje a rotina USR não apenas dos dados em BASIC, como também de um apagamento quando for executada a instrução NEW. Essa instrução só limpa a memória até a posição indicada por MEMTOP, deixando a rotina USR intacta, para ser usada com novos programas em BASIC.

Fora da área reservada para ela, a rotina USR pode ainda ser colocada em algumas posições da memória relativamente seguras. Como, por exemplo:

— Dentro de uma instrução REM; comece o programa com uma instrução REM seguida de tantos caracteres quantos bytes forem necessários para a rotina USR. Como a área de programa começa em 16509 (Veja no capítulo anterior) e uma linha de programa é precedida de quatro bytes de identificação, a instrução REM ocupará o byte número 16513 e a rotina começará então em 16514. Esse endereço e os seguintes conterão a rotina USR. Coloque a rotina nesses bytes usando POKE como faria para outras posições.

— Em uma variável string; dimensione uma variável string que possa conter a rotina em questão e então coloque com LET um caractere para cada código de máquina. O apêndice C contém uma relação de caracteres e os correspondentes códigos em linguagem de máquina para facilitar essa operação.

Nesses dois locais a rotina estará protegida, pois não será substituída por outra informação quando houver alteração na memória. Porém, o deslocamento das áreas de memória deslocará também a rotina, o que torna pouca confiávell um endereço de partida para a rotina USR.

Concluindo, o local mais seguro para uma rotina em linguagem de máquina é a área própria para ela.onde estará imóvel e protegida mesmo contra um NEW. A principal desvantagem nesse local é que a rotina não será gravada por um SAVE.

Apêndice A
O CP-200 para os que já entendem BASIC

Introdução

Se você já conhece a linguagem BASIC, não deverá encontrar problemas ao utilizar o CP-200. Entretanto, ele possui algumas peculiaridades:

1. As instruções, os comandos e as funções não são introduzidos digitando-se letra por letra (como P-R-I-N-T), mas possuem suas próprias teclas, como você pode verificar consultando o capítulo I.

Neste livro, tais palavras são escritas em negrito.

2. O CP-200 não possui as instruções: **READ, DATA** e **RESTORE** em seu BASIC, nem funções definidas pelo usuário (**FN** e **DEF**); da mesma forma, você não poderá digitar mais de uma instrução por linha.

3. Os recursos para se lidar com **strings** são fáceis de compreender, mas não são padrão — veja o capítulo X.

4. O conjunto de caracteres do CP-200 é exclusivo (Não corresponde ao ASCI.

5. O modo como o CP-200 representa a tela em sua memória é, também, exclusiva.

6. Se você está acostumado a utilizar **PEEK** e **POKE** em outros computadores, lembre-se de que todos os endereços serão diferentes para o CP-200.

Velocidade

O CP-200 trabalha em duas velocidades de processamento **fast** e **slow.** Na velocidade **slow** (lenta), na qual o CP-200 opera normalmente, a imagem de TV é gerada continuamente, enquanto a computação é executada durante os períodos de apagamento, na parte superior e inferior da imagem.

Na velocidade **fast** (rápida), a imagem da TV é desligada durante o processamento e apresentada apenas ao fim de um programa, enquanto aguarda por dados de entrada ou durante uma pausa (veja o capítulo XI).

A modalidade **fast** é cerca de 4 vezes mais rápida que a **slow** e é de grande utilidade, especialmente em programas extremamente longos ou que exijam uma grande quantidade de processamento.

Teclado

Os códigos de caracteres do CP-200 representam não só as letras e dígitos, como também abrangem as várias instruções, comandos e funções — todos eles introduzidos através do teclado. Para tornar isso possível, chegou-se a atribuir até 5 significados distintos a algumas teclas, diferenciados, em parte, pelo uso da tecla SHIFT, como também pela utilização da máquina em diferentes modos de introdução (vide cap. I).

O modo é indicado pelo cursor, uma letra em fundo claro que indica aquilo que será introduzido quando você pressionar a próxima tecla.

O modo K (para comandos) ocorre automaticamente, indicando que o computador aguarda uma instrução ou um n.º de linha de programa. Se não estiver combinada com a tecla SHIFT, a tecla seguinte será interpretada como uma instrução ou como um dígito.

O modo L é acionada imediatamente após você ter introduzido uma instrução indicando que você já pode digitar os caracteres.

O modo F é indicativo de função e ocorre quando você aciona o comando FUNCTION (as teclas **SHIFT** e **ENTER** simultaneamente), para, então, solicitar a função (localizada abaixo da tecla). Tem a duração de apenas um acionamento, retornando em seguida ao modo anterior.

O modo G (para gráficos) ocorre sempre após o acionamento do comando GRAFHICS (SHIFT e tecla 9) e perdura até que seja acionado novamente.

Você poderá ainda ter no cursor a letra S: ela é indicativa de erro de sintaxe e ocorre sempre que o computador não "entende" o que você deseja introduzir.

A Tela

Possui 24 linhas, cada uma comportando até 32 caracteres, e é dividida em 2 partes: a parte superior, compreendendo as 22 primeiras linhas, e a parte inferior, compreendendo as 2 últimas linhas. A parte superior é reservada tanto para a exibição das listagens, como para a saída dos programas, enquanto que a parte inferior é usada para introduzir linhas imediatas, linhas de programa e dados, além de apresentar os códigos de erro.

Entrada via teclado: aparece à esquerda do cursor, na metade inferior da tela, à medida que cada caractere é digitado, seja ele uma letra, um dígito, uma função, um comando ou uma instrução.

NOTAS: 1. Você poderá mover o cursor para a esquerda através da tecla ← (SHIFT e 5) pressionadas simultaneamente. Para movê-lo à direita, faça o mesmo com → (SHIFT e 8).

2. O caractere localizado a esquerda do cursor pode ser removido através de DELETE (SHIFT e 0).

3. Você também poderá apagar toda a linha acionando EDIT (SHIFT e 1), seguida do ENTER.

Sempre que você presiona a tecla ENTER, a linha é executada, introduzida no programa ou empregada como dado de entrada, a menos que contenha um erro de sintaxe, caso em que aparece o cursor S indicando a posição do erro.

À medida em que as linhas de programa são introduzidas, uma listagem vai impressa na parte superior da tela. Você poderá verificar esse tópico com mais detalhes no decorrer do capítulo V.

A última linha a ser introduzida é denominada **linha atual** — sempre indicada pelo símbolo ▒ após o n.º de linha; isso, porém, pode ser mudado utilizando-se as teclas ↓ (SHIFT e 6) e ↑ (SHIFT e 7). Se o comando EDIT for acionado (a linha atual pode ser transferida e editada (alterada) na parte inferior da tela).

Quando uma linha imediata é executada ou um programa é rodado, a saída de dados aparece na parte superior da tela e lá permanece até que uma linha de programa seja digitada, ou até que

ENTER seja acionada sozinha ou ainda até que as teclas ↓ ou ↑ sejam pressionadas. Na parte inferior aparece uma indicação sob a forma *cod/linha*, onde *cod* indica o tipo do erro e a localização que ocorreu durante a execução. (A inexistência de erros será expressa pelo código 0); enquanto que *Linha* corresponde ao número da última linha executada, ou é 0 no caso de comandos. No caso de introduzir-se uma linha imediata, será impresso o dígito 0 na tela. (Veja isto com maiores detalhes no capítulo III).

Tais indicações permanecerão na tela até que uma nova tecla seja acionada, reaparecendo, assim a modalidade K .

Durante a execução de um programa ou de uma linha imediata, a tecla SPACE opera de modo análogo à tecla BREAK, parando o computador com indicação **D**/*linha*.

Isto é reconhecido:

1. Ao final de uma instrução, enquanto o programa está rodando;
2. Quando o computador está procurando por um programa na fita.

O BASIC

Os números são armazenados com uma precisão de 9 ou 10 dígitos.

O maior número que se pode obter no CP-200 gira em torno de 10^{38}, enquanto o menor deles (positivo) é de $4*10^{-39}$.

Os números podem ser armazenados no CP-200 em ponto flutuante, com um byte de expoente (e, onde $1 \leqslant e \leqslant 255$) e quatro bytes de mantissa (m, onde $1/2 \leqslant m < 1$), o que representa o número $m*2^{e-128}$.

Já que $1/2 \leqslant m < 1$, o bit mais significativo da mantissa é sempre 1.

Assim sendo, podemos trocá-lo por um bit para representar o sinal —0 para números positivos, 1 para negativos.

O zero ganhou uma representação especial, nas quais todos os 5 bytes são 0. As variáveis numéricas sempre deverão ser iniciadas por uma letra procedida por combinações arbitrárias de letras e números. Os espaços são ignorados.

Os nomes das variáveis de loops FOR-NEXT podem ter apenas uma única letra.

Isto é válido também para as matrizes numéricas, enquanto que os nomes referentes às matrizes string serão compostos por uma única letra, seguida de $

Tanto as matrizes numéricas quanto as matrizes string exibirão tantos itens quantos puderem ser comportados na memória.

O número de itens e o tamanho dos mesmos serão especificados através da instrução DIM. Seus índices começam em 1.

Seccionamento

Poderemos isolar trechos de strings utilizando seccionadores.

Um seccionador pode ser:

1. vazio
2. uma expressão numérica
3. duas expressões numéricas opcionais ligadas por TO

Sub-string podem ser seccionadas de duas formas diferentes:

a) expressão string (Seccionador)

Na relação abaixo,

L	representa qualquer letra;
V	representa uma variável;
x,y,z	representam expressões numéricas;
m.n	representam expressões numéricas que são arredondadas para o número inteiro mais próximo;
e	representar uma expressão;
f	representa uma expressão com valor literal (string);
s	representa uma instrução.

Instruções

Note que expressões arbitrárias são permitidas em qualquer lugar, exceto para número de linha no início da instrução.

Todas as instruções, exceto INPUT, podem ser usadas tanto em linhas imediatas como em linhas de comando.

CLEAR — Cancela todas as variáveis, liberando o espaço ocupado por elas.

CONT — Continua a execução após um BREAK ou STOP.

CLS — Limpeza da tela.

COPY — Envia uma cópia da tela para a impressora, se estiver acoplada a esta. Denotação D, se BREAK for pressionado.

DIM *L (nl,..nk)* — Cancela qualquer matriz com o nome *L* e define uma matriz de números com *K* dimensões *nl,..nk*. Zera todos os elementos.
Erro 4 ocorre se não há espaço na memória para a matriz. Uma matriz é indefinida até ser dimensionada em uma instrução DIM.

DIM *L$ (nl,..nk)* — *Cancela qualquer matriz ou string com nome L$ e define uma matriz de caracteres com k* dimensões *nl,..nk*. Preenche de espaços todos os elementos. Pode ser considerada uma matriz de strings de tamanho fixo *nk*, com *k−1* dimensões *nl,...,nk−1*.
Erro 4 ocorre se não há espaço na memória para a matriz. Uma matriz é indefinida até que seja dimensionada por uma instrução DIM.

FOR L = x TO *y* STEP *z* — Cancela qualquer variável *L* e define uma variável de controle com valor *x*, limite *y*, passo *z*, e o endereço de *loop* 1 mais o número da linha da instrução FOR (−1 se for um comando).
Verifica se o valor inicial é maior (se passo ⩾ 0) ou menor (se passo < 0) que o limite, e em caso afirmativo, volta para a instrução NEXT *L*, no início da linha. Veja NEXT *L*.
Erro 4 ocorre se não houver espaço na memória para a variável de controle.

GOSUB *n* — Passa a sub-rotina que começa pela linha de número especificado.
Erro 4 pode ocorrer se não houver RETURNs suficientes.

GOTO *n* — Salta para linha *n* (ou, se não houver, para primeira após a mesma).

b) matriz string (índice, ..., índice, Seccionador)

A forma "b" possui ainda a seguinte variante:

matriz string (índice,..., índice) (Seccionador)

Na forma (a), suponha que a expressão de *string* corresponda a S$; se o seccionador estiver vazio (item 1), o resultado será considerado como sendo a própria *string* (S$).

Se o seccionador for uma expressão numérica (item 2) definida por *n*, o resultado corresponderá ao caractere definido pela posição *n* (*sub-string* com apenas um caractere).

Supondo que o seccionador seja composto de duas expressões (item 3) e que a primeira expressão numérica tenha o valor *m* e a segunda assuma o valor *n*. Se $1 \leqslant m \leqslant n \leqslant$ tamanho de S$, o resultado será uma *sub-string* formada pelos caracteres contidos entre as posições definidas por *m* e *n*.

Exemplificando:

Na matriz S$ = "Ipiranga", se m = 3 e n = 5, teremos:

```
IPIRANGA
  ▲ ▲
  3 5
S$ (3 TO 5) = "IRA"
```

Você poderá compreender melhor as operações *string*, lendo o capítulo X.

Operações

Os seguintes símbolos são que envolvem dois operandos:

+ Adição (em números) ou concatenação (em strings);
— Subtração;
* Multiplicação;
/ Divisão;
** Elevação a uma potência. Erro B se o operando esquerdo for negativo;

= Igual
> Maior que;
< Menor que;
>= Menor ou Igual a;
>= Maior ou Igual a;
<> Diferente de;

} Ambos os operandos devem ser do mesmo tipo (numéricos ou string). O resultado é sempre numérico: 1 se a comparação for verdade e 0 se a comparação for falsa.

Tabela de prioridades

Funções e Operações têm as seguintes prioridades:

Operações	Prioridade
Indexação de Matrizes e Seccionamento	12
Todas as funções exceto NOT e sinal de menos	11
**	10
Sinal	9
*,/	8
+,— (subtração)	6
=,>,<,<=,<>,>	5
NOT	4
AND	3
OR	2

IF *x* THEN *s*
Se *x* for verdadeiro (> zero), então S é executado.
O formato "IF *x* THEN *número da linha*" não é permitido.

INPUT *V*
Pára e espera que seja feita uma entrada de dados.
O arquivo de imagem é impresso no modo FAST. INPUT não pode ser usado como comando, pois ocorrerá erro 8.
Se o primeiro caractere em uma linha de INPUT é STOP, o programa pára com denotação D.

LET *V*=*e*
Atribui o valor *e* à variável *v*. LET não pode ser omitido. Uma variável simples é indefinida, até figurar em uma instrução LET ou INPUT. Se *v* é uma string, então o valor *e* do string é truncado ou preenchido com zeros à direita, para ter o mesmo comprimento que a variável *V*.

LIST
Lista o programa iniciando na 1.ª linha.

LIST *n*
Lista o programa na televisão, iniciando na linha *n*, e faz de *n* a linha atual. Erros 4 ou 5, se a listagem for muito longa para a tela. CONT fará exatamente o mesmo, novamente.

LLIST
Imprime o programa iniciando na 1.ª linha.

LLIST *n*
Como LIST, porém usando a impressora e não a TV.
Não faz nada se não houver impressora; pára com denotação D, se BREAK for pressionada.

LOAD "*f*"
Procura o programa chamado *f* na fita e carrega-o com suas variáveis. Se f = "", então carrega o primeiro programa encontrado.
Se BREAK for pressionado, então (1) se nenhum programa for lido na fita, pára com denotação D e o programa antigo; (2) se uma parte do programa foi lida, então executa NEW.

LPRINT...
Como PRINT, mas usando a impressora. Uma linha de texto é enviada à impressora;
(1) Quando a impressão salta de uma linha para a próxima;
(2) Após LPRINT que não termina em uma vírgula ou um ponto e vírgula.
(3) Quando uma vírgula ou item TAB requer uma nova linha.
(4) No fim do programa, se há algo que não foi impresso. Num item AT, apenas o número da coluna tem efeito; o número da linha é ignorado. Um item AT nunca manda uma linha de texto para a impressora.
Não há efeito algum se não houver a impressora. Pára com denotação D se for pressionado BREAK.

NEW
Inicia o sistema BASIC, cancelando o programa e as variáveis e limpando a memória até (mas não incluindo) o byte cujo endereço está na variável MEMTOP (bytes 16388 e 16389).

NEXT *L*
(1) Encontra a variável de controle *L*.
(2) Retornará à linha que contém FOR... se o valor de *L* estiver entre os valores: inicial (*x*) e limite (*y*):

Erro 1 se não há variável de controle L.

PAUSE *n* — Pára o processamento e exibe a imagem durante *n* quadros (a 60 quadros por segundo) ou até uma tecla ser pressionada. Caso o valor não esteja na faixa $0 \leqslant n \leqslant 65535$, ocorrerá um erro B. erro B.
Se $n > 32767$, então não há limite para a pausa, mas dura até uma tecla ser pressionada.

PLOT *m,n* — Ativa o ponto gráfico na tela definido pelos valores absolutos de *m* e *n*. Move a posição de impressão para após o referido ponto. Deve-se obedecer as seguintes condições $0 \leqslant m \leqslant 63$ e $0 \leqslant n \leqslant 43$; caso contrário, erro B.

POKE *m,n* — Escreve o valor *n* do byte de endereço *m*. $0 \leqslant m \leqslant 65535, -255 \leqslant n \leqslant 255$, ou então em B.

PRINT... — O "..." é uma seqüência de ítens PRINT, separados por vírgula ou ponto e vírgula. Eles estão escritos no arquivo de imagem na televisão. A posição (linha e coluna) onde o próximo caracter deve ser impresso é chamado de posição de impressão (PRINT).
Um item PRINT pode ser:
(1) Vazio, isto é, nada;
(2) Uma expressão numérica.
Primeiro, um sinal menos é impresso se o valor é negativo. Em seguida, atribui a X o módulo do valor.
Se $x \leqslant 10^{-5}$ ou $x \geqslant 10^{13}$, então a impressão usa notação científica. A mantissa tem até 8 dígitos e o ponto decimal (ausente em caso de 1 dígito) está após o primeiro. O expoente é E, seguido de + ou —, seguido de mais 2 dígitos. Em caso contrário, X é impresso em notação decimal ordinária de até 8 dígitos significativos e sem zeros após o ponto decimal. Um ponto decimal no início é sempre seguido de um zero.
0 é impresso com, um único dígito 0
(3) Uma expressão string
Os caracteres são impressos segundo o código de caracteres do CP-200. Alguns ocupam mais de um dígito na tela (PRINT, INPUT, etc.). O caracter de aspas é impresso como ". Caracteres não usados e caracteres de controle são impresos como "?"
(4) AT m,n
A posição de impressão é mudada para linha m (contando do topo), coluna n (contando da esquerda).
Se $|m| = 22$ ou 23, erro 5
(5) TAB n
n é reduzido para módulo 32. Então, a posição de impressão é movida para coluna n, permanecendo na mesma linha, a menos que isso envolva retorno na mesma linha; neste caso, move-se para próxima linha.
$0 \leqslant n \leqslant 255$, ou então erro B.
Um ponto e vírgula entre 2 itens imobiliza a.

posição de impressão de forma que o 2º item siga logo após o primeiro.
Uma vírgula, por outro lado, move a posição de impressão de no mínimo um lugar; e, após isso, tantos quanto forem necessários para deixá-la na coluna 0 ou 16, introduzindo uma nova linha, se necessário.
Ao fim de uma instrução PRINT, se não terminar com um ponto e vírgula ou vírgula, uma nova linha será colocada.
Erro 4 (fora da memória) pode ocorrer com 3k ou menos memória.
Erro 5 significa que a tela está repleta.
Em ambos os casos, a solução é CONT, que limpará a tela e continuará.

RAND — RAND 0

RAND *n* — Atribui valor à variável do sistema (chamado Seed), usada para gerar o próximo valor de RND. Se $n \neq 0$, o Seed recebe o valor *n*; se $n = 10$ lhe é dado um valor de uma outra variável do sistema (chamada FRAMES) que conta os quadros impressos até então na televisão, o que deve ser bastante randômico.
Erro B ocorre se *n* não estiver na faixa de 0 a 65535.

REM... — Indicador de mensagem. Identifica comentários em um programa. (Abrev. REMark); ignora o restante da linha.

RETURN — Retira o número da linha do stack de GOSUB e salta para a linha seguinte. Erro 7 ocorre quando não há: número de linha no stack, ou seja, GOSUB não deverá estar propriamente balanceado pelos RETURNS.

RUN — RUN 0

RUN n — CLEAR, e, então, GOTO *n*

SAVE f — Grava o programa e variáveis na fita e denomina-os de *f*.
SAVE não deve ser usado em uma rotina GOSUB.
Erro F ocorre se *f* for uma string vazia, o que não é permitido.

SCROLL — Gira o arquivo de imagem uma linha para cima perdendo a linha do topo e criando outra embaixo.

STOP — Finaliza o programa com denotação 9, CONT reiniciará na próxima linha.

UNPLOT m,n — Como PLOT, porém "branqueia" o elemento de impressão.

AND — Operação Binária em que o operando deverá ser sempre um número.
Se o operando da esquerda for numérico:

$$A \text{ AND } B = \begin{cases} A \text{ se } B \neq 0 \\ 0 \text{ se } B = 0 \end{cases}$$

Se o operando da esquerda for uma string.

$$A\$ \text{AND } B = \begin{cases} A\$ \text{ se } B \neq 0 \\ \text{“”} \text{ se } B = 0 \end{cases}$$

ASN — Número — Arco-seno em radianos
erro A se x não estiver entre −1 e 1.

ATN	Número	Arco-tangente em radianos
CHR$	Número	O caractere cujo código é x, arredondado para o número inteiro mais próximo. Erro B se x não estiver na faixa de 0 a 255.
CODE	String	O código do primeiro caractere em x (ou 0, se x for string vazio).
COS	Número (em radianos)	Co-seno
EXP	Número	e^x
INKEY$	Sem operando	Lê o teclado. O resultado é o caractere representando (em modo L) a tecla pressionada. Será uma string vazia se nenhuma tecla for pressionada.
INT	Número	Parte inteira (sempre arredondada para menos)
LEN	String	Comprimento da string
LN	Número	Logaritmo natural (na base e). Erro A se $x \leq 0$; 0 se $x \neq 0$; 1 se $x = 0$
		NOT tem prioridade 4.
OR	Operação binária Ambos os operandos devem ser números	A OR B = $\begin{cases} 1 \text{ se } B \neq 0 \\ A \text{ se } B = 0 \end{cases}$
		OR tem prioridade 2.
PEEK	Número	O valor do byte da memória cujo endereço é x (arredondado para o inteiro mais próximo). Erro B se x não estiver na faixa de 0 a 65535.
PI	Sem operando	π (3,14159265...)
RND	Sem operando	Gera uma seqüência pseudo aleatória de números entre 0 e 1.
SGN	Número	O sinal (0,1 ou —1) de x
SIN	Número (em radianos)	seno
SQR	Número	Raiz quadrada; erro B se $x < 0$.
STR$	Número	Transforma um número em um string que o represente literalmente.
TAN	Número (em radianos)	Tangente
USR	Número	Chama a sub-rotina em linguagem de máquina cuja primeiro endereço é x. Ao retornar, o resultado é o conteúdo do par de registradores bc. Calcula x como se fosse uma expressão numérica. Erro C se x contiver um erro de

		sintaxe, ou fornecer o valor do string. Outros erros são possíveis, dependendo do string.
—	Número	negação

Apêndice B — Código de Erros

Esta tabela retrata todos os códigos de Erros do CP-200 descrevendo-os e apresentando uma lista de situações (instruções e funções) nós quais possam ocorrer.

No Apêndice A, abaixo de cada função ou instrução, você poderá encontrar uma descrição mais detalhada do que significa a indicação do erro.

CÓDIGO	SIGNIFICADO	SITUAÇÕES
0	Execução bem sucedida ou salto para uma linha de maior número que qualquer outra existente. Uma indicação 0 não altera o número da linha usado por CONT.	Qualquer
1	A variável de controle não existe (não foi estabelecida por uma instrução FOR), mas existe uma outra variável com o mesmo nome.	NEXT
2	Foi utilizada uma variável indefinida. No caso de uma variável simples, isso ocorrerá se a mesma for usada antes de ter sido referenciada por uma instrução LET. Para matrizes isso acontecerá se as mesmas forem usadas antes de ser dimensionadas pela 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
3	A matriz não contém o elemento especificado. Caso o índice esteja fora de faixa (ou seja, negativo ou acima de 65535), ocorrerá o erro B.	Matrizes
4	Espaço insuficiente na memória. Observe que o número de linha na indicação (após o /) poderá estar incompleto na tela, devido à falta de memória; assim, por exemplo, 4/20 poderá surgir como 4/2.	LET, INPUT, DIM, PRINT, LIST, PLOT, UNPLOT, FOR, GOSUB; Às vezes, durante avaliação de funções.
5	Não há mais espaço na tela. CONT criará espaço, limpando a tela.	PRINT, LIST, PLOT UNPLOT.

CÓDIGO	SIGNIFICADO	SITUAÇÕES
6	Resultado fora dos limites de armazenamento: os cálculos produziram um número superior a 10^{38}.	Qualquer cálculo aritmético.
7	RETURN sem um GOSUB correspondente	RETURN
8	Você tentou um comando INPUT não permitido.	INPUT
9	Instrução STOP executada. CONT fará a execução começar da próxima linha.	STOP
A	Argumento inválido para certas funções.	SQR,LN,ASN,ACS.
B	O CP-200 só aceita números inteiros compreendidos entre o 65535. Quando um número inteiro faz-se necessário, o argumento é arredondado para o número inteiro mais próximo. Caso esse arredondamento esteja fora da faixa, surge erro B. Para acesso à matriz, veja também indicador 3.	RUN, RAND, POKE, DIM, GOTO, GOSUB, LIST, PAUSE, PLOT, UNPLOT, CHR$, PEEK, USR. Quando de um acesso a uma matriz.
C	O texto do argumento (de string) de VAL não forma uma expressão numérica válida.	VAL
D	1. Programa interrompido por BREAK. 2. A linha INPUT começa com STOP	ao fim de qualquer instrução ou em LOAD, SAVE, LPRINT, LLIST ou COPY.
E	Não utilizado	
F	O nome fornecido para o programa é uma string vazia.	

Apêndice C
Código de caracteres

Este é o conjunto completo de caracteres do CP-200, com códigos em decimal e hexadecimal.

Imaginando os códigos sob a forma de linguagem de máquina Z80, vamos encontrar os mneumônicos correspondentes nas três colunas à direita. Como você deve saber, algumas instruções do Z80 são compostas, começando com CBh ou EDh, como se pode verificar nas duas colunas da direita.

CARACTERE	CÓDIGO		ASSEMBLER Z80		
	DEC	HEX	MNEUMONICO	Após CB	Após ED
espaço	0	00	NOP	RLC B	
▘	1	01	LB BC, nn	RLC C	
▝	2	02	LD (BC),A	RLC D	
▀	3	03	INC BC	RLC E	
▖	4	04	INC B	RLC B	
▌	5	05	DEC	RLC L	
▚	6	06	LD B,n	RLC (HL)	
▛	7	07	RLCA	RLC A	
▒	8	08	EX AF,AF'	RRC B	
▒	9	09	ADD HL,BC	RRC C	
▒	10	0A	LD A,(BC)	RRC D	
"	11	0B	DEC BC	RRC E	
£	12	0C	INC C	RRC H	
$	13	0D	DEC C	RRC L	
:	14	0E	LD C,n	RRC (HL)	
?	15	0F	RRCA	RRC A	
(	16	10	DJNZ dis	RL B	
)	17	11	LD DE,nn	RL C	
>	18	12	LD (DE),A	RL D	
<	19	13	INC DE	RL E	
=	20	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 DE	RR E	
0	28	1C	INC E	RR H	
1	29	1D	DEC E	RR L	
2	30	1E	LD E,n	RR (HL)	
3	31	1F	RRA	RR A	
4	32	20	JR NZ,dis	SLA B	
5	33	21	LD HL,nn	SLA C	
6	34	22	LD (nn),HL	SLA D	
7	35	23	INC HL	SLA E	
8	36	24	INC H	SLA H	
9	37	25	DEC H	SLA L	

CARACTERE	CÓDIGO		ASSEMBLER Z80		
	DEC	HEX	MNEUMONICO	Após CB	$Após ED
A	38	26	LD H,n	SLA (HL)	
B	39	27	DAA	SLA A	
C	40	28	JR Z,dis	SRA B	
D	41	29	ADD HL,HL	SRA C	
E	42	2A	LD HL,(nn)	SRA D	
F	43	2B	DEC HL	SRA E	SRA E
G	44	2C	INC L	SRA H	
H	45	2D	DEC L	SRA L	
I	46	2E	LD L,n	SRA (HL)	
J	47	2F	CPL	SRA A	
K	48	30	JR NC,dis		
L	49	31	LD SP, nn		
M	50	32	LD (nn),A		
N	51	33	INC SP		
O	52	34	INC (HL)		
P	53	35	DEC (HL)		
Q	54	36	LD (HL),n		
R	55	37	SCF		
S	56	38	JR C,dis	SRL B	
T	57	39	ADD HL,SP	SRL C	
U	58	3A	ID A,(nn)	SRL D	
V	59	3B	DEC SP	SRL E	
W	60	3C	INC A	SRH H	
X	61	3D	DEC A	SRL L	
Y	62	3E	LD A,n	SRL (HL)	
Z	63	3F	CCF	SRL A	
RND	64	40	LD B,B	BIT 0,B	IN B,(C)
INKEY$	65	41	LD B,C	BIT 0,C	OUT (C),B
PI	66	42	LD B,D	BIT 0,D	SBC HL,BC
	67	43	LD B,E	BIT 0,E	LD (nn),BC
	68	44	LD B,H	BIT 0,H	NEG
	69	45	LD B,L	BIT 0,1	RETN
	70	46	LD B,(HL)	BIT 0,(HL)	IM 0
	71	47	LD B,A	BIT 0,A	LD I,A
	72	48	LD C,D	BIT 1,B	IN C,(C)
	73	49	LD C,C	BIT 1,C	OUT (C),C
	74	4A	LD C,D	BIT 1,D	ADC HL,BC
	75	4B	LD C,E	BIT 1,E	LD BC,(nn)
	76	4C	LD C,H	BIT 1,H	
	77	4D	LD C,L	BIT 1,I	RETI
	78	4E	LD C,(HL)	BIT 1,(HL)	
não usado	79	4F	LD C,A	BIT 1,A	LD R,A
	80	50	LD D,B	BIT 2,B	IN D,(C)
	81	51	LD D,C	BIT 2,C	OUT (C),D
	82	52	LD D,D	BIT 2,D	SBC HL,DE
	83	53	LD D,E	BIT 2,E	LD (nn),DE
	84	54	LD D,H	BIT 2,H	
	85	55	LD D,I	BIT 2,L	
	86	56	LD D,(HL)	BIT 2,(HL)	IM 1
	87	57	LD D,A	BIT 2,A	ID A,I
	88	58	LD E,B	BIT 3,B	IN E,(C)
	89	59	LD E,C	BIT 3,C	OUT (C),E
	90	5A	LD E,D	BIT 3,D	ADC HL,DE
	91	5B	LD E,E	BIT 3,E	LD DE,(nn)

CARACTERE	CÓDIGO		ASSEMBLER Z80		
	DEC	HEX	MNEUMONICO	Após CB	$Após ED
	92	5C	LD E,H	BIT 3,H	
	93	5D	LD E,I	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	60	LD H,B	BIT 4,B	IN H,(C)
	97	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
	100	64	LD H,H	BIT 4,H	
não usado	101	65	LD H,I	BIT 4,L	
	102	66	LD H,H(HL)	BIT 4,(HL)	
	103	67	LD H,A	BIT 4,A	RRD
	104	68	LD L,B	BIT 5,B	IN L,(C)
	105	69	LD L,C	BIT 5,C	OUT (C),L
	106	6A	LD L,D	BIT 5,D	ADC HL,HL
	107	6B	LD L,E	BIT 5,E	LD DE,(nn)
	108	6C	LD L,H	BIT 5,H	
	109	6D	LD L,L	BIT 5,L	
	110	6E	LD L,(HL)	BIT 5,(HL)	
	111	6F	LD L,A	BIT 5,A	RLD
cursor para cima	112	70	LD (HL),B	BIT 6,B	
cursor para baixo	113	71	LD (HL),C	BIT 6,C	
cursor para esquerda	114	72	LD (HL),D	BIT 6,D	SBC HL,SP
cursor para direita	115	73	LD (HL),E	BIT 6,E	LD (nn),SP
GRAPHICS	116	74	LD (LH),H	BIT 6,H	
EDIT	117	75	LD (HL),L	BIT 6,L	
ENTER	118	76	HALT	BIT 6,(HL)	
DELETE	119	77	LD (HL),A	BIT 6,A	
K / L modo	120	78	LD A,B	BIT 7,B	IN A,(C)
FUNCTION	121	79	LD A,C	BIT 7,C	OUT (C),A
não usado	122	7A	LD A,D	BIT 7,D	ADC HL,SP
não usado	123	7B	LD A,E	BIT 7,E	LD SP,(nn)
não usado	124	7C	LD A,H	BIT 7,H	
não usado	125	7D	LD A,L	BIT 7,I	
número cursor	126	7E	LD A,(HL)	BIT 7,(HL)	
	127	7F	LD A,A	BIT 7,A	
■	128	80	ADD A,B	RES 0,B	
▟	129	81	ADD A,C	RES 0,C	
▙	130	82	ADD A,D	RES 0,D	
▬	131	83	ADD A,E	RES 0,E	
▛	132	84	ADD A,H	RES 0,H	
▌	133	85	ADD A,L	RES 0,I	
▚	134	86	ADD A,(HL)	RES 0(HL)	
▜	135	87	ADD A,A	RES 0,A	
▒	136	88	ADD A,B	RES 1,B	
▒	137	89	ADD A,C	RES 1,C	
▒	138	8A	ADD A,D	RES 1,D	
"	139	8B	ADD A,E	RES 1,E	
£	140	8C	ADD A,H	RES 1,H	
$	141	8D	ADD A,L	RES 1,L	

CARACTERE	CÓDIGO		ASSEMBLER 280		
	DEC	HEX	MNEUMONICO	Após CB	$Após ED
:	142	8E	ADC A,(HL)	RES 1,(HL)	
?	143	8F	ADC A,A	RES 1,A	
(	144	90	SUB B	RES 2,B	
)	145	91	SUB C	RES 2,C	
>	146	92	SUB D	RES 2,D	
<	147	93	SUB E	RES 2,E	
=	148	94	SUB H	RES 2,H	
+	149	95	SUB L	RES 2,L	
-	150	96	SUB (HL)	RES 2,(HL)	
*	151	97	SUB A	RES 2,A	
/	152	98	SBC A,B	RES 3,B	
;	153	99	SBC A,C	RES 3,C	
'	154	9A	SBC A,D	RES 3,D	
.	155	9B	SBC A,E	RES 3,E	
0	156	9C	SBC A,H	RES 3,H	
1	157	9D	SBC A,L	RES 3,L	
2	158	9E	SBC A,(HL)	RES 3,(HL)	
3	159	9F	SBC A,A	RES 3,A	
4	160	A0	AND B	RES 4,B	LDI
5	161	A1	AND C	RES 4,C	CPI
6	162	A2	AND D	RES 4,C	INI
7	163	A3	AND E	RES 4,D	OUTI
8	164	A4	AND H	RES 4,H	
9	165	A5	AND L	RES 4,L	
A	166	A6	AND (HL)	RES 4,(HL)	
B	167	A7	AND A	RES 4,A	
C	168	A8	XOR B	RES 5,B	IDD
D	169	A9	XOR C	RES 5,C	CPD
E	170	AA	XOR D	RES 5,D	IND
F	171	AB	XOR E	RES 5,E	OUTD
G	172	AC	XOR H	RES 5,H	
H	173	AD	XOR L	RES 5,L	
I	174	AE	XOR (HL)	RES 5,(HL)	
J	175	AF	XOR A	RES 5,A	
K	176	B0	OR B	RES 6,B	LDIR
L	177	B1	OR C	RES 6,C	CPIR
M	178	B2	OR D	RES 6,D	INIR
N	179	B3	OR E	RES 6,E	OTIR
O	180	B4	OR H	RES 6,H	
P	181	B5	OR L	RES 6,L	
Q	182	B6	OR (HL)	RES 6,(HL)	
R	183	B7	OR A	RES 6,A	
S	184	B8	CP B	RES 7,B	LDDR
T	185	B9	CP C	RES 7,C	CPDR
U	186	BA	CP D	RES 7,D	INDR
V	187	BB	CP E	RES 7,E	ORDR
W	188	BC	CP H	RES 7,H	
X	189	BD	CP L	RES 7,L	
Y	190	BE	CP (HL)	RES 7,(HL)	
Z	191	BF	CP A	RES 7,A	
""	192	C0	RET NZ	SET 0,B	
AT	193	C1	POR BC	SET 0,C	
TAB	194	C2	JP NZ,nn	SET 0,D	
não usado	195	C3	JP nn	SET 0,E	

CARACTERE	CÓDIGO		ASSEMBLER 280		
	DEC	HEX	MNEUMONICO	Após CB	$Após ED
CODE	196	C4	CALL NZ,nn	SET 0,H	
VAL	197	C5	PUSH BC	SET 0,L	
LEN	198	C6	ADD A,n	SET 0,(HL)	
SIN	199	C7	RST 0	SET 0,A	
COS	200	C8	RET Z	SET 1,B	
TAN	201	C9	RET	SET 1,C	
ASN	202	CA	JP Z,nn	SE 1,D	
ACS	203	CB		SET 1,E	
ATN	204	CC	CALL Z,nn	SET 1,H	
LN	205	CD	CAL nn	SET 1,L	
EXP	206	CE	ADC A,n	SET 1,(HL)	
INT	207	CF	RST 8	SET 1,A	
SQR	208	D0	RET NC	SET 2,B	
SGN	209	D1	POP DE	SET 2,C	
ABS	210	D2	JP NC,nn	SET 2,D	
PEEK	211	D3	OUT n,A	SET 2,E	
USR	212	D4	CALL NC,nn	SET 2,H	
STR$	213	D5	PUSH DE	SET 2,1	
CHR$	214	D6	SUB n	SET 2,(HL)	
NOT	215	D7	RST 16	SET 2,A	
**	216	D8	RET C	SET 3,B	
OR	217	D9	EXX	SET 3,C	
AND	218	DA	JP C,nn	SET 3,D	
<=	219	DB	IN A,n	SET 3,C	
>=	220	DC	CALL C,nn	SET 3,H	
<>	221	DD	prefixo de instruções usando IX	SET 3,1	
THEN	222	DE	SBC A,n	SET 3,1	
TO	223	DF	RST 24	SET 3,A	
STEP	224	E0	RET PO	SET 4,B	
LPRINT	225	E1	POP HL	SET 4,C	
LLIST	226	E2	JP PO,nn	SET 4,D	
STOP	227	E3	EX (SP),HL	SET 4,E	
SLOW	228	E4	CALL PO,nn	SET 4,H	
FAST	229	E5	PUSH HL	SET 4,L	
NEW	230	E6	AND n	SET 4,(HL)	
SCROLL	231	E7	RST 32	SET 4,A	
CONT	232	E8	RET PE	SET 5,B	
DIM	233	E9	JP (HL)	SET 5,C	
REM	234	EA	JP PE,nn	SET 5,D	
FOR	235	EB	EX DE,HL	SET 5,E	
GOTO	236	EC	CALL PE,nn	SET 5,H	
GOSUB	237	ED		SET 5,L	
INPUT	238	EE	XOR n	SET 5,(HL)	
LOAD	239	EF	RST 40	SET 5,A	
LIST	240	F0	RET P	SET 6,B	
LET	241	F1	POP AF	SET 6,C	
PAUSE	242	F2	JP P,nn	SET 6,D	
NEXT	243	F3	DI	SET 6,E	
POKE	244	F4	CALL P,nn	SET 6,H	
PRINT	245	F5	PUSH AF	SET 6,1	
PLOT	246	F6	OR n	SET 6,(HL)	

CARACTERE	CÓDIGO		ASSEMBLER Z80		
	DEC	HEX	MNEUMONICO	Após CB	Após ED
RUN	247	F7	RST 48	SET 6,A	
SAVE	248	F8	RET M	SET 7,B	
RAND	249	F9	LD SP, HL	SET 7,C	
IF	250	FA	JP M,nn	SET 7,D	
CLS	251	FB	EI	SET 7,E	
UNPLOT	252	FC	CALL M,nn	set 7,h	
CLEAR	253	FD	prefixo de instruções usando IY		
			USANDO IY	SET 7,1	
RETURN	254	FE	CP n	SET (HL)	
COPY	255	FF	RST 56	SET 7,A	

Glossário

A

ACESSO — Maneira pela qual um conjunto de dados é introduzido na memória do computador.
ALEATORIZAR — Dispor dados de forma aleatória, ou seja, sem que obedeçam a uma seqüência ou relação pré-determinada.
ALFANUMÉRICO — Qualquer um dos caracteres usados em computação, tais como: letras, números ou sinais de pontuação.
ALIMENTAR — Suprir o computador com os dados a fim de que sejam processados.
ARQUIVO — Conjuntos de dados selecionados e reunidos necessários para desenvolver um certo processo. Acumulação de informações na memória do computador.
ARRAY — Conjunto de dados relacionados a uma variável e que podem ser identificados separadamente através de um índice. O mesmo que Matriz Unidimensional.
ASCII — Abreviatura de *American Standard Code for Information Interchange* (Código Padrão Americano para Intercâmbio de Informação). A cada caractere do computador corresponde um código — um sistema de símbolos com que se representa este caractere. Este código é utilizado com o objetivo de facilitar o intercâmbio destas informações erttre os vários sistemas de processamento.
NOTA: No Apêndice C encontra-se o conjunto completo de caracteres do CP-200 com seus respectivos códigos em decimal e hexadecimal.

B

BASIC — Abreviatura de *Beginner's All-Purpose Symbolic Instruction Code* (Código Simbólico de Instrução, Universal, para Principiantes). Linguagem de alto nível em programação, dirigida a principiantes.
BINÁRIO — Sistema numérico que emprega a base 2, ao invés da base 10 (decimal), dispondo dos dígitos 0 e 1 para formar números.
BIT — Abreviatura de BInary digiT (Dígito Binário) Unidade de informação tem o mesmo efeito.
BREAK — Interrupção na execução de um programa. Em BASIC a instrução STOP tem o mesmo efeito.
BYTE — A menor unidade endereçável na memória do computador, constituída de 8 bits e capaz de representar 256 valores diferentes.

C

CARACTERE — Elemento de um conjunto de símbolos utilizados na representação gráfica de dados. Os símbolos mais usados são os algarismos de 0 até 9, as letras, de A até Z, ou outros símbolos que o computador pode processar.

COMANDOS — Um sinal ou mais sinais eletrônicos emitidos para iniciar ou finalizar um certo processo. Pode ser, também, uma parte de uma instrução que determina qual a próxima operação a ser desenvolvida.
COMPUTADOR — Aparelho eletrônico capaz de receber informações, submetê-las a um conjunto especificado e predeterminado de operações lógicas e matemáticas, e fornecer o resultado dessas operações. Consiste, normalmente, de dispositivos de entrada e saída (E/S), unidades de armazenagem, de cálculos aritméticos e lógicos, além de uma unidade de controle (CPU).

D

DADOS — São Letras, números ou símbolos que podem ser codificados, memorizados e processados a fim de obter-se a solução de um determinado problema. Notar que o resultado do problema também se constitui num dado.
DEBUG — Processo de localização e correção de quaisquer erros num programa de computador. Teste de programa, a fim de assegurar de que funciona corretamente.
DELETE — Remover, eliminar dados, como, por exemplo cancelar dados de um registro de um arquivo.
DISPOSITIVO — É uma parte física do sistema do computador usada para a entrada e saída de dados. Ex.: teclado, impressora e tela.

E

EDIT/EDITAR — Processo de reajustar e corrigir dados ou de aperfeiçoar a forma apresentada nas informações de uma linha de programa.
ENDEREÇO — Uma posição da memória onde o dado está armazenado. Normalmente é representada em decimal ou hexadecimal (base 16).
ENTRADA — Transferência de dados de fora para dentro do computador. Esta passagem de informações é possível através de fita cassete, teclado, etc.
ENTRADA/SAÍDA (E/S) — Termo genérico que designa os dados que entram ou saem do computador. Dispositivo que torna possível tal comunicação.

I

IMPRESSORA — Equipamento que atua como periférico do computador, capaz de imprimir e reproduzir caracteres, fornecendo dados impressos, sob a forma de uma listagem.
INTERFACE — O equipamento que realiza e permite a interligação

entre os equipamentos periféricos e o computador.
ÍNDICE — Valor numérico que determina a posição de um elemento dentro de uma matriz. É necessário um índice por dimensão da matriz. Também denominado subscrito.

K

KILOBYTE, K ou KBYTE — Conjunto de 1024 byte de memória. Assim, 64 K de memória representa 64*1024 = 65536 bytes de memória.

L

LINGUAGEM — Conjunto de instruções ou normas que "ensinam" ao computador como dispor e ordenar os símbolos e caracteres para obter-se uma informação.
LINGUAGEM DE ALTO NÍVEL — Uma linguagem em que os comandos são possíveis de serem entendidos pelo operador, sem que o mesmo tenha, necessariamente, que conhecer a arquitetura do computador ou seu conjunto de instruções internas.
LINGUAGEM DE MÁQUINA — Conjunto de caracteres, sinais e símbolos, usualmente especificados em códigos hexadecimais que são processados pelo microcomputador, numa operação de decodificação das informações e dados nele introduzidos.
LISTAGEM — Lista os dados, instruções ou resultados de um programa que são relacionados em seqüência pela impressora.
LOOP — Conjunto de instruções que são repetidas até que se atinja uma condição pré-determinada.

M

MATRIZ — É um grupo organizado de elementos que podem ser identificados total ou separadamente.
Em BASIC, qualquer nome de variável pode denominar uma matriz. Podem, ainda, assumir múltiplas dimensões: Matriz (n) significa que ela está inserida em uma dimensão: Matriz (n,m) significa que ela está inserida em duas dimensões.
MEMÓRIA — Dispositivo que pode receber e guardar dados e informações, e fornecê-las de novo, quando for acionado um sinal conveniente.
MEMÓRIA DE ACESSO ALEATÓRIO — RAM (*Random Acess Memory*) — Memória em que se tem acesso a qualquer posição, tanto para a escrita como para a leitura. Ao desligar-se o computador, todo conteúdo da memória RAM é perdido.
MEMÓRIA EXCLUSIVA DE LEITURA — ROM *(Read Only Memory)* — Memória exclusiva de leitura. Armazena as informações e "dita" ao computador as normas de operação, de forma que se atinja um resultado.
As informações guardadas na ROM são chamadas de Programa Monitor e Interpretador de BASIC e não podem ser alteradas por programção.
MICROCOMPUTADOR — Computador cuja **CPU** (*Unidade Central de Processamento*) é formada por um microprocessador.

MICROPROCESSADOR — É a unidade central de processamento de um microcomputador formada de um único CI (Circuito Integrado).

P

PROGRAMA — Seqüência de instruções que determinam ao computador o procedimento adotado para resolver determinado problema.

R

ROTINA — Conjunto de instruções dispostas de forma ordenada que instrui o computador sobre como realizar determinada operação.

S

SAÍDA — Processo de transferência de dados da memória do computador para um dado dispositivo (tela, impressora, etc).
SINTAXE — As regras gramaticais para um comando ou instrução. Sintaxe, geralmente, refere-se à pontuação ou ordem dos elementos dentro de uma instrução.
STRING — Seqüência de símbolos e caracteres que são encadeados com um significado estritamente literal. Por exemplo, os caracteres 1234 assumirá seu valor numérico, enquanto que o string "1234" representará apenas uma justaposição dos caracteres 1, 2, 3 e 4. (Note que o string é incluso entre aspas).
STRING NULO — Um string cujo comprimento é zero. Por exemplo: a definição A$ = "" faz com que A$ seja um string nulo.
SALTO — Tipo de instrução que ordena ao computador que se dirija a um ponto determinado do programa, que não é a instrução seguinte na seqüência normal.

U

UNIDADE CENTRAL DE PROCESSAMENTO (UPC ou CPU) — Parte de um computador que controla e realiza as operações aritméticas e lógicas.

V

VARIÁVEL NUMÉRICA — Uma grandeza que pode assumir e presentar qualquer seqüência de números, ou apenas um deles, isoladamente.
VARIÁVEL STRING — Uma grandeza que pode assumir e representar qualquer seqüência de caracteres.

Z

Z80 - Tipo de microprocessador que opera com 8 bits.

Índice Remissivo

Título: _____ Data: __/__/_____
Versão: _____
folha_____de_____

Programador: _____

Descrição: _____

_____

Condições de Entrada:	*Condições de Saída:*

Algoritmo:

Listagem:	
Número de Linha	Linha de Programa

Data: __/__/_____
Versão: _____
folha_____de_____

Título: _____

Programador: _____

Descrição: _____

_____

Condições de Entrada:	*Condições de Saída:*

Algoritmo:

Listagem:	
Número de Linha	Linha de Programa

Data: __/__/_____
Versão: _____
folha_____de_____

Título: _____

Programador: _____

Descrição: _____

_____

Condições de Entrada:	*Condições de Saída:*

Algoritmo:

Listagem:	
Número de Linha	Linha de Programa