MICROINFORMÁTICA

JUDITH MILLER

DOMINANDO O TK90X® E TK95®

LIVROS TÉCNICOS E CIENTÍFICOS EDITORA S.A.

DOMINANDO O TK90X® E TK95®

SÉRIE MICROINFORMÁTICA

ALBRECHT	– Análise Numérica
BACK	– CP/M Sistema Operacional para Microcomputadores
BASTOS	– Programação COBOL
BIANCHI	– Introdução aos Microcomputadores
BIANCHI/BEZERRA	– Microcomputadores-Arquitetura, Projeto e Programação
BORATTO	– BASIC para Engenheiros e Cientistas
BORGES	– BASIC Aplicações Comerciais
BOSCH	– Cobol Fundamentos e Aplicações
CARDOSO	– Programação Estruturada em COBOL
CORDEIRO PAULO	– COBOL – Técnicas e Dispositivos Especiais
DIAS	– O Sistema de Informação e a Empresa
DIAS/GAZZANEO	– Projeto de Sistemas de Processamento de Dados
DIAS/LUCENA/LIMA	– Programação Fortran
EADIE	– Microcomputadores – Teoria e Prática
FURTADO	– Teoria dos Graphos
FURTADO/PASSOS	– Introdução à Programação com PL/1
GANE/SARSON	– Análise Estruturada de Sistemas
GRILLO	– Programação Estruturada com FORTRAN e WATFIV
GUIMARÃES/LAGES	– Introdução à Ciência da Computação
GUIMARÃES/LAGES	– Algoritmos e Estrutura de Dados
GUSMAN/VASCONCELLOS	– Fluxogramas e Programação Cobol
HELT/BONFIM/MARTINS	– PECS – Estatística em Microcomputadores
KATZAN	– Segurança de Dados em Computação
KUGLER/FERNANDES	– Planejamento e Controle de Sistemas de Informação
LARSEN	– Computação para Crianças
LOMUTO/LOMUTO	– Introdução ao Unix
MACIEL/MOREIRA/PINHEIRO	– Transcrição de Dados
MAYNARD	– Programação Modular
McCALEB	– A Microinformática na Empresa
McNITT	– Simulação em BASIC
PACITTI	– Fortran Monitor
PACITTI	– Programação, Princípios
PACITTI/ATKINSON	– Programação e Métodos Computacionais, Vols. 1 e 2
PEREIRA	– Computes, Grillo!
PRAÇA	– Cobol para Micros
PRADO	– Administração de Projetos com PERT/CPM
PRATES	– BASIC Aplicado – Um Enfoque Profissional
PUCCINI	– Programação Linear
ROYO DOS SANTOS	– Processamento de Dados
RUAS	– Curso de Cálculo Numérico
SAPIRA/NESANELOVICZ	– Introdução à Linguagem APL
SCHMITZ/TELES	– Pascal e Técnicas de Programaçao
SOUZA	– Introdução à Linguagém BASIC
SUCESU	– Dicionário de Informática
TAROUCO	– Redes de Comunicação de Dados
VASCONCELLOS	– Computadores Eletrônicos e Processamento
VASCONCELLOS/SZERMAN	– O Centro de Processamento de Dados
VON STAA	– Engenharia de Programas
WOOLDRIGE/LONDON	– O Computador e o Executivo
ZUCCHI	– Transmissão de Dados em Redes de Computadores
ZWICKER	– IBM BASIC

Série: *MICROINFORMÁTICA*

Coordenador: Antônio Roberto Ramos Nogueira

BUI	— Planejamento Executivo com BASIC
BHARUCHA	— dBASE II: Manual do Usuário
BOLOCAN	— Conquistando o Symphony
CAMPBELL/SIMINOFF/YATES	— Micros de Lógica Sinclair
CRUZ	— Guia Prático de Sistemas Gráficos no Apple II/IIe
DEWITT	— Arte e Gráficos no Apple II/IIe
DOBLER	— Linguagem C
ERSKINE	— Linguagem Assembly: 8086 e 8088
FINKEL/BROWN	— TRS-80 Programação Usando Arquivos de Dados
GARCIA/NOGUEIRA	— dBASE Total
GRUENBERGER	— Programando no Apple
PIMENTEL	— Apple: Assembly 6502
PRESS	— IBM PC e suas Aplicações
SCHEIDER	— Multiplan — Manual do Usuário
SILVA/HEIBEL	— SuperCalc²
SOUZA	— WordStar
SWENNSON/POMBEIRO	— Lotus 1-2-3
TREVISAN	— Curso de Programação BASIC
WILLIAMS	— Lotus 1-2-3 de A a Z

CARTÕES DE REFERÊNCIA

ALVES	— Apple
ALVES	— dBASE II
ALVES	— Apple II
DICHAUNE	— WordStar
HADA	— VisiCalc
INOUE/PRATES	— Lotus 1-2-3
KLEIBER	— Cobol 80
LEITE	— 8086/8088
PRATES	— CP/M
PRATES	— MS-DOS
PRATES	— dBASE III
PRATES	— MBASIC
PRATES	— CP/500 2ª Ed.
PRATES	— MSX
PRATES	— PC BASIC
PRATES	— TK90X
PRATES	— dBASE III PLUS
PRATES	— Dataflex
RAMALHO/PRATES	— Turbo Pascal
SANTOS	— Z80
SILVA/HEIBEL	— SuperCalc/SuperCalc²

DOMINANDO O TK90X® E TK95®

JUDITH MILLER

Convento do Sagrado Coração de Woldingham

Tradução Jayme Teixeira Filho

LIVROS
TÉCNICOS E
CIENTÍFICOS EDITORA S.A.

Rio de Janeiro-RJ • São Paulo-SP

Comissão de Computação:
Antônio Roberto Ramos Nogueira
Donaldo de Souza Dias
João José Neto
Luiz de Castro Martins
Ysmar Viana e Silva Filho

Revisor de texto:
Maria da Graça Maia

Revisor de provas:
Marcos Romeu Alves
Alberto Fernando de Araújo

Capa
Programação Visual/AG — Carlos Roberto Studart

CIP-Brasil. Catalogação-na-fonte.
Sindicato Nacional dos Editores de Livros, RJ.

Miller, Judith
M592d Dominando o TK 90X e TK 95 / Judith Miller; tradução Jayme Teixeira Filho. — Rio de Janeiro; São Paulo: LTC — Livros Técnicos e Científicos Editora S.A., 1988.

(Micro Informática)

Tradução de: Beginning BASIC with the ZX Spectrum.

1. BASIC (linguagem de programação para computadores). 2. TK 90X (Microcomputador). 3. TK 95 (Microcomputador). I. Título. II. Série.

86-1423 CDD — 001.6424
CDU — 800.92 BASIC

ISBN (Edição original): 0-333-37995-0
ISBN: 85-216-0511-0

LIVROS TECNICOS E CIENTIFICOS EDITORA S.A.

MATRIZ	FILIAL
Rua Vieira Bueno, 21 20.920 – Rio de Janeiro – RJ Brasil – End. Telegráfico: LITECE Tels.: 580-6055 Vendas: 580-9374	Rua Vitória, nº 486 – 2º andar 01.210 – São Paulo – SP Tel: (011) 223-9866 Caixa Postal 4.817

GOSTARÍAMOS DE REGISTRAR NOSSOS AGRADECIMENTOS À **MICRODIGITAL ELETRÔNICA LTDA.**, NA PESSOA DO SR. MARCIO TADEU DE OLIVEIRA, TÉCNICO DE TREINAMENTO EM HARDWARE, PELA VALIOSA COLABORAÇÃO PRESTADA.

O EDITOR

Dedicado aos meus Pais

Prefácio

O objetivo deste livro é ensinar programação na linguagem **BASIC,** como usada pelos TK 90X e TK 95, para o iniciante. Não é um livro texto, apesar de cobrir praticamente todos os tópicos de programação. A Matemática foi reduzida a um mínimo, e por isso funções aritméticas tais como **ABS, EXP, LN, PI, SGN e SQR** assim como as funções trigonométricas **SIN, COS, TAN, ASN, ARC e ATN** não foram incluídas.

Cada unidade está escrita de forma a se basear apenas na informação previamente coberta pelas unidades anteriores. O que significa se o leitor trabalhar uma unidade de cada vez, ele ou ela não precisará ter nenhum conhecimento além do já encontrado nas unidades anteriores. Se, no entanto, o leitor cobrir as unidades fora de ordem, então aparecerão dificuldades, exceto se ele ou ela tiver experiência prévia.

O livro é melhor usado com um microcomputador à mão e é recomendado que os programas-exemplo sejam nele digitados. Cada unidade dá plena oportunidade de praticar os conhecimentos recém-adquiridos e as soluções para as atividades estão incluídas no fim do livro. Deve ser lembrado, no entanto, que em programação geralmente podem haver várias soluções para um problema, e por isso se o programa do leitor funcionar com sucesso ele é tão aceitável quanto a solução proposta.

Woldingham, 1984 **JUDITH MILLER**

Usando o Teclado do TK95

Apesar de este livro estar baseado na versão padrão do TK 90X, todos os programas e referências ao **BASIC** se aplicam igualmente ao TK 95.

Se você estiver usando um TK 95, haverá pequenas variações nos passos a serem seguidos para obter certos símbolos das teclas. Por isso, por favor, desconsidere as instruções do texto sobre como obter "Modo Estendido" e outras funções; ao invés disso, siga as instruções do **"Spectrum + User Guide"** ou a tabela comparativa, a seguir. Você verá que, em todos os casos, as modificações tornarão muito mais fácil entrar e manipular informações com o teclado.

O TK 95 é essencialmente um TK 90X, com um teclado de verdade. Todo software que funcionar no TK 90X será compatível com o TK 95 e a interface com outros itens não se modificará.

A diferença óbvia que o usuário encontrará está no uso do teclado mais completo, com o qual a entrada fica muito mais fácil.

Críticas ao teclado colorido foram levadas em consideração e o posicionamento de muitas legendas foi sutilmente modificado para melhorar a legibilidade.

Tabela Comparativa dos Equipamentos: ZX SPECTRUM, ZX SPECTRUM PLUS, TK90X e TK 95

	ZX SPECTRUM	ZX SPECTRUM +	TK 90X	TK 95
número de teclas	40	58	40	57
tecla RESET	não	sim	não	não
barra de espaço	não	sim	não	sim
teclas-seta	nas teclas numéricas	separadas ao lado da barra de espaço	nas teclas numéricas	separadas à direita do teclado
teclas DELETE, EDIT, CAPS LOCK e GRAPHS SEPARADAS	não	sim	não	sim
funções acima da tecla	sim	sim	sim	não

Acionamento das Palavras-Chaves

TK 90X	TK 95
palavra-chave acima (em vermelho): SYMBOL SHIFT + tecla desejada.	palavra-chave acima ou símbolos: SYMBOL SHIFT + tecla desejada.
palavra-chave abaixo (em branco): tecla desejada, estando o cursor no modo K.	palavra-chave abaixo: pressionar a tecla desejada, estando o cursor em modo K.
palavra-chave sobre a tecla (em azul) deve ser acionado o modo estendido, pressionando-se, simultaneamente as teclas CAPS SHIFT e SYMBOL SHIFT e, em seguida, a tecla desejada.	palavras-chaves na parte frontal inferior da tela: acima ⟶ EXTENDED MODE + tecla desejada
palavra-chave sob a tecla (em vermelho): aciona-se o modo estendido e, em seguida, mantendo-se pressionada a tecla SYMBOL SHIFT pressiona-se a tecla desejada.	abaixo ⟶ EXTENDED MODE e, em seguida, SYMBOL SHIFT + tecla desejada.
modo gráfico: CAPS SHIFT + tecla 9. Em seguida digitar a tecla desejada ou CAPS SHIFT + tecla desejada.	modo gráfico: GRAPHICS e, em seguida, pressionar a tecla desejada ou CAPS SHIFT + tecla desejada.

Características Técnicas do TK95

UNIDADE DE PROCESSAMENTO:
Microprocessador Z80A (processador de 8 bits) 3,58 MHz

MEMÓRIA:
16 Kbytes ROM (READ-ONLY-MEMORY)
48 Kbytes RAM (RANDOM-ACCESS-MEMORY)

SOFTWARE:
O interpretador BASIC está contido em 16 Kbytes. Também contém controles do monitor, do gravador e periféricos.

CONTROLES:
Cursor de reportagem: indicação de erro em português.
Cursor de programa: >
Cursor de edição: K, L, C, E, G, S

LINGUAGEM DE MÁQUINA:
Programável em linguagem de máquina usando as funções; PEEK, POKE, USR, SAVE CODE, LOAD CODE, VERIFY CODE, IN, OUT, BIN.

PROGRAMAS (SOFTWARE APLICATIVO):
A Microdigital tem disponíveis programas na área profissional, educacional e recreativa.

TECLADO:
- Profissional tipo máquina de escrever, amplo e de fácil manuseio, com 57 teclas.
- Auto repeat.
- Identificação sonora de aceitação.
- Teclas diretas para os comandos duplos, com identificação em cor diferente.
- 93 funções matemáticas.
- 63 caracteres alfanuméricos, com números e letras minúsculas e maiúsculas.
- 16 caracteres gráficos.
- 38 caracteres acentuados português/espanhol.
- 21 caracteres gráficos definidos pelo usuário (8x8) editor UDG incorporado.

LIGHT PEN:
- Poderosa ferramenta para gerar gráficos coloridos, juntamente com textos.
- Pode-se fazer desenhos coloridos, com a possibilidade de animação através de superposição de telas.
- Possibilidade de se armazenar em fita cassete as imagens geradas pelo usuário.

VÍDEO:
- Aceita conexão com TV colorido ou P&B Sistema PAL-M, no canal 3.
- Trabalha em modo invertido controlado por software.
- Som modulado pela TV.
- Display de texto: 24 linhas de 32 caracteres alfanuméricos por linha.
- Display gráfico: resolução de 256 X 192 elementos gráficos.
- Alta resolução e modo texto em 8 cores de duas tonalidades, que podem ser combinados de qualquer forma.
- Editor de UDG (matriz 8x8), para criação de símbolos gráficos definidos pelo usuário.
- Apresenta recursos de flashing e brightness.
- Controle independente de cor (INK e PAPER) por caractere na área de trabalho.
- Controle independente de cor da moldura (border).

UNIDADE DE FITA:
Os programas e dados podem ser armazenados em fita cassete comum, através de gravador convencional. Velocidade de gravação e leitura: 1200 bauds. Permite gravação, leitura e verificação de programas BASIC, blocos de memória, imagens de vídeo, matrizes numéricas e alfanuméricas. Possibilita a integração de dois ou mais programas BASIC através do comando merge.

PERIFÉRICOS:
- Interface impressora
- Light pen
- Interface RS 232
- Mouse

ACESSÓRIOS:
O TK95 é acompanhado de:
1 — cabo de vídeo coaxial.
1 — simetrizador.
1 — fonte de alimentação com chave ON/OFF.
1 — fita brinde, programa arco-íris.
1 — cabo de leitura e gravação.
1 — manual de operação.

CORES:
0 preto
1 azul
2 vermelho
3 lilás

4 verde
5 ciano
6 amarelo
7 branco

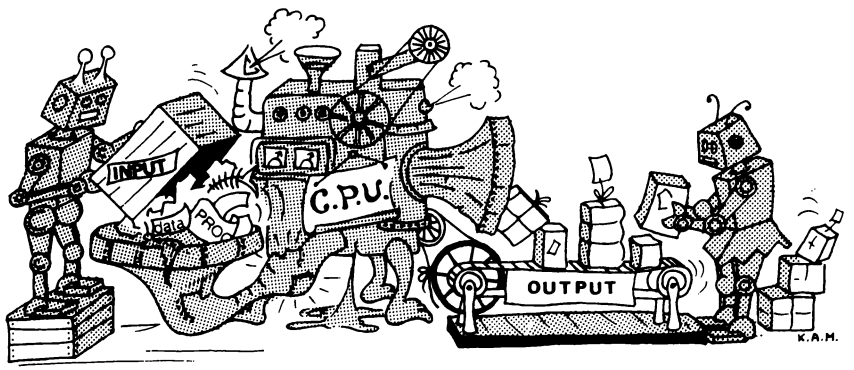
SOUND:
- Saída de som pelo alto-falante da TV.
- Sintetizador de sons controlado por software.
- Comando BASIC que possibilita reproduzir 10 oitavas (130 semitons).

JOYSTICK:
Interface incorporada para joystick, de 4 posições de movimentação e uma de disparo, controladas também pelo teclado.

OPERAÇÃO:
O TK95 é ligado à entrada de antena do televisor, através de um simetrizador e cabo coaxial.
Através da digitação do teclado são introduzidos diferentes comandos ou instruções em linguagem BASIC que podem ser executados assim que forem desejados, ou armazenados em fita com uso do comando SAVE. Para recuperar programas e dados da fita, usa-se o comando LOAD.

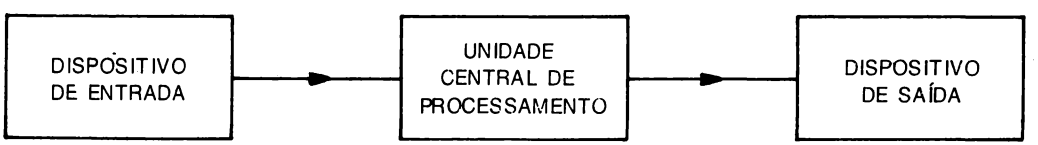
Sumário

Introdução



O Sistema do Microcomputador

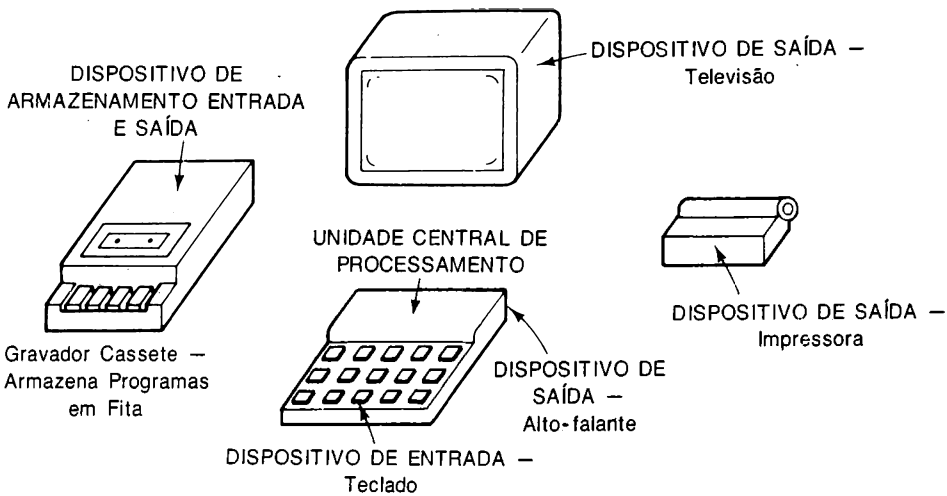
O Sistema é composto de três partes principais; como ilustrado abaixo:



O dispositivo de entrada permite que você entre instruções ou informações no computador. O dispositivo em geral é um teclado que mais parece uma máquina de escrever.

A Unidade Central de Processamento (usualmente abreviada UCP) executa as instruções e manipula as informações para dar um resultado.

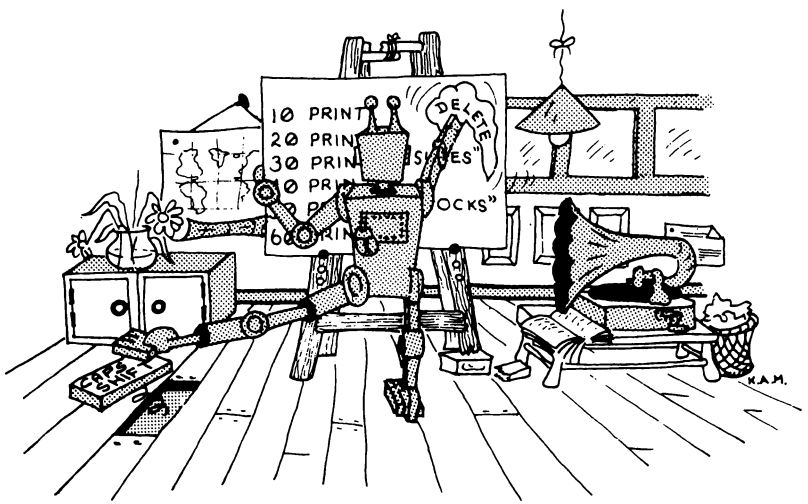
O dispositivo de saída permite a você receber os resultados do processamento.
O dispositivo pode ser uma tela de TV ou uma impressora ou ambos.



Um computador é uma máquina que nos permite resolver certos tipos de problemas. Usa símbolos ou **caracteres** do tipo que usamos no dia-a-dia; por exemplo, as letras do alfabeto, os números de 0 a 9, sinais de pontuação e alguns caracteres especiais tais como +, -, < e *. O TK90X usa uma linguagem chamada BASIC - Beginner's All-purpose Symbolic Instruction Code.

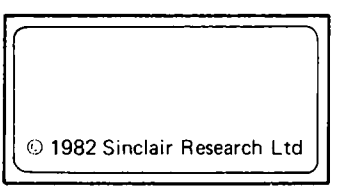
Unidade 1:

Usando o Teclado



Ligue o computador.

Após alguns segundos aparecerá na tela:



Aperte a tecla ENTER.

Um **K** pulsante, chamado **cursor,** aparecerá na tela. É um sinal do computador de que quer algo para fazer; ele está pedindo um comando. Um comando deve começar com uma **palavra-chave.** Palavras-chaves estão escritas em branco em todas as teclas literais.

Por exemplo:

a letra A tem a palavra-chave NEW
a letra P tem a palavra-chave PRINT
a letra R tem a palavra-chave RUN

Quando o cursor é um **K** o computador está dizendo a você que ele está no **modo palavra-chave** e se qualquer letra for pressionada aparecerá na tela uma palavra-chave, e não uma letra. Se você quer que o computador imprima algo, você deve começar com comando com a palavra-chave PRINT, encontrada na letra P.

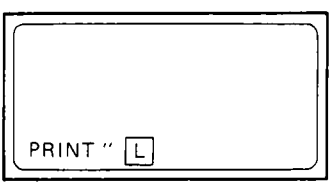
Pressione PRINT. (Note que a palavra aparece inteira; em muitos computadores é preciso digitar todas as letras.)

(Se você pressionar a palavra errada, corrija isso apertando CAPS SHIFT (no canto inferior esquerdo) e DELETE (no canto superior direito) **ao mesmo tempo.**)

O cursor mudou para um **L** brilhante; que é um sinal do computador de que, agora, quer saber o que é para ser impresso. O computador quer algumas letras ou outros caracteres para imprimir e está agora no **modo literal.** O TK90X imprimirá letras (texto) apenas se começar por vírgulas invertidas ou aspas ("). Estas estão também na letra P e são obtidas pressionando SYMBOL SHIFT e a tecla P **ao mesmo tempo.**

Pressione SYMBOL SHIFT e P ao mesmo tempo.

A tela agora mostrará:



Se você olhar as teclas notará que a maioria tem palavras ou símbolos **escritos nelas** em vermelho. Por exemplo:

a tecla M tem o ponto ou ponto decimal
a tecla N tem uma vírgula
a tecla X tem um sinal POUND(1)

Como no caso das aspas, estes sinais só podem ser obtidos pressionando SYMBOL SHIFT ao mesmo tempo que as teclas desejadas. Por conveniência, no futuro, esses símbolos em vermelho serão chamados de **teclas** SHIFT.(2)

Digite uma pequena frase.
Também são necessárias aspas no final da linha.
Pressione SYMBOL SHIFT e P.

---

(1)N.T.: POUND — Unidade monetária da Grã-Bretanha. (£).
(2)N.T.: No original: SHIFTED KEYS.

Agora o computador está pronto para executar o comando.

Pressione ENTER.

Sua frase aparece agora no topo da tela; o computador executou seu comando. O computador também dá uma mensagem: 0 OK, 0 = 1. Isto significa que tudo está OK; não há erros e uma linha foi executada.

Pressione ENTER.

As letras desaparecem e o **K** pulsante reaparece na tela. O computador está de volta ao **modo palavra-chave.**

O TK90X pode imprimir letras pequenas, como você já viu, chamadas **letras minúsculas** ou letras grandes chamadas **letras maiúsculas.** Quando as teclas são pressionadas, as minúsculas são automaticamente impressas. Para obter maiúsculas, a tecla CAPS SHIFT deve ser mantida pressionada enquanto a letra desejada é digitada.

DIGITE: PRINT " (lembre-se que aspas são obtidas com SYMBOL SHIFT e P).
DIGITE: O alfabeto usando maiúsculas e minúsculas alternadamente. (Se você cometer um engano, apague a letra errada com CAPS SHIFT e DELETE.)

Ao final do alfabeto, encerre com aspas.

Pressione ENTER.

O alfabeto aparecerá no topo da tela e a mesma mensagem de antes será escrita na parte de baixo da tela.

Pressione ENTER.

Para escrever sentenças, as palavras devem ser corretamente espaçadas; isso é conseguido usando a tecla BREAK/SPACE. Pratique isso digitando a próxima frase.

DIGITE: PRINT " Estou aprendendo a programar em um TK90X usando a linguagem BASIC."

Pressione ENTER.

É possível usar corretamente os sinais de pontuação ao escrever. Por exemplo:

(,) vírgula – tecla SHIFT N
(;) ponto-e-vírgula – tecla SHIFT O
(:) dois pontos – tecla SHIFT Z
(.) ponto – tecla SHIFT M
(?) ponto de interrogação – tecla SHIFT C
(!) ponto de exclamação – tecla SHIFT 1
(') apóstrofe – tecla SHIFT 7

Pratique usando o teclado para digitar qualquer um dos exercícios seguintes. Lembre-se de começar cada linha com a palavra-chave PRINT seguida de aspas. Ter-

mine cada exercício com aspas e então pressione ENTER. Se você esquecer a palavra-chave ou as aspas e tentar a linha de texto, o computador enviará um ponto de interrogação, que indica erro.

1. DIGITE: a , B ; C : d, e ! F ? g H , i ; J.
2. Digite os números de 1 a 20 com uma vírgula separando cada número (note que o número 0 está na tecla DELETE e aparece com isto: Ø. Um O maiúsculo não pode ser usado em seu lugar).
3. Digite a frase:

A raposa MARROM veloz SALTOU sobre o CAO.

4. Escreva uma pequena carta a um amigo.

Se você deseja digitar várias palavras em maiúsculas, o computador pode ser posto em **modo maiúsculo.** Isto é feito pressionando CAPS SHIFT e CAPS LOCK (tecla 2) ao mesmo tempo. O cursor mudará para **C** brilhante, lembrando a você que ele está no **modo maiúsculo.** Tente isso.

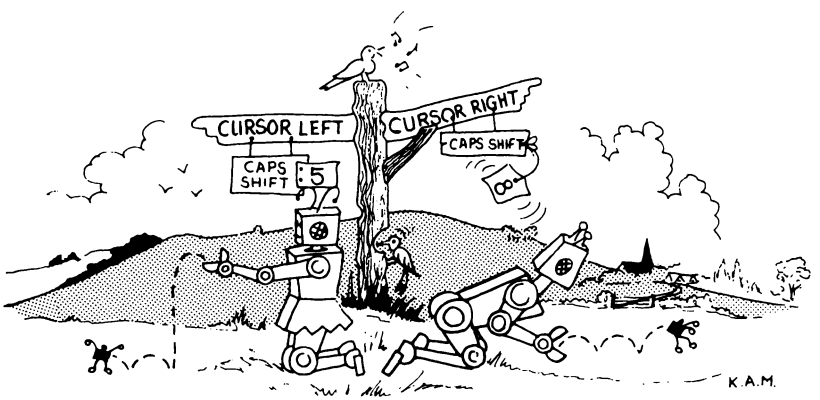
DIGITE: PRINT "PROGRAMAR É DIVERTIDO"
Pressione CAPS SHIFT e CAPS LOCK outra vez; isso colocará o computador novamente no **modo literal.**

Pressione ENTER.

Ao fim de cada exercício sempre limpe a tela ou remova o programa antigo do computador. No momento, tudo que você precisa fazer é pressionar ENTER. A Unidade 3 explicará como remover um programa.

Unidade 2:

Corrigindo Erros



Na Unidade 1 você aprendeu que uma letra pode ser removida pressionando CAPS SHIFT e DELETE. Algumas vezes os erros não são notados imediatamente e você pode ter digitado uma frase inteira antes de se dar conta que errou. Para corrigir esse erro não é necessário apagar a frase toda.

Digite esta frase como está escrita, inclusive com os erros.

PRINT "Ah cinco erhos de gafia nestaa frasei."

Não pressione enter.

Para corrigir a frase, o cursor **L** tem que ser movido. CAPS SHIFT e a tecla 5 movem o cursor para a esquerda, e CAPS SHIFT e a tecla 8 movem para a direita. A palavra 'erhos' deve ser escrita 'erros'.

Usando as teclas certas, mova o cursor até **a direita da letra errada**: erhLos. (Ao mover o cursor segure a tecla CAPS SHIFT com firmeza, caso contrário aparecerão 5s e 8s na sua frase.)

Agora pressione CAPS SHIFT e DELETE e o 'H' desaparecerá. Pressione a tecla R e a palavra fica correta.

Está faltando um 'R' na palavra 'gafia'.

Mova o cursor até a posição da letra que está faltando e aperte a tecla R.

Agora corrija o resto da frase, que deve ficar: "Há cinco erros de grafia nesta frase."

Pressione ENTER.

A sentença correta aparece agora no topo da tela.

Outros exemplos para praticar.

(a) O thempo hojee sta fexado e mublado; Eu acchho ke wai jover.

Versão correta: O tempo hoje está fechado e nublado. Eu acho que vai chover.

(b) A lebe pullou a serca.

Versão correta: A lebre pulou a cerca.

(c) O pade resou pela pas mundial.

Versão correta: O padre rezou pela paz mundial.

Unidade 3:

Fazendo um Programa

DIGITE: 10 PRINT "Isto é um programa."
Aperte ENTER.

A frase toda, além do número e a palavra-chave PRINT, aparece no topo da tela. O computador está agora fazendo um **programa,** armazenando informações para usar depois.

DIGITE: 20 PRINT "Um programa é formado por uma lista de instruções ou comandos."
Pressione ENTER.
DIGITE: 30 PRINT "As linhas de programa são numeradas de dez em dez."
Pressione ENTER.

Note que no fim de cada linha você tem que pressionar ENTER. Isto transfere a linha para o topo da tela e a torna parte do programa. Você não pode digitar outro comando antes disto ser feito.

Pressione a palavra-chave RUN na tecla R seguida de ENTER.

Agora o computador executa o programa e dá uma mensagem similar à que deu anteriormente. Cada frase será impressa na mesma ordem que a do programa original.

Em outras palavras, os números no começo de cada linha indicam ao computador em que ordem devem seguir as instruções.

Tecle ENTER.

O programa voltou à tela. Ele pode ser executado quantas vezes você quiser, mas o computador só o fará se você comandar RUN seguido por ENTER. Isto deve ser feito toda vez que um programa é digitado. É o único jeito de dizer se um programa está correto. O computador não executará um programa que tenha erros ou "**gatilhos**".[1] Também é a única forma de obter o resultado do seu programa.

Tecle NEW, na tecla A seguido de ENTER.

O programa se foi. Ele foi apagado da memória do computador.

Tente os seguintes exercícios. Lembre-se de comandar RUN para cada programa, e de remover antes de começar o próximo.

(1) Digite um programa que imprima seu nome e endereço, como num envelope.

(2) Digite um programa que imprima seu nome, idade e data do nascimento.

(3) Digite um programa que imprima informações sobre sua família.

O programa pode ser trazido de volta à tela teclando-se ENTER. Mais tarde, à medida que seus programas se tornarem maiores, eles não caberão necessariamente numa tela; você deve então teclar LIST (tecla K) seguido de ENTER. Isto trará de volta uma tela com parte do programa e a mensagem "scroll". Então você pressiona qualquer tecla, exceto N de 'não' ou BREAK/SPACE e o resto do programa aparecerá. Isto pode ser repetido até o final do programa. Se você quer só um trecho do programa, então tecle LIST e o número da linha que você deseja, seguido de ENTER.

---

[1] N.T.: No original "BUG" = gatilho, erro, engano, truque.

Unidade 4:

Mais a Respeito de Numeração de Linhas

Linhas de programa são geralmente numeradas em intervalos de dez; por exemplo, 10, 20, 30, 40 etc. A razão disso é que, da primeira vez que um programa é feito, normalmente ficam faltando linhas por erro ou esquecimento. Com intervalos de dez entre as linhas, torna-se possível inseri-las mais tarde. Por exemplo:

```
10      Primeira linha do programa
11  ┐
12  │
13  │
14  │
15  } Espaço para mais nove linhas, se necessário
16  │
17  │
18  │
19  ┘
20      Segunda linha do programa
```

Digite as seguintes instruções que são dadas a uma garota que não sabe como mudar suas meias.

```
10 PRINT "Como trocar de meia"
20 PRINT "Tire seus sapatos"
30 PRINT "Coloque meias limpas"
40 PRINT "Ponha os sapatos"
```

Como um computador, a garota é muito burra e apenas fará exatamente o que lhe foi dito; assim um dia ela não consegue pôr os sapatos, pois ninguém lhe disse que tinha que tirar as meias sujas! Logo, o programa precisa ser melhorado.

DIGITE:

```
25 PRINT "Tire as meias sujas"
```

Tecle ENTER.

Você notará que a linha 25 está automaticamente inserida no lugar correto do seu programa. Se o programa tivesse sido numerado 1, 2, 3 e 4, isso não seria possível, já que o computador aceita apenas números inteiros no início das linhas; você não pode usar 1,5 ou 1,75.

DIGITE:

```
10 PRINT "Como lavar as mãos"
20 PRINT "Abra a torneira"
30 PRINT "Ponha sabão na esponja"
40 PRINT "Esfregue as mãos"
50 PRINT "Feche a torneira"
60 PRINT "Seque as maõs"
```

Pelo menos quatro instruções foram omitidas deliberadamente.

Digite as instruções que faltam, usando a correta numeração de linhas.

Unidade 5:

Editando um Programa

A edição permite ao programador alterar um erro numa linha ou melhorar uma linha que já é parte de um programa.

Digite este programa, incluindo todos os erros deliberados.

```
10  PRINT "olaa fred."
20  PRINT "Vocegosta decomputaçao."
30  PRINT "Nao, nao MUITO"
40  PRINT "nao se preocupe, voce
gostar disso."
```

Há um ou mais erros em cada linha deste programa.

Linha 10 — 'olaa' está escrito errado e 'fred' não está começando por maiúscula.
Linha 20 — falta espaço entre "Você" e "gosta", falta um espaço entre "de" e "computacao" e falta o ponto de interrogação.
Linha 30 — a palavra "muito" está em maiúsculas.
Linha 40 — a frase não está começando em maiúscula e falta uma palavra.

Esses erros de gramática não serão notados pelo computador; ele nunca recebeu aulas de Português e por isso aceita qualquer grafia ou pontuação. Você pode escrever qualquer asneira que o computador a imprimirá, se lhe for ordenado.

Comande RUN; você verá que todos os erros são simplesmente ignorados. O programa é impresso exatamente como foi digitado. Retorne o programa à tela teclando ENTER.

No entanto, com o nosso progresso para programas mais complicados, será necessário corrigir erros, principalmente se forem em BASIC, que o computador entende, e por isso vamos aprender como fazê-lo.

Para corrigir ou editar o programa nós precisamos trazer cada linha com erro de volta para baixo na tela. Veja a linha 40, você verá um sinal de maior, >, conhecido como cursor de programa, entre o número e a palavra-chave PRINT.

Pressione a tecla EDIT (tecla 1) e CAPS SHIFT ao mesmo tempo.

A linha 40 deve ter retornado ao final da tela onde pode ser corrigida do jeito normal. (Há ainda uma cópia desta linha no programa.)

Corrija a linha 40 (veja unidade 2). Quando estiver corrigida, pressione ENTER.

A linha correta toma agora o lugar da incorreta. Você não fica com duas linhas 40.

Depois temos que corrigir a linha 30. Para fazer isso, temos que mover o cursor do programa (>) até essa linha.

Veja a tecla 7, ela tem uma pequena seta branca apontando para cima. Pressione a tecla 7 e CAPS SHIFT ao mesmo tempo. Isso move o cursor do programa até a linha acima; ele agora deverá estar entre 30 e PRINT.

Pressione EDIT e CAPS SHIFT ao mesmo tempo — a linha 30 está agora embaixo na tela. Corrija-a e devolva-a ao programa.

Mova o cursor para a linha 20 (tecla 7 + CAPS SHIFT).

Traga a linha 20 para baixo na tela (tecla EDIT + CAPS SHIFT).

Retorne ao programa.

Edite a linha 10.

Se você mover o cursor depressa demais, apertando a tecla 7 mais de uma vez, você poderá passar da linha que deseja corrigir.

O cursor do programa pode ser movido para baixo pressionando a tecla 6 e CAPS SHIFT.

Algumas vezes é necessário apagar uma linha inteira de um programa. Isso é facilmente feito digitando o número da linha seguido de ENTER. Tente isso no programa anterior.

DIGITE: 10

Pressione ENTER.

Atividade

Digite um programa que imprima quatro frases sobre sua escola. Cada frase deve conter um ou dois erros deliberados. Execute o programa (Comande RUN). Corrija ou edite o programa. Execute o programa correto.

(Lembre-se de remover o programa antigo do computador — palavra-chave NEW seguida de ENTER.)

Unidade 6:

Modo Estendido. Usando o Toca-Fita Cassete

Modo Estendido

Alguns dos comandos usados nesta lição precisam que o computador esteja no **"modo estendido".** Isso porque precisamos usar algumas das palavras em verde e vermelho escritas acima e abaixo das teclas. Por exemplo, vamos precisar do VERIFY, escrito em vermelho abaixo da tecla R, e LLIST, escrito em verde acima da tecla V. O computador é colocado no modo estendido. Pressionando-se CAPS SHIFT e SYMBOL SHIFT ao mesmo tempo, o que muda o cursor pulsante para E. Uma vez no modo estendido, qualquer palavra em verde acima de uma tecla pode ser usada; por exemplo, teclar A produzirá READ. Para obter as palavras vermelhas, a tecla SYMBOL SHIFT deve ser usada novamente, ao mesmo tempo que a tecla desejada; por exemplo, SYMBOL SHIFT e a tecla X produzirão INK.

Usando o gravador cassete

Programas gravados em fitas cassete são conhecidos como **"software".** Apenas softwarè preparado para o TK90X pode ser usado nesta máquina. Se você comprar

uma fita, digamos um jogo, tenha certeza de que o seu computador tem memória suficiente para esse programa. Seu TK90X pode ter 16k ou 48k de memória; você não pode executar um programa de 48k numa máquina de 16k.

Carregando um programa

1. Usando o fio fornecido, conecte a tomada de EAR do computador à tomada EAR do gravador. Use fios da mesma coloração. **Não conecte os fios do microfone.**
2. Ponha o controle de volume do gravador quase no máximo. Não toque nesse controle enquanto estiver carregando o programa. Se você tiver controle de grave/agudo, coloque-o quase no máximo.
3. Insira a fita no gravador, com a face que deseja para cima.
4. Digite no computador a palavra-chave LOAD (tecla J) seguida do nome do programa entre aspas. Por exemplo, se você estiver carregando a fita Horizon, fornecida com o TK90X, digite: LOAD "sidea". (Note a forma como "sidea" está escrita; você deve digitar de forma idêntica. "SIDEA" **não** é o mesmo que "sidea".)
5. Pressione ENTER.
6. Aperte a tecla PLAY do gravador cassete.
7. **Agora espere!** Se você estiver carregando seu próprio programa, então o nome dele aparecerá na tela. Com fitas compradas, outras mensagens e figuras aparecerão na tela. Linhas horizontais são visíveis nas bordas da tela; isso é normal. Se o programa não estiver carregando, você receberá uma mensagem de erro, tal como 'R" "erro de carregamento da fita". Nesse caso pare e volte a fita.

Erros de carregamento podem ser causados por:

(a) O controle de volume está alto ou baixo demais. Se estiver muito baixo o computador não conseguirá captar os sinais. Se estiver muito alto, eles estarão distorcidos. Mexa no nível suavemente, ajustando-o.
(b) Os fios estão conectados errado.
(c) O nome foi digitado errado.

Verifique todas essas coisas antes de tentar novamente.

8. Quando a mensagem OK aparecer na tela seu programa terá sido carregado, e aí pare a fita. Com fitas compradas, algumas vezes você pode ser informado sobre quando parar a fita.

Salvando um programa

Qualquer programa digitado no computador pode ser salvo ou gravado numa fita.

1. Usando o fio fornecido, conecte a tomada do MIC do computador com a do MIC do gravador. Use fios de mesma cor. **Não conecte as tomadas EAR.**
2. Aumente o controle de volume até quase o máximo, se você não tiver controle automático.
3. Insira uma fita virgem, ou qualquer fita ainda com espaço, no gravador.
4. Digite SAVE (Tecla S) seguido do nome do programa entre aspas. Você pode dar o nome que quiser, desde que seja composto de números e letras apenas, e não tenha mais de 10 caracteres. Por exemplo: SAVE "Robots".
5. Tecle ENTER.

6. Uma mensagem aparece na tela: Ponha a fita em movimento e pressione qualquer tecla. Pressione PLAY e RECORD no gravador e depois qualquer tecla do computador.

7. **Agora espere!** Linhas horizontais próximas das margens da tela vão aparecer.

8. Quando o fim do programa é alcançado, será dada a mensagem OK; pare a fita. Infelizmente, OK não significa que seu programa foi gravado; você precisará verificar isso.

Verificando se o seu programa foi gravado

1. Volte a fita ao começo.

2. Conecte a tomada EAR do gravador com a tomada EAR do computador. Use fios de mesma cor. Desconecte os fios dos MIC.

3. Digite a palavra-chave VERIFY (abaixo da tecla R — modo estendido e SYMBOL SHIFT e a tecla + R) seguido do nome do programa entre aspas. Por exemplo: VERIFY "Robots".

4. Pressione ENTER.

5. Pressione PLAY no gravador cassete.

6. **Espere!** O nome do programa deverá aparecer junto com as linhas horizontais se o programa tiver sido gravado com sucesso. Quando o computador terminar a verificação ele enviará a mensagem OK. Isso significa que o programa está na fita. Se o nome não aparece ou a mensagem "R erro de carregamento da fita" aparece, então algo saiu errado. Tente salvar o programa novamente.

Razões para o programa não ser gravado:

(a) Controle de volume muito alto ou muito baixo.
(b) Fios conectados erradamente.
(c) Você pode ter tentado gravar na parte transparente do início da fita.

Se o gravador cassete tiver um contador, anote o número antes de salvar o programa, pois isto pode ser útil como referência se você estiver trabalhando no meio da fita, economizando tempo de busca do computador.

Unidade 7:

Testando seu Conhecimento - Questões Sobre a Introdução e Unidades 1-5

Tente responder de cabeça tantas questões quantas você consiga. Procure as que você não sabe na unidade correspondente — as questões são divididas em seções. Verifique suas respostas no fim do livro.

Introdução: O sistema do microcomputador

1. Cite um dispositivo de entrada.
2. Cite dois dispositivos de saída.
3. Cite um dispositivo que pode ser usado tanto como entrada quanto como saída.
4. O que acontece na Unidade Central de Processamento?
5. (a) O que é um caractere?
   (b) Dê exemplos de caracteres.
6. (a) Qual a linguagem usada por este computador?
   (b) O que significa exatamente este nome?

Unidade 1: Usando o teclado

7. Qual o modo em que o computador está quando aparece na tela um **K** pulsante?
8. Qual o modo em que o computador está quando é ligado a 1ª vez?
9. Como é chamado o **K** pulsante?
10. Em qual modo o computador tem que estar antes de você poder usar um comando como o PRINT?
11. Os comandos aparecem na tela como uma palavra ou você tem que digitar cada letra?
12. Em que tecla você encontra o comando PRINT?
13. Como você apaga uma letra?
14. Qual o modo em que o computador está quando aparece um **L** pulsante na tela?
15. Quando o **K** pulsante muda para **L**?
16. (a) Aspas são parte da linguagem do computador; quando são usadas?
    (b) Como se obtém aspas no teclado?
17. Aspas estão em vermelho na tecla P; dê outro exemplo de um símbolo em vermelho numa tecla.
18. O que significa "tecla shift"?
19. Qual a diferença entre maiúsculas e minúsculas?
20. Como se obtém maiúsculas?
21. Como se obtém espaço entre palavras?
22. Pode a letra O ser usada no lugar do zero?
23. (a) O que significa **"modo maiúsculo"**?
    (b) Como você põe o computador neste modo?
    (c) Como você tira o computador deste modo?

Unidade 2: Corrigindo erros

24. O **L** pulsante pode ser movido p/direita ou para a esquerda ao se pressionar a tecla 5 ou 8. Qual outra tecla tem que ser pressionada ao mesmo tempo?
25. Veja a palavra abaixo, na qual o **L** pulsante está colocado no centro. Se CAPS SHIFT e DELETE forem acionadas, qual letra será apagada, o 'u' ou o 't'?

compu**L**tador

Unidade 3: Fazendo um programa

26. O que deve ser digitado no início de uma linha se você deseja que o comando ou instrução passe a fazer parte de um programa?
27. Se você terminou de digitar uma linha de programa, como você a transfere para o topo da tela?
28. No que consiste um programa?
29. Como você faz para que o computador execute um programa?
30. Quando um programa é executado, o número das linhas e os comandos aparecem na tela?
31. Depois do programa ser executado, como você faz para listá-lo na tela?
32. Pode um programa ser executado mais de uma vez?
33. Como você apaga um programa da memória do computador?

Unidade 4: Mais sobre numeração de linhas

34. Por que geralmente numeramos as linhas de um programa de dez em dez?
35. O que está errado no seguinte programa:

```
10  PRINT  "OLA."
20  PRINT  "Como vai?"
40  PRINT  "Sinto muito."
30  PRINT  "Não muito bem."
```

Unidade 5: Editando um programa

36. O que significa editar um programa?
37. (a) O que é o sinal > que aparece na linha quando um programa é chamado?
    (b) Como se move este sinal para cima e para baixo?
38. Como você devolve uma linha para o final da tela, quando ela já faz parte do programa?
39. Depois de corrigir uma linha de programa e retornar a linha ao programa, você fica com duas versões da linha. Qual a correta e qual a incorreta?

Geral

40. Reescreva o programa abaixo corrigindo todos os erros.

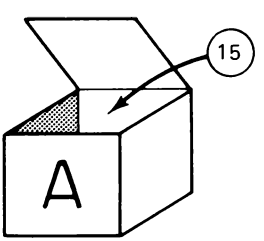
```
1   PRINT  "Você gosta de robots?
20  PRINT  Sim.
15  PRINT  "Eu não."
30  LET    "Eu acho que eles são grandes caixas idiotas de metal."
```

Se houver muitas questões que você não conseguiu responder, provavelmente será melhor ler essas unidades novamente. Se você respondeu corretamente, de cabeça, 30 ou mais questões então você sabe agora como usar o TK90X e está apto a proceder a programação mais avançada.

Unidade 8:

Variáveis Numéricas ou Simples

Uma variável numérica ou simples é uma área do armazenamento na memória do computador, à qual é dado um nome. O espaço de armazenamento só armazena números, daí seu nome. O jeito mais fácil de imaginar isso é pensar numa caixa com nome e que contém um número. No exemplo abaixo, a caixa "A" armazena o nº 15.



CAIXA OU VARIÁVEL CHAMADA 'A'

O computador usa a palavra-chave LET (tecla L) e o sinal de igual (=), o qual é a tecla SHIFT L, para atribuir um valor à variável. Por exemplo:

LET A=15

O computador agora sabe que toda vez que "A" for usado no programa, realmente se estará referindo ao conteúdo da caixa. O próximo programa ilustra isso.

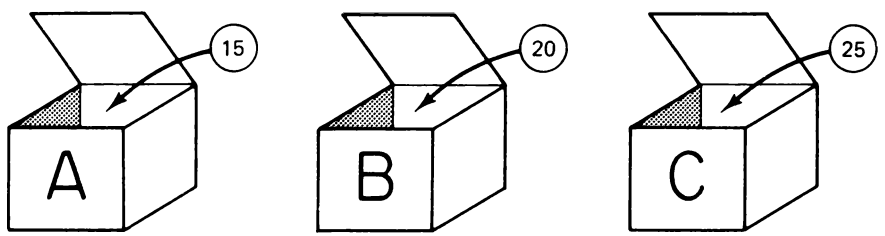
DIGITE:

```
10  LET  A=15
20  PRINT  A
    RUN        NÃO REMOVA
```

O programa mostra que na linha 20, o comando PRINT não está seguido de aspas, pois não estamos pedindo para imprimir a letra A, mas o conteúdo da variável A. Assim, 15 aparece na tela. Se quiséssemos imprimir a letra A, deveríamos usar aspas na forma normal. Isto pode ser mostrado colocando mais uma linha no programa.
Inclua:

```
15  PRINT  "A caixa ou variável A contém um número"
```

Execute (RUN) o programa e então remova.
Podem existir várias variáveis num programa, desde que com nomes diferentes.



Aqui temos três caixas ou valores numéricos. A caixa A permanece com o mesmo valor e fica inalterada pela adição de duas caixas a mais. A caixa B contém 20 e a C, 25. Um programa usando essas variáveis necessitaria de três comandos LET:

DIGITE:

```
10  LET A=15
20  LET B=20
30  LET C=25
40  PRINT A
50  PRINT B
60  PRINT C
```

Execute (RUN), MAS NÃO REMOVA

O resultado é simplesmente: 15
20
25

O nome da variável não tem que estar em maiúsculas, pode estar em minúsculas. Tente isto:

Edite as linhas 40, 50 e 60

```
40  PRINT  a
50  PRINT  b
60  PRINT  c
```

Execute (RUN) E NÃO REMOVA

O resultado é o mesmo. O computador não distingue entre maiúsculas e minúsculas para nome de variável. Outros nomes podem também ser usados para variáveis, mas há duas regras a observar:

(1) O nome deve começar com uma letra.
(2) O nome deve ser composto apenas de letras e números.

Por exemplo, os seguintes nomes são permitidos:

maçãs
CARRO
ZX81
Spectrum
total 2
Dois gatos

} Todos começam com letras
} Todos consistem apenas de letras e números
} Espaços entre palavras também são permitidos

Os seguintes nomes não são permitidos:

2 maçãs (começa com um número)
Spectrum! (contém ponto de exclamação)
CARRO + QUILO (contém sinal de adição)

Uma última observação importante sobre variáveis numéricas é que cada caixa pode armazenar apenas um valor de cada vez. Ela sempre armazenará o último valor dado.

Inclua no programa anterior:

```
 70  LET A=150
 80  LET B=200
 90  PRINT A
100  PRINT B
110  PRINT C
```

RUN

O resultado é: 15 (primeiro valor de A)
20 (primeiro valor de B)
25 (primeiro valor de C)
150 (segundo valor de A)
200 (segundo valor de B)
25 (primeiro valor de C — esta caixa continua inalterada).

Não é possível apreciar ainda a verdadeira importância das variáveis numéricas neste estágio, e você pode estar pensando: por que não imprimir logo o valor dos

números diretamente (por exemplo, PRINT 15, PRINT 20 etc.)? No entanto, elas são uma parte muito importante da programação e gradualmente você verá que têm muita utilidade. Você as encontrará a seguir, em detalhes, na Unidade 11.

Atividades

Escreva o resultado dos seguintes programas:

1.
```
10  LET x=32
20  LET y=44
30  PRINT x
40  PRINT X
50  PRINT "y"
```

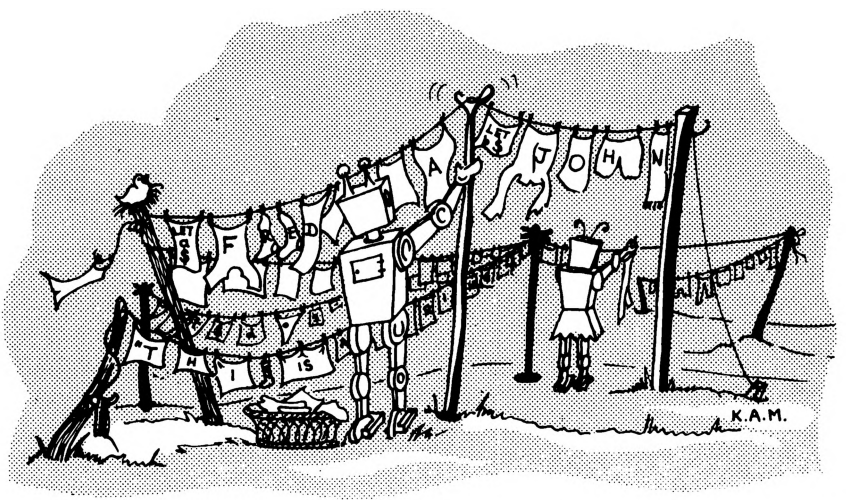
2.
```
10  LET W=68
20  LET caes=2
30  LET W2=53
40  LET w=72
50  PRINT W
60  PRINT "caes"
70  PRINT W2
```

Quais dos seguintes nomes de variáveis são permitidos:

(a) ABC (b) 2ABC (c) abc (d) Dois ABC (e) Abc!
(f) Abc (g) A+B+C (h) A(B)C (i) A2B2C (j) a b c

Unidade 9:

Variáveis String



Revisão: Caracteres são letras do alfabeto, os números de 0 a 9, sinais de pontuação e os símbolos especiais, tais como +, —, *.

Um grupo de caracteres é um **string.** Alguns exemplos:

cão (uma palavra)
Rainha Vitória (um nome)
ZX81 (uma mistura de letras e números)
12, W(2) (uma mistura de caracteres

Uma string pode ser qualquer mistura de caracteres; mesmo um espaço em branco é considerado caractere. Uma **variável string** é um local de armazenamento de grupos de caracteres ou strings. É muito similar a uma variável numérica exceto que o nome tem de ser uma **única letra do alfabeto** seguido do sinal **dólar** ($), que é a **tecla shift 4.** Os caracteres a serem armazenados em uma variável string devem estar contidos entre aspas. Por exemplo:

```
LET a$ = "Fred"
LET b$ = "cao"
LET z$ = "1,2,3,4,5."
```

DIGITE:

```
10  LET a$="Isto é um string"
20  LET s$="Um string é um grupo
de caracteres"
30  PRINT a$
40  PRINT a$
```

Note que, como no caso de variáveis numéricas, não é necessário aspas no comando PRINT, pois o que é para ser impresso é o valor de a$ e s$, e não os caracteres a$ e s$.

Execute (RUN) o programa. Atente para o resultado.

Variáveis string podem ser compostas com outras palavras para fazer parte de sentenças. As variáveis não ficam entre aspas, como já foi mencionado, mas precisam ser separadas das outras palavras da frase por ponto-e-vírgula (;). O ponto-e-vírgula faz parte do BASIC e não pode ser omitido. Por exemplo:

```
10  LET a$ = "Fred"
20  PRINT "Ola"; a$; "Como vai?"
```

O resultado deste programa seria:

```
OlaFredComo vai?
```

Como você pode ver as palavras são coladas uma na outra. Isso por causa do ponto-e-vírgula. Daí ser necessário incluir espaços como parte da frase. (Note que os espaços serão daqui para frente muito utilizados e serão representados por um pequeno traço (-). Em listagens obtidas do próprio computador **apenas o espaço aparecerá**.) Por exemplo,

```
10  LET a$ = "Fred"
20  PRINT "Ola-";a$;"-Como vai?"
```

O resultado será

```
Olá Fred Como vai?
```

Note que os espaços, entre aspas, são contados como caracteres e são impressos, apesar de invisíveis, como outro caractere qualquer.

DIGITE:

Digite detalhes pessoais aqui: não as palavras "seu nome" etc.

```
10  LET n$="seu nome"
20  LET a$="sua idade em anos"
30  LET t$="sua cidade natal ou
vila"
40  LET c$="seu país"
50  PRINT "Meu nome é ";n$
60  PRINT "Minha idade é ";a$
70  PRINT "Eu vivo em ";t$;"que fica
no país ";c$
```

RUN observe o resultado.

Se o seu programa não rodar, examine-o com cuidado; você deve ter esquecido alguma variável.

Existem apenas 26 espaços para armazenamento de strings, um para cada letra do alfabeto.

Quais dos seguintes nomes podem ser usados para armazenar strings, com o nome de variáveis?

(a) a$, (b) M8, (c) T7$, (d) B$, (e) C$3, (f) 8P$
(g) 2$, (h) K$, (i) w$, (j) aa$, (k) ax$, (l) a

Considerando todas as variáveis que você encontrou até agora diga quais dos seguintes comandos BASIC estão corretos:

(a) LET A=87
(b) LET R$="gato"
(c) LET m$=1234
(d) LET K8="Spectrum"
(e) LET L17=38
(f) LET ovos=50
(g) LET leite="vinte-dois"
(h) LET 8A=6

Atividades de Programação (lembre-se de digitar e executar cada programa)

1. a$="João"
   s$="escola"
   Escreva um programa que imprima essas duas variáveis numa frase.
2. m$="Maria"
   a$="Ana"
   b$="12 anos"
   c$="13 anos"
   Escreva um programa usando essas variáveis para imprimir as idades de Maria e Ana.
3. Use variáveis string para nomes de dois amigos e o lugar onde moram. Usando essas variáveis corretamente, imprima duas sentenças que forneçam essa informação.
4. Imprima uma carta que convide um amigo para uma festa. Use as seguintes variáveis:
   n$=nome do amigo.
   d$=data da festa
   w$=nome do dia da festa
   t$=hora da festa.

Variáveis string podem ser compostas; isto é conhecido como **concatenação.**

DIGITE:

```
10 LET a$="Senhora "
20 LET b$="Miller"
30 PRINT a$+b$
40 LET c$="Con"
50 LET d$="vento"
60 PRINT a$+b$;" ensina no ";c$+d$
```

Execute (RUN) o programa e observe o resultado.

Atividades

5. Digite um programa que, usando variáveis strings, imprima seu nome concatenando com seu sobrenome.

6. Usando três variáveis string, divida o nome de uma cidade. Faça um programa que concatene-as e imprima o nome da cidade.

7. O GAROTO GULOSO COMEU VÁRIOS BISCOITOS COM GELÉIA.

Use uma variável string para cada palavra da frase. Por exemplo:

a$ = "O" b$ = "GAROTO" etc.

Faça um programa que concatene essas variáveis, de formas diferentes, montando diversas frases.

Unidade 10:

Extraindo Partes de Strings

DIGITE:

```
10  PRINT "abcdefgh"
20  PRINT "abcdefgh" (1 TO 3)
                      ↑
                  tecla shift F — não digite as
                  letras T e O separadamente
30  PRINT "abcdefgh" (5 TO 8)
40  PRINT "abcdefgh" (2 TO 6)
50  PRINT "abcdefgh" (4)
```

Execute o programa.

O resultado é:
```
abcdefgh
abc
efgh
bcdef
d
```

Você pode ver que se pode extrair qualquer parte do string, usando-se números entre parênteses ao lado do string. Por exemplo (1 TO 3) significa imprimir todos os caracteres do primeiro que é "a" ao terceiro que é "c". Logo o resultado é 'abc'. Se houver apenas um número entre parênteses isso significa, como na linha 50, que será impresso apenas o caractere relativo àquele número; no caso, o quarto caractere: 'd'.

DIGITE: 10 PRINT "aybxcwp"

Adicione linhas de programa para produzir o seguinte resultado:

ayb
cwp
x
bxcw

Execute (RUN) seu programa e verifique os resultados.
Ao invés de digitar sempre um string de caracteres é muito mais rápido usar uma variável string.

DIGITE:

```
10  LET a$="Papai Noel"
20  PRINT a$
30  PRINT a$(1 TO 5)
40  PRINT a$(7 TO 10)
```

Execute (RUN) SEM REMOVER

O resultado é: Papai Noel
Papai
Noel

Você pode estar intrigado com o porquê de 'Noel' não ser (6 a 9), pois o 'N' é a sexta letra; mas é que o espaço em branco entre as duas palavras também é contado.
Adicione instruções para produzir o seguinte resultado.

(a) no
(b) oel
(c) Pa
(d) Pap
(e) l

Use isto como guia:

```
1 2 3 4 5 6 7 8 9 10
P a p a i   N o e l
```

Execute (RUN) o programa e observe os resultados.

Extração e concatenação

Isto é simplesmente tirar partes de uma palavra e adicionar a outras. Veja as linhas 40 e 50 do programa abaixo.

DIGITE:

```
10  LET b$="marcação"
20  LET c$="limitada"
30  PRINT b$+c$
40  PRINT c$(1 TO 5)+b$(5 TO 8)
50  PRINT b$(1 TO 5)+c$(5 TO 8)
```

RUN NÃO REMOVA

O resultado é:
```
marcaçao limitada
limitaçao
marcada
```

Cada palavra é tratada separadamente; por estarem guardadas em variáveis diferentes, a primeira letra de cada uma delas está associada ao número 1.

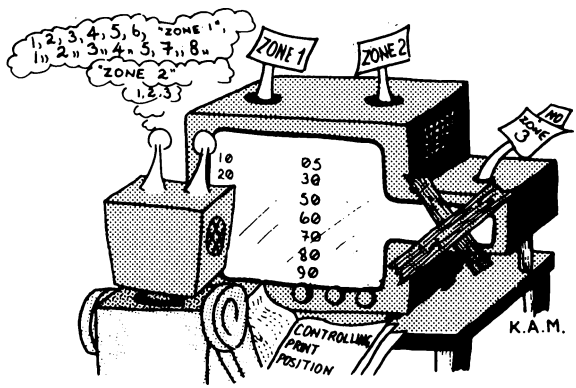
Atividades

1. Altere a linha 30 de tal forma que 'marcacao limitada' apareça sem espaço em branco entre as duas palavras.
2. Inclua instruções para separar as sílabas das palavras.
3. Faça um programa que armazene as palavras "armacao" e "rede". E imprima "ar", "mar" e "acao".

Adicione instruções para imprimir outras palavras além dessas, a partir das duas acima.

Unidade 11:

Usando o Computador Como uma Calculadora



Para aritmética você usa quatro operadores: adição, subtração, multiplicação e divisão. A linguagem BASIC tem os mesmos operadores apesar de dois serem escritos diferentemente.

Símbolo Matemático	Significado	Símbolo em Basic	Tecla
+	soma	+	shift K
—	subtração	—	shift J
×	multiplicação	*	shift B
÷	divisão	/	shift V

Atividades

1. Escreva as seguintes expressões aritméticas usando os símbolos em BASIC.

(a) 3 + 7	(d) 5 × 2 + 8	(g) 5 × 6 × 7
(b) 3 × 7	(e) 24 — 4 × 3	(h) 81 — 27 ÷ 2
(c) 8 ÷ 4	(f) 30 ÷ 3 + 2	(i) 23 ÷ 2 × 9

2. As seguintes expressões foram escritas em BASIC. Calcule os resultados.

(a) 2 + 10 + 5
(b) 2 * 100
(c) 2 * 5 * 10
(d) 50/2
(e) 12 − 7
(f) 6 * 5/10

Um computador sempre executa multiplicação e divisão antes de soma e subtração.

3. Qual é a resposta correta?

(a) 2 + 5 * 4 = 22 ou 26?
(b) 3 * 6 − 5 = 13 ou 3?
(c) 4 + 8/2 = 6 ou 8?
(d) 6/2 − 2 = 1 ou 2?

Digite o seguinte programa para verificar suas respostas.

```
10 PRINT 2 + 6 * 4
20 PRINT 3 * 6 − 5
30 PRINT 4 + 8/2
40 PRINT 6/2 − 2
```

Quando dois operadores são de importância igual (multiplicação e divisão têm a mesma importância, e soma e subtração têm, ambos, a mesma), o computador executa as operações da esquerda para a direita. Por exemplo:

10 * 10/5 = 100/5 = 20
10/10 * 5 = 1 * 5 = 5
10 − 5 + 6 = 5 + 6 = 11

O computador também dará prioridade às partes da expressão em parênteses, antes da multiplicação e divisão. Por exemplo:

3 + 2 * 4 = 11 Mas (3 + 2) * 4 = 20

Atividades

4. Calcule

(a) 6 + 4 * 2
(b) (6 + 4) * 2
(c) 10 − 2 * 3
(d) (10 − 2) * 3
(e) 6 + 18/2
(f) (6 + 18)/2

Digite um programa que verifique suas respostas:

5. Se A é igual a 2, B igual a 5 e C igual a 10, calcule o seguinte:

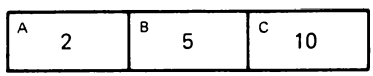
(a) A + B + C
(b) B − A
(c) B * C
(d) C/A
(e) A + C − B
(f) A * C

DIGITE:

```
10  LET A=2
20  LET B=2
30  LET C=10
40  PRINT A+B+C
50  PRINT B-A
60  PRINT B*C
70  PRINT C/A
80  PRINT A+C-B
90  PRINT A*C
```

Execute o programa e verifique suas respostas ao exercício anterior.

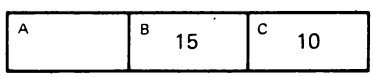
Como você pode ver no programa que acabou de digitar, a aritmética é usualmente feita pelo uso de variáveis numéricas e a palavra-chave LET. As linhas 10, 20 e 30 atribuem valores às variáveis, e daí os espaços para armazenamento ficam como abaixo:



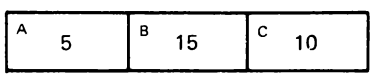
Considere agora o programa:

```
10  LET  B = 15
20  LET  C = 10
30  LET  A = B-C
```

Depois das linhas 10 e 20 a memória do computador armazenará o seguinte:



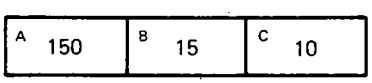
Após a linha 30 teremos:



Se nós acrescentarmos a instrução:

```
40  LET  A = B * C
```

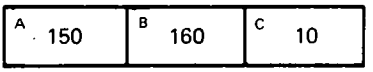
a memória fica:



Observe que o conteúdo de A mudou, armazenando, agora, apenas a última informação. Se incluirmos a instrução

```
50  LET  B = A + C
```

teremos



6. Copie os programas abaixo e os espaços de armazenamento. Preencha com os valores respectivos a cada linha que for executada.

PROGRAMA

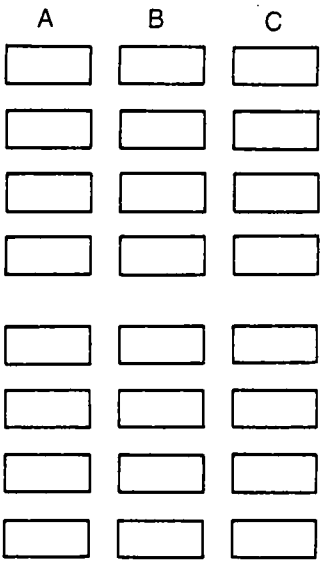
(a)

```
10  LET  A = 12
20  LET  B = 5
30  LET  C = A * B
40  LET  A = C — B
```

(b)

```
10  LET  A = 20
20  LET  B = A * 3
30  LET  C = A/4
40  LET  B = A * C
```

ESPAÇOS DE ARMAZENAMENTO



Atividades de Programação

7. Digite um programa que tenha 2 variáveis numéricas e depois imprima:

(a) sua soma (adição)
(b) sua diferença (subtração)
(c) seu produto (multiplicação)
(d) seu quociente (divisão)

8. O comprimento de um retângulo é 35 centímetros e a largura é 16 centímetros. Digite um programa que use variáveis para comprimento e largura e calcule:

(a) a área do retângulo (comprimento * largura)
(b) o perímetro do retângulo (a soma de todos os quatro lados)

Como as variáveis string, as variáveis numéricas também podem ser compostas com outras palavras. Ponto-e-vírgula é usado para separar palavras de variáveis.

DIGITE:

```
10  LET c=100
20  LET p=40
```

```
30  LET t=80
40  PRINT  "Meu TK90X custou
 ";c;"cruzados"
50  PRINT "A impressora custou "; p;
" cruzados"
60  PRINT "e a televisão custou ";t;"
cruzados"
70  PRINT "O custo total de meu equipamento
foi ";c+p+t;" cruzados"
```

Execute (RUN) o programa e observe os resultados.

Adicione uma linha ao programa que imprima o custo total de equipamento numa sala com seis microcomputadores. (Observação: o computador deve fazer o cálculo, e não você!)

Atividades de Programação

9. O programa abaixo afirma que o Sr. Irineu comprou 9 sacos de cimento a 93 cruzados por saco, e calcula o custo. Reescreva o programa corrigindo os erros:

```
10  LET  x="Sr. Irineu"
20  LET  b$=93
30  LET  C="9"
40  PRINT  Um saco de cimento custa-";b$;"-Cz$.
50  PRINT  x$"-comprou-";c"- sacos."
60  LET  "O custo de";c;"-sacos é";b$/c
```

10. O resultado do programa abaixo dá o nome de três crianças e os resultados de suas provas. Escreve a informação e então calcula a média. Reescreva o programa corrigindo todos os erros:

```
10   LET  a$=Susana Silva
20   PRINT  b$="Joao Souza"
30   LET  c="Maria Costa"
40   LET  a="43"
50   RUN  b$=68
60   LET  c="setenta e três"
70   PRINT  "a$";"- tirou -";a$;"- pontos."
80   PRINT  b$;- tirou -;b;- pontos."
90   PRINT  c"- tirou;"c$;"pontos."
100  LET  "A medida foi -";a$+b$+c/3
```

11. Complete o programa abaixo que calcula a distância de uma viagem de carro. A velocidade do carro é 65 km/h e ele viajou 6 horas.

```
10  ____s=65
20  LET  _=6
30  PRINT  "A velocidade do carro e"; __;"km por hora."
40  PRINT  "O tempo gasto e"; _;"horas."
50  _____ "Distancia da viagem e"; ___;"km."
```

12. Digite um programa que use variáveis para sua:

(a) idade
(b) peso
(c) altura
(d) tamanho de sapato

e escreva essas informações num parágrafo descritivo.

13. Digite um programa que usa variáveis para todas as idades diferentes de pessoas na sua família e que calcula e imprime:

(a) a soma de idade de sua família
(b) a média de idade de sua família

14. Um caminhão carrega 1154 kg de areia por viagem. Faz 19 viagens a cada 4 semanas. Escreva um programa que calcule e imprima:

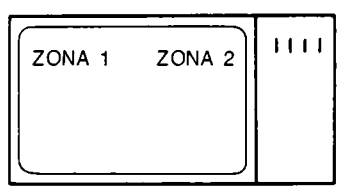
(a) a quantidade de areia carregada a cada 4 semanas
(b) a quantidade carregada em 48 semanas

15. Uma bisnaga de pão custa 44 centavos, um copo de leite 2 cruzados e um quilo de bacon 10,4 cruzados. Escreva um programa que calcule e imprima o preço de 3 bisnagas, 5 copos de leite e meio quilo de bacon, e depois dê o custo total.

Unidade 12:

Controlando a Posição de Impressão

Há duas zonas de impressão na tela do TK90X. A Zona 1 fica à esquerda e a Zona 2 à direita. A Zona 2 começa na 16ª posição da linha na tela.



Para mover um comando PRINT para a ZONA 2 usa-se uma vírgula.

DIGITE: PRINT "Zona 1", "Zona 2"
Tecle ENTER e observe o resultado.
Agora DIGITE: PRINT "Zona 1";"Zona 2"
Tecle ENTER e observe o resultado.
DIGITE: PRINT 1;2;3;4;5;6
Tecle ENTER

O resultado na tela é: 123456

DIGITE: PRINT 1,2,3,4,5,6
Tecle ENTER

O resultado na tela é:
```
1       2
3       4
5       6
```

Note que a vírgula depois do 1 move o 2 para a Zona 2, mas a vírgula após o 2 move o 3 de volta para a Zona 1. Em outras palavras, a vírgula move o PRINT para a próxima zona disponível.

DIGITE: PRINT 1,, 2,, 3,, 4
Tecle ENTER

O resultado na tela é:
```
1
2
3
4
```

Você entende por que isso aconteceu?

Atividades

1. Diga o que aparecerá na tela quando você digitar o seguinte:

   PRINT 1,2,3;4;5,6

   PRINT 1;2,3,,4,,,5

   PRINT 1,,2,,3,,4,5,,6

2. Escreva o comando PRINT que produza o seguinte resultado na tela:

   ```
   1       2
   45      3
   ```

3. Escreva o comando PRINT que produza o seguinte resultado na tela:

   ```
   12      34
   5       6
   ```

Verifique suas respostas digitando os comandos no computador.

Algumas vezes desejaremos deixar espaços em branco entre as linhas. Isto pode ser conseguido usando apenas o próprio comando PRINT.

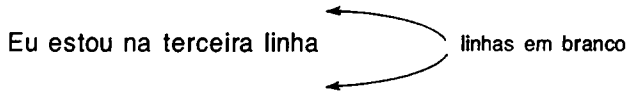
DIGITE:

```
10 PRINT "Eu estou na linha topo"
20 PRINT
```

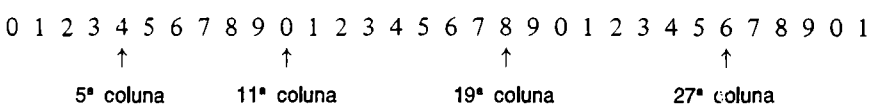
```
30  PRINT "Eu estou na terceira linha"
40  PRINT
50  PRINT "Eu estou na quinta linha"
```

Execute (RUN)

O resultado é: Eu estou na linha do topo



Em outras ocasiões nós podemos precisar mover a posição de impressão através da tela, mas não necessariamente para a próxima zona, daí não podermos usar a vírgula. Para mover a posição de impressão usaremos a função TAB. É uma abreviação para **tabulação,** o que significa arrumar na forma de tabela. TAB está acima da tecla P e é obtida colocando-se o computador no modo estendido e (tecle CAPS SHIFT e SYMBOL SHIFT ao mesmo tempo). TAB é usada com um número; o número indica quantos espaços queremos mover na tela. Pense nos espaços na tela como colunas começando em 0 e terminando em 31, da esquerda para a direita. Daí temos 32 colunas ou posições de impressão. Por exemplo:



Note que como as colunas começam em 0, TAB 4 move a posição de impressão para a 5ª coluna, e TAB 10 move-se para a coluna 11.

DIGITE:

```
10  PRINT "01234567890123456789
012345678901"
20  PRINT "a";TAB 5;"b";TAB 10;"c";
TAB 15;"d";TAB 20;"e";TAB 25;"f";
TAB 30;"g"
```

Execute (RUN)

O resultado é:

```
0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
a         b         c         d         e         f         g
```

Assim, a coisa importante a ser lembrada é que quando você digita TAB 'n' (n representa qualquer número), a impressão começará na coluna seguinte.

O próximo programa mostra como uma tabela de informações pode ser produzida.

DIGITE:

```
10  PRINT "NOME";TAB 10;"IDADE";TAB 20;
"ALTURA"
```

```
20  PRINT
30  PRINT "Sandra";TAB 10;"13";
TAB 20;"153 cm"
40  PRINT "Jane";TAB 10;"14";
TAB 20;"155 cm"
50  PRINT "Susan";TAB 10;"12";
TAB 20;"147 cm"
```

Execute (RUN)

O resultado é uma lista de nomes, idades e alturas em três colunas, sob o cabeçalho apropriado. Note neste programa e no anterior o uso de ponto-e-vírgula (;) para separar as instruções TAB das palavras entre aspas.

Atividade

4. Escreva um programa que imprima a informação na forma de tabela como abaixo:

Dia	**Temperatura (°C)**	**Pluviometria (Polegadas)**
Segunda	10	0.2
Terça	11	0.4
Quarta	13	0.6
Quinta	11	1.0
Sexta	6	0

Unidade 13:

Início e Fim de Programas

Cada programa deve começar com um comando de comentário (**REMARK,** em inglês) abreviado por REM (tecla E). O comando REM apenas documenta o programa, dando-lhe um nome para referências futuras.[1] O computador ignora os comandos REM mas é boa prática de programação incluí-los. De agora em diante comece seus programas com o comando REM. Por exemplo:

```
10  REM  Área de triângulos
                  ou
10  REM  Nomes de países
```

REM são comandos que podem ser usados em qualquer lugar do programa, para mostrar o início de uma nova seção. Por exemplo:

```
10  REM  Areas de triangulos
20  LET  altura=10
30  LET  base=8
```

[1] N.T.: Após o comando REM quaisquer palavras ou caracteres são considerados comentários, não sendo interpretados como comandos.

```
 40  LET   area=base/2*altura
 50  PRINT   area
 60  REM   Areas de retangulos
 70  LET   largura=15
 80  LET   comprimento=25
 90  LET   area=largura*comprimento
100  PRINT   area
```

É também uma boa norma de programação avisar o computador quando o programa acabou. Fazemos isso adicionando o comando STOP (shift A). Por exemplo:

```
110  STOP
```

Unidade 14:

O Comando INPUT

Uma outra forma de dar informações ao computador é com a palavra-chave INPUT. Ela permite que as informações sejam entradas durante a execução do programa, inclusive informações diferentes a cada nova execução. A palavra-chave INPUT pode ser encontrada na tecla I e, como PRINT e LET, pode ser usada apenas quando o cursor é um K (**modo palavra-chave**).

DIGITE:

```
10  REM  Exemplo de input
20  PRINT  "Digite um numero para mim"
30  INPUT x
40  PRINT  "Voce acabou de digitar ";x
50  STOP
```

RUN

Na tela aparecerá: Digite um número para mim

No fim da tela você verá o cursor **L** — o computador espera por sua informação; ele espera até você digitar um número.

DIGITE: 63 Tecle ENTER
Na tela haverá agora: Digite um número para mim
Você acabou de digitar 63

Execute (RUN) o programa com outros números várias vezes.

INPUT pode ser usado tanto para variáveis string quanto para numéricas.

DIGITE:

```
10 REM  Input com variaveis string
20 PRINT  "Qual o seu nome?"
30 INPUT  n$
40 PRINT  "Meu nome e ";n$
50 PRINT  "Onde voce mora?"
60 INPUT  a$
70 PRINT  "Eu moro em ";a$
80 STOP
```

Execute o programa. Note que o cursor **L** aparece entre aspas para lembrá-lo de que um string está sendo pedido.

Usar INPUT significa poder rodar o programa quantas vezes quiser, a cada vez mudando o conteúdo das variáveis. Na Unidade 11 você escreveu um programa para achar a área e o perímetro de um retângulo, usando o comando LET que permitia rodar o programa apenas uma vez com um conjunto de medidas. No entanto, usando INPUT o programa pode ser executado várias vezes usando diferentes medidas a cada vez.

DIGITE:

```
10 REM  Area e perimetro de retangulos
20 PRINT  "Entre com o comprimento do retangulo"
30 INPUT  L
40 PRINT  "Entre com a largura do retangulo"
50 INPUT  B
60 PRINT  "A area do retangulo e "; 1 * b; " cm 2"
70 PRINT  "O perimetro e "; (1 + b) * 2; " cm"
80 STOP
```

Execute o programa várias vezes com diferentes medidas.

As linhas 20 e 40 no programa anterior são conhecidas como **input prompts**[1]; em outras palavras, elas informam a você que informação o computador está querendo.

Um input prompt pode ser escrito como um comando PRINT separando, como no programa anterior, ou no próprio comando INPUT, como nos dois exemplos a seguir.

DIGITE:

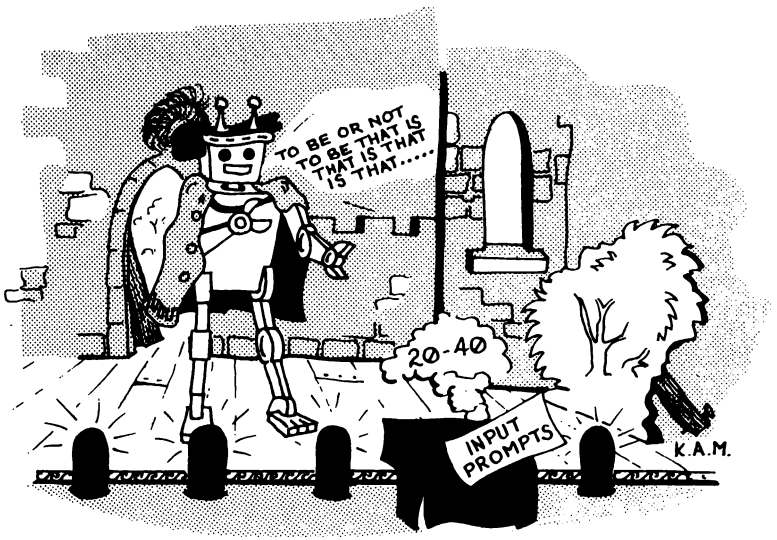
```
10 REM  Input prompt
20 INPUT  "Digite um numero "; n1
```

---

[1] N.T.: INPUT PROMPTS = Sinais ou avisos de entrada. Foi mantido no original inglês por ser razoavelmente difundido.

```
30  INPUT  "Digite um segundo numero "; n2
40  PRINT  n1; " + "; n2; " = "; n1 + n2
50  STOP
```

Execute o programa várias vezes. Note que o input prompt aparece no fim da tela junto com o cursor, e não no topo da tela como no exemplo anterior.



DIGITE:

```
10  REM  Escrevendo um poema
20  INPUT  "Qual o seu nome? "; n$
30  PRINT  "Um poema de "; n$
40  INPUT  "Digite uma palavra que rime com computador "; a$
50  PRINT  "Aqui esta o poema"
60  PRINT  "Antes eu temia"
70  PRINT  "O computador"
80  PRINT  "Mas agora eu"
90  PRINT  "sinto "; a$
100 STOP
```

RUN o programa várias vezes alterando as variáveis n$ e a$.

Você pode imaginar que, num futuro próximo, sua TV, seu telefone e seu computador pessoal estarão interligados a um sistema "inteligente", de tal forma que, ao assistir a um comercial de TV que o interesse, você possa discar um número e obter o seguinte:

DIGITE:

```
10  REM  Serviço de informacoes de videotexto
20  PRINT  "Ola"
30  PRINT  "Este e um servico de informacoes de videotexto"
```

```
 40  PRINT
 50  PRINT  "Por favor, responda as seguintes perguntas"
 60  PRINT
 70  PRINT  "Seu nome? ";
 80  INPUT  n$
 90  PRINT  n$
100  PRINT  "Seu numero de telefone? ";
110  INPUT  t$
120  PRINT  t$
130  PRINT  "Nome de sua rua? ";
140  INPUT  h$
150  PRINT  h$
160  PRINT  "Numero de sua casa? ";
170  INPUT  r$
180  PRINT  r$
190  PRINT  "Nome de sua cidade? ";
200  INPUT  c$
210  PRINT  c$
220  PRINT  "Nome de seu pais? ";
230  INPUT  p$
240  PRINT  p$
250  PRINT
260  PRINT
270  PRINT
280  PRINT  "Obrigado pela atencao "
290  PRINT  "Detalhes de nossos serviços e produtos"
300  PRINT  "Ser-lhe-ao enviados em breve"
310  PRINT  "Seus dados pessoais estao sob sigilo absoluto"
320  STOP
```

Atividades

Escreva programas para os problemas abaixo. Teste os programas no computador.

1. Entre quatro números e os imprima. Encontre sua soma e sua média. Imprima os resultados.

2. Usando as variáveis abaixo, escreva uma carta convidando alguém para uma entrevista. Use INPUT para que a carta possa conter diferentes informações a cada vez.

Variáveis

- n$ = nome da pessoa a ser entrevistada
- d$ = data da entrevista
- t$ = hora da entrevista
- p$ = lugar da entrevista
- m$ = nome do entrevistador
- s$ = nome da pessoa que enviou a carta
- l$ = data da carta

3. Entre a soma de dinheiro que você deseja guardar. Entre a soma de dinheiro que você pode poupar a cada semana. Calcule o número de semanas que levará para alcançar a soma desejada. Transforme o número de semanas em meses. Imprima os resultados.

Unidade 15:

Descrições de Programas - Algoritmos

Um **algoritmo** é a descrição de um programa por escrito. É uma forma de expressar uma solução de um problema para o qual o programa pode então ser escrito. Há duas formas principais de escrever algoritmos:[1]

1. Usando palavras
2. Usando um fluxograma ou plano.

Trataremos nesta unidade do método 2.

Fluxogramas

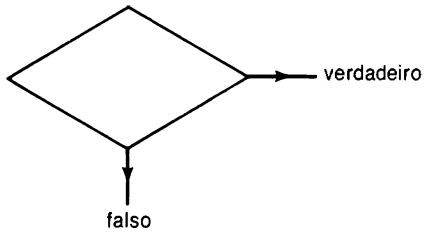
Um **fluxograma** usa diferentes formatos de caixas.

---

[1] N.T.: Tecnicamente, um algoritmo é um conjunto de instruções, dispostas seqüencialmente, executadas segundo algum critério lógico, para atingir um determinado fim, e que sempre pára. Há muitas técnicas de construção e descrição de algoritmos. Embora há muito ultrapassada para algoritmos grandes e complexos, a técnica de fluxograma é aceitável, ainda, para algoritmos simples e de poucas instruções.

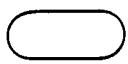
(a) Para **input** (entrada de dados) e **output** (saída de dados).

(b) Para **decisões** (decidindo se uma afirmação é verdadeira ou falsa).



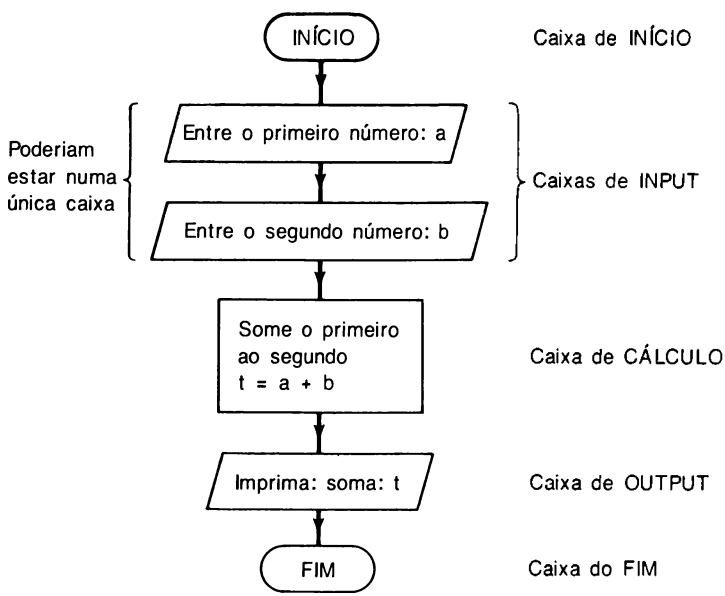
(c) Para **cálculos** (expressões aritméticas).

(d) Para **início** (começo do programa) e **fim** (fim do programa).



Para ver como essas caixas são usadas, vamos considerar o problema de entrar dois números no computador, somando-os e imprimindo o resultado.

Fluxograma



Podemos escrever um programa usando este fluxograma como guia.

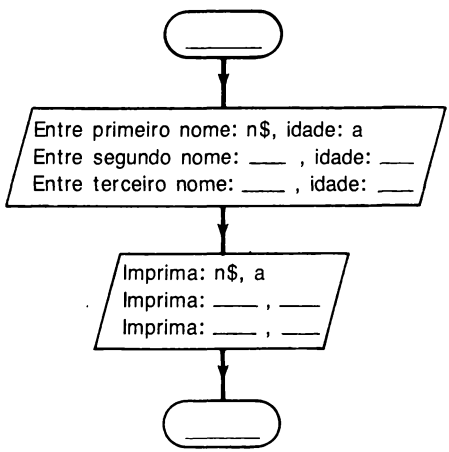
```
PROGRAMA    10  REM  Somando Numeros
            20  INPUT  "Entre o primeiro numero"; a
            30  INPUT  "Entre o segundo numero"; b
            40  LET  t = a + b
            50  PRINT  t
            60  STOP
```

Atividades

1. Entre os nomes de três garotas e suas idades e imprima a informação numa tabela.

Variáveis a usar: n$: primeiro nome
g$: segundo nome
h$: terceiro nome
a : idade da primeira garota
b : idade da segunda garota
c : idade da terceira garota

Copie e complete o fluxograma.



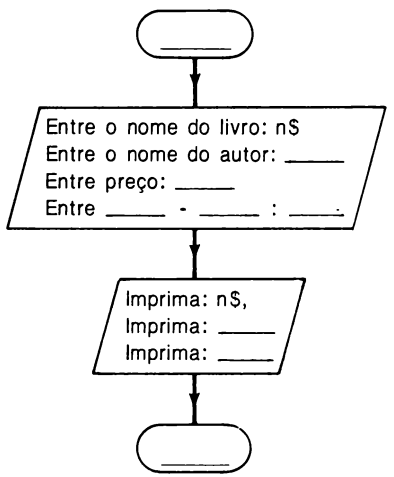
Copie e complete o programa

```
10  REM  Nomes e idades
20  INPUT  "Nome da primeira garota? "; n$
30  INPUT  "Idade da primeira garota? "; _____
40  INPUT  "Nome da segunda garota? "; _____
50  _____  "Idade da _____garota? "; _____
    INPUT  —Nome da terceira garota?_____ , ____
70  _____  "____ _____ _____ _____? "; _____
80  PRINT  "NOME", "IDADE"
90  PRINT  n$, a
100 _____  _____ , _____
    _____  _____ , _____
120 _____
```

2. Entre o nome de um livro, o autor, o preço e a data de publicação. Imprima essas informações.

Use as Variáveis: n$: nome do livro
a$: nome do autor.
p : preço
d : data de publicação.

Copie e complete o fluxograma



Copie e complete o programa

```
10  REM  Livro e autor
20  ____ "Nome do livro?"; _________
30  INPUT  "_____ ?"; _____
40  _____ "Preco do livro?" _____ p
__  _____ ____  "_____"; _____
60  PRINT _____ ; "por"; _____
70  _____ "Preco do livro"; _____
80  _____ ____ " _____ "; _____
__  _____
```

3. Liste as variáveis, desenhe os fluxogramas e escreva os programas para os problemas abaixo. Preencha com suas próprias informações quando necessário. **Teste seus programas no computador.**

(a) Calcule a área de um triângulo e imprima o resultado (área = 1/2 base × altura).

(b) Entre os nomes de três pessoas e suas datas de nascimento respectivas e imprima essa informação. (Serão as datas de nascimento numéricas ou string? — pense nisso antes de escolher as variáveis.)

(c) Entre uma palavra de pelo menos sete letras e imprima três outras palavras usando apenas as letras da palavra original.

(d) Entre os nomes de três cidades e suas populações. Imprima essa informação. Calcule a população total e imprima o resultado.

(e) Um garoto tinha 216 chocolates e os repartiu igualmente entre seus 12 amigos. Imprima o número de chocolates que cada amigo recebeu.

(f) Entre com um comprimento em pés e transforme em centímetros (1 pé = 30,5 centímetros).

(g) Uma garota tem Cz$ 950,00 para gastar. Ela compra um short por Cz$ 260,00, um vestido por Cz$ 440,00 e um par de tênis por Cz$ 160,00. Calcule o que ela gastou e quanto sobrou. Imprima essa informação.

Unidade 16:

Laços[1] *- Usando GOTO*

GOTO é um comando ou palavra-chave encontrado na tecla G. Como para outros comandos, é preciso que o cursor seja um **K** (**modo palavra-chave**) para que possa ser usado.

```
DIGITE: 10  PRINT  "Ola"
        20  GOTO     10
RUN
```

A palavra "Ola" preencherá a tela e a mensagem 'scroll?' será impressa.

Tecle Y para sim.

A tela será novamente preenchida com a mesma palavra. Isso pode ser repetido quantas vezes desejar.

Tecle N para não.

O programa agora pára com a mensagem:

D BREAK-CONT repeats, 10:1

---

[1]N. do T.: No original 'LOOP'. Embora amplamente difundida no meio técnico, a palavra 'LOOP' tem uma equivalente exata em português — 'LAÇO' — também razoavelmente difundida. Optamos pela versão em português.

Tecle ENTER.

O programa volta à tela.

Saltar de volta para outra parte do programa é conhecido como **looping.**[1] O exemplo anterior pode ser repetido continuamente e por isso é conhecido como **laço infinito**. O comando GOTO pode ser usado para forçar a execução de qualquer linha.

DIGITE:

```
10  LET  x=0
20  LET  x=x+1
30  PRINT  x
40  GOTO  20
```

RUN o programa.

A tela será preenchida com os números 1 a 22.

Tecle Y.

A tela se encherá com os números de 23 a 44. Como no exemplo anterior, um **laço infinito** como este pode ser repetido um sem-número de vezes.

Como o programa funciona? Você pode explicar?

Tecle N para 'não' e remova o programa.

DIGITE:

```
10  REM  Total de números
20  LET  total=0
30  INPUT  "Entre um número "; n
40  LET  total=total+ n
50  PRINT  n, total
60  GOTO  30
```

RUN

Este é mais um laço infinito, por isso depois de entrar vários números aperte STOP (shift A) seguido por ENTER, o que produzirá a mensagem:

H STOP in INPUT, 30 : 1

Tecle ENTER e o programa voltará à tela.

Este programa é parecido com o anterior, onde x começava em 0 e era adicionado de 1 a cada vez que o comando GOTO na linha 40 mandava de volta à linha 20. Neste programa, a variável 'total' começa em 0, mas na linha 30 um input prompt pede um número que é então adicionado ao 'total' na linha 40. A variável total muda, portanto, de valor, deixando de ser 0. A linha 50 imprime o número lido numa coluna e o 'total' em outra. O computador volta à linha 30, para repetir o laço com outro número, e então o conteúdo do total se modifica novamente.

Vamos considerar o programa novamente, com os números 5, 10 e 15 sendo lidos.

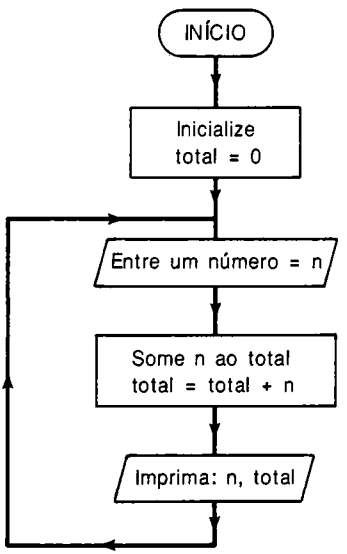
---

(1) N.T.: Embora estejamos usando 'LAÇO' para 'LOOP', o caso de um programa "entrar em loop" não seria bem interpretado pela expressão "entrar em laço".

Resultado do programa

Linha 20: a variavel 'total' e igual a 0
Linha 30: o numero 5 e lido
Linha 40: a variavel 'total' passa a ser 0 + 5 = 5
Linha 50: imprime 5 5
Linha 60: o programa volta a linha 30
Linha 30: o numero 10 e lido
Linha 40: a variavel 'total' passa a ser 5 + 10 = 15
Linha 50: imprime 10 15
Linha 60: o programa volta a linha 30
Linha 30: o numero 15 e lido
Linha 40: a variavel 'total' muda outra vez e passa a ser 15 + 15 = 30
Linha 50: imprime 15 30
Linha 60: o programa volta a linha 30

Fluxograma para programa



Note que não há caixa para o comando GOTO. Ele é indicado pela direção das setas.

O comando GOTO pode também ser usado para saltar à frente no programa; várias instruções podem ser puladas ou omitidas se necessário. O próximo programa é um exemplo disso.

DIGITE:

```
10  PRINT  "O TK90X e o ";
20  GOTO  40
30  PRINT  "pior ";
40  PRINT  "melhor ";
50  PRINT  "computador"
```

RUN

O que aconteceu com a linha 30?

GOTO é sempre usado com o número de linha, o qual pode ser uma expressão aritmética. Por exemplo:

GOTO 20 + 40 é o mesmo que GOTO linha 60

Isso pode ser útil se você quer adicionar uma variável a um número de linha; veja o programa seguinte.

DIGITE:

```
10  INPUT  "Entre o mes do ano como um numero
           de 1 a 12 ";m
20  PRINT  "Mes ";m;" e ";
30  GOTO     50+m
50  GOTO     10
51  PRINT  "Janeiro": GOTO  10
52  PRINT  "Fevereiro": GOTO  10
53  PRINT  "Marco": GOTO  10
54  PRINT  "Abril": GOTO  10
55  PRINT  "Maio": GOTO  10
56  PRINT  "Junho": GOTO  10
57  PRINT  "Julho": GOTO  10
58  PRINT  "Agosto": GOTO  10
59  PRINT  "Setembro": GOTO  10
60  PRINT  "Outubro": GOTO  10
61  PRINT  "Novembro": GOTO  10
62  PRINT  "Dezembro": GOTO  10
```

Note que cada uma dessas linhas tem dois comandos: PRINT E GOTO. Isso é possível usando dois pontos (:) que separam os comandos. Após digitar (:) o cursor volta automaticamente a ser **K**.

RUN

Para parar o programa tecle STOP seguido por ENTER.

Neste programa, a linha 30 causará um salto para a linha 51 se você entrar o número '1'; para a linha 52 se você entrar o '2', e assim por diante.

Se o número 0 é digitado por engano o que acontece? E se for um número maior que 12?

Note que neste programa, nas linhas 51 a 62, foi possível escrever dois comandos em cada linha, graças aos dois pontos (:). Por exemplo, PRINT uma palavra e então GOTO para outra linha. O dois pontos (:) é um sinal para que o computador volte ao modo palavra-chave **K**, para que outro comando possa ser digitado.

Atividades

1. Desenhe um fluxograma para o programa — exemplo sobre

"o TK90X é o melhor computador"

2. Faça um programa que leia dois números separados, some-os, imprima o resultado e salte de volta para o início, formando um loop infinito. Teste seu programa no computador.

3. Um homem tem três contas correntes com os seguintes valores: Cz$ 381,20, Cz$ 623,65 e Cz$ 127,84. Ele decidiu transferir tudo para uma conta só e quer saber o total que tem. Ele continua guardando somas diferentes a cada mês na nova conta. O banco manda-lhe um extrato no fim de cada mês, com a data, quanto ele guardou e o total. Esse esquema continua por vários anos. Desenhe o fluxograma e escreva o programa que calcule e imprima essas informações. Teste no computador.

Unidade 17:

Comparações

Um computador, ao contrário de uma calculadora, pode comparar duas porções de informação. Ele pode decidir se uma afirmação é verdadeira ou falsa, e agir de acordo. Para fazer isso, o computador usa o comando IF, encontrado na tecla U, seguido da palavra THEN, na tecla shift G. IF, assim como outros comandos, pode ser usado apenas quando o computador está no **modo palavra-chave**; o cursor é um **K**. O comando simplesmente indica que IF uma certa afirmação é verdadeira THEN executa a instrução a seguir na mesma linha. Veja o exemplo da página seguinte, que compara informações usando o sinal de igual (=) e o de diferente (< >).

Execute uma vez com a resposta certa e uma vez com a resposta errada.

Neste programa a linha 40 significa que se x for igual a 10 + 15 então imprima correto; senão ignore o resto das instruções da linha. Aí o programa passa para a linha 50, que testa a afirmação novamente, e se x for diferente de 10 + 15 então imprime 'errado', a resposta correta é 10 + 15'. Caso contrário, ignora o resto da linha.

Pergunta: Por que foi possível usar o PRINT no meio das linhas 40 e 50? Por que não apareceu a letra P?

DIGITE:

```
10  PRINT  "10+15=";
20  INPUT  "Entre com a resposta "; x
30  PRINT  x
40  IF  x=10+15  THEN PRINT "Correto etc"
```

tecla U

tecla shift G

próxima instrução

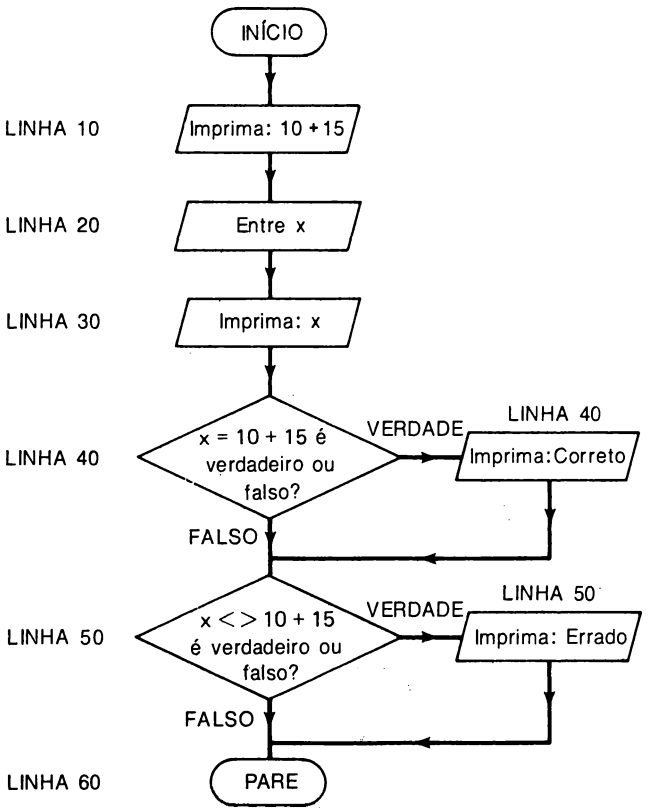
Ao digitar essas linhas observe o que acontece ao cursor quando você tecla THEN

o conteúdo da variante numérica x está sendo comparado à soma 10 + 15

```
50  IF  x <> 10+15  THEN PRINT "Errado, a resposta correta é ";
10 + 15
60  STOP
```

tecla shift W (e não shift T e shift R)

Fluxograma desse programa



O exemplo a seguir mostra que não são só as variáveis numéricas que podem ser comparadas: variáveis string também.

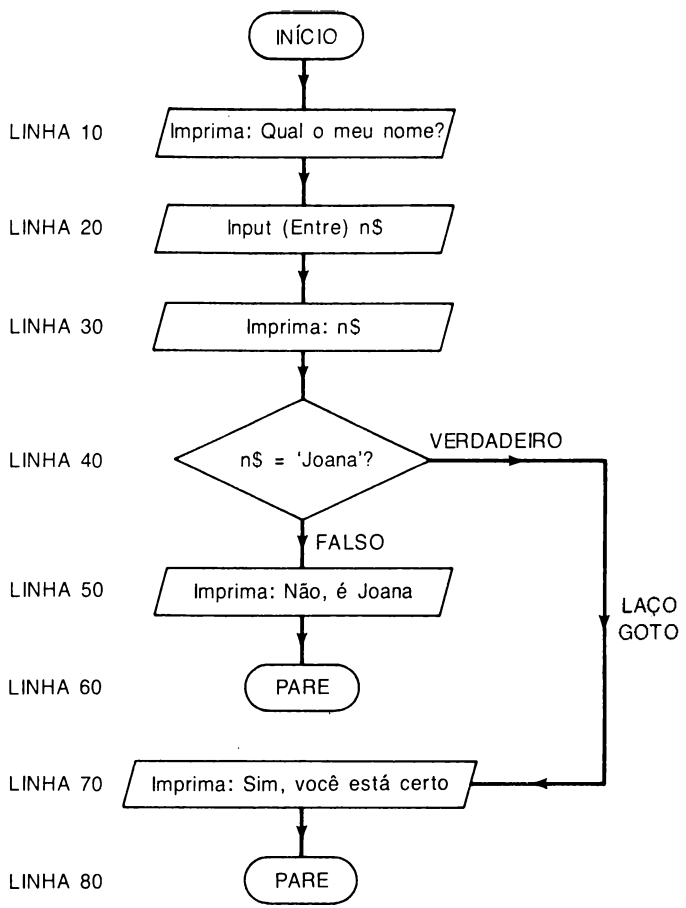
DIGITE:

```
10  PRINT  "Qual o meu nome? ";
20  INPUT  "Entre um nome "; n$
30  PRINT  n$
40  IF  n$="Joana" THEN GOTO 70
50  PRINT  "Nao, e Joana"
60  STOP
70  PRINT  "Sim, voce esta certo"
80  STOP
```

Execute o programa diversas vezes, tentando nomes diferentes.

Este programa ilustra um laço com GOTO baseado na linha 40. Essa linha significa que se n$ for igual a 'Joana' então desvie para a linha 70. Senão ignore a instrução e vá para a linha 50.

Fluxograma do programa



Pergunta:

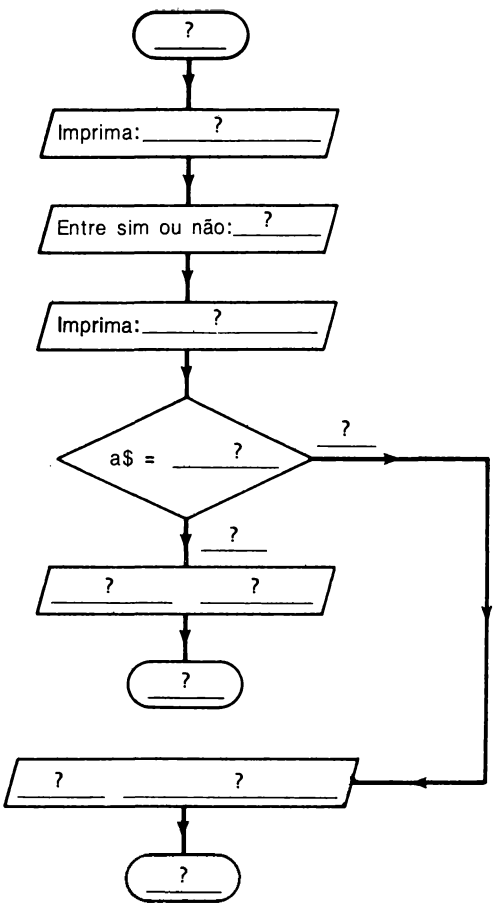
Por que é preciso o comando STOP na linha 60?

Atividades

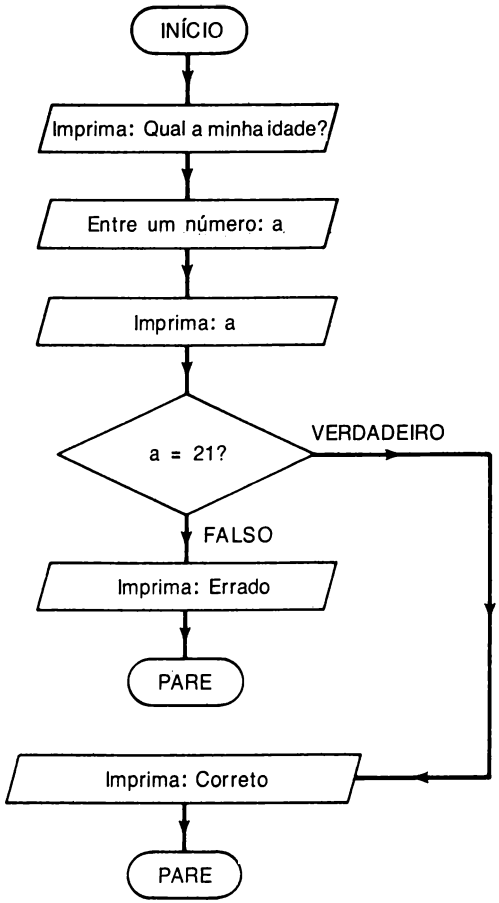
1. Estude o programa a seguir e depois copie e complete o fluxograma.

```
10  PRINT  "Voce gosta de tortas e biscoitos? ";
20  INPUT  "Entre sim ou nao "; a$
30  PRINT  a$
40  IF  a$="sim"  THEN GOTO 70
50  PRINT  "Voce vai continuar magro"
60  STOP
70  PRINT  "Voce vai ficar gordo"
80  STOP
```

Fluxograma



2. Estude o fluxograma abaixo e escreva o programa.



3. Desenhe o fluxograma e escreva o programa para o problema seguinte.

Imprima uma expressão aritmética onde um número é subtraído de outro. Entre a resposta e imprima. Imprima se a resposta está certa ou errada. Se a resposta for errada, imprima a correta. Teste seu programa no computador.

4. Escreva o programa que pergunta a uma pessoa a cor de seus olhos. Use uma variável string para representar a cor de seus olhos. Imprima se o palpite está certo ou errado. Teste seu programa no computador.

O próximo programa é um pouco mais complicado, pois compara informações nas linhas 30 e 80 e usa vários laços com GOTO.

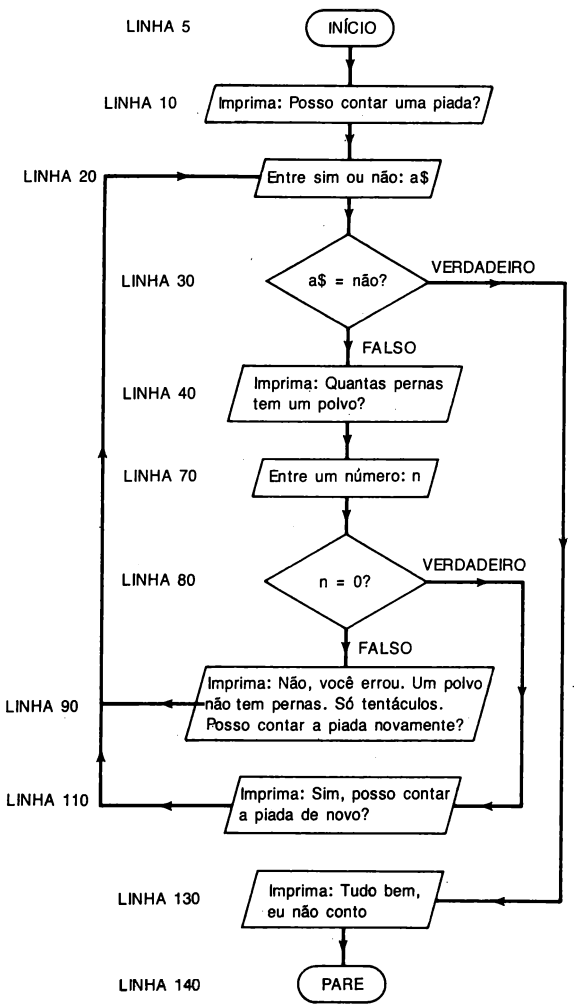
DIGITE:

```
 5  REM  CONTANDO UMA PIADA
10  PRINT  "Posso contar uma piada?"
20  INPUT  "Entre sim ou nao "; a$
30  IF  a$="nao"  THEN GOTO 130
40  PRINT  "Quantas pernas tem um polvo?"
```

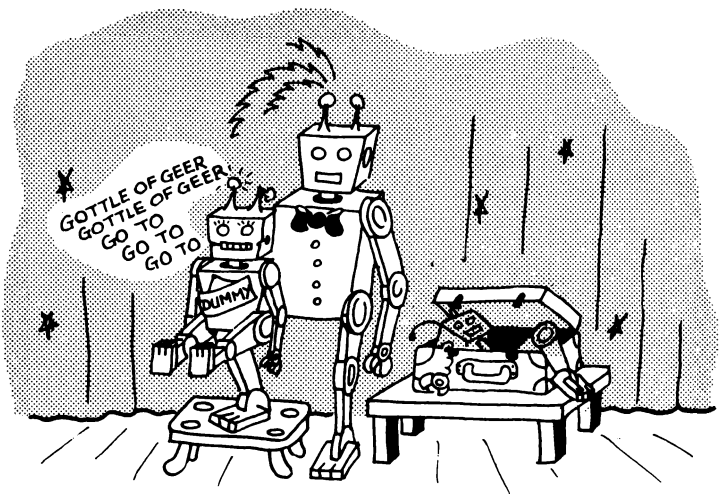
```
50  PRINT
60  PRINT
70  INPUT  "Entre um numero "; n
80  IF  n = 0  THEN GOTO 110
90  PRINT  "Nao, voce errou. Um polvo nao tem pernas, so tentaculos. Posso
contar a piada novamente?"
100 GOTO  20
110 PRINT  "Sim, posso contar a piada novamente?"
120 GOTO  20
130 PRINT  "Tudo bem, eu nao conto"
140 STOP
```

Execute o programa diversas vezes, mudando a entrada nas linhas 60 e 70. Como o programa funciona? Use o fluxograma a seguir.

Fluxograma



Uma outra técnica útil pode ser aprendida aqui. É o uso de um **valor fora de intervalo** para interromper um laço infinito, devido a um comando GOTO.



O próximo programa, que transforma notas de alunos num exame em porcentagens, ilustra essa técnica.

DIGITE:

```
 10  REM  CALCULANDO PORCENTAGENS
 20  PRINT  "Entre total de notas do exame ";
 30  INPUT  t
 40  PRINT  t
 50  PRINT
 60  PRINT  "Entre nota do aluno ";
 70  INPUT  m
 80  IF  m= -9999  THEN GOTO 140
 90  PRINT  m
100  LET  p=(m/t) * 100
110  PRINT  p; "%"
120  PRINT
130  GOTO 60
140  STOP
```

Execute o programa, e, depois de testá-lo bastante, entre −9999 e observe o que acontece.

Como você pode ver, se o valor fora de intervalo −9999 é lido na linha 70, o computador não continua o programa, pulando para a linha 140 e parando. Se a linha 80 não existisse, o laço continuaria indefinidamente, por causa do comando GOTO na linha 130. O valor usado não precisa ser −9999, bastando ser qualquer valor fora do intervalo normal de variação de notas de alunos. Seria útil acrescentar uma linha, indicando ao usuário como parar o programa. Por exemplo:

```
 65  PRINT  "ou −9999 se você quer parar o programa"
```

O valor usado nesta técnica não precisa ser um número, podendo ser uma palavra, se estivermos lidando com variáveis string. Por exemplo:

```
10  REM  LISTA DE NOMES
20  INPUT  "Entre o nome de uma pessoa "; n$
30  IF  n.$="cenoura"  THEN GOTO  60
40  PRINT  n$
50  GOTO  20
60  STOP
```

A técnica do valor fora do intervalo não é a única para parar um laço infinito. Você aprenderá outro método na Unidade 19.

Atividades

5. Escreva um programa que continuamente pergunte nomes de cidades da Inglaterra e imprima-os numa lista. Inclua um valor fora do intervalo para parar o programa. Teste seu programa no computador.

6. Desenhe um fluxograma e escreva um programa que testa tabuada (tal como; 6 × 7 = 42, 4 × 5 = 20 etc.). Números usando variáveis numéricas devem ser entrados para fazer uma soma, e a soma impressa. Uma resposta deve então ser dada e a pessoa avisada se errou ou acertou. Se a resposta estiver errada, é permitida à pessoa uma nova tentativa, não sendo a resposta correta impressa nesse momento. Se a pessoa acertar, ganha direito a uma nova rodada, e assim o computador deve voltar ao início do programa onde outra soma é armada. Teste seu programa no computador.

Unidade 18:

Mais Comparações

Além de igual (=) e diferente (<>), há outras formas de comparar informação num comando IF-THEN.

Significado	**Símbolo**	**Tecla**
maior que	>	shift T
menor que	<	shift R
maior ou igual	> =	shift E
menor ou igual	< =	shift Q

Estes quatro novos símbolos são usados nos exemplos abaixo.

Exemplos:

(1) 8 > 4? Sim.
(2) 8 < 4? Não.
(3) 8 > = 8? Sim.
(4) 8 > = 7? Sim.
(5) 8 > = 9? Não.
(6) 8 < = 8? Sim.
(7) 8 < = 6? Não.
(8) 8 < = 9? Sim.

Atividades

1. Responda sim ou não.

(a) 10 > 12?
(b) 12 < 15?
(c) 24 > = 24?
(d) 32 > = 31?
(e) 15 > 9?
(f) 17 < = 17?
(g) 102 > = 105?
(h) 33 < = 43?
(i) 47 = 52?
(j) 53 < > 54?
(k) 3 * x = 15? (x = 5)
(l) 4 + 5 > 9?
(m) 6 + y < 19? (y = 6)
(n) a − b < = 6?
(a = 10, b = 4)
(o) r * s > = 7?
(r = 3, s = 2)
(p) (10 + 2)/6 > 0?
(q) (w + z) * 2 > 40?
(w = 5, z = 10)
(r) n * n < > 64? (n = 8)
(s) c + d + e < 24?
(c = 7, d = 9, e = 11)
(t) (f + g) * (10−3) < = 56?
(f = 2, g = 6)

2. Complete os exemplos a seguir com o símbolo correto (**não** use 'diferente' (< >) ou 'igual' (=)).

Exemplo 13 _____ 12 deve ficar 13 > 12
5 _____ 5 pode ser 5 < = 5 ou 5 > = 5
12 _____ 13 deve ficar 12 < 13

(a) 16 _____ 23
(b) 14 _____ 10
(c) 10 _____ 10
(d) 5 + 6 _____ 11 − 3
(e) 3 * 7 _____ 13 + 8
(f) x _____ y
(x = 0, y = 1)
(g) x/2 _____ y + 4
(x = 6, y = 2)
(h) x * x _____ y/2
(x = 10, y = 150)
(i) (a + b) * 3 _____ a + b * 3
(a = 3, b = 5)
(j) a + b − c _____ a + c − 10
(a = 1, b = 6, c = 4)

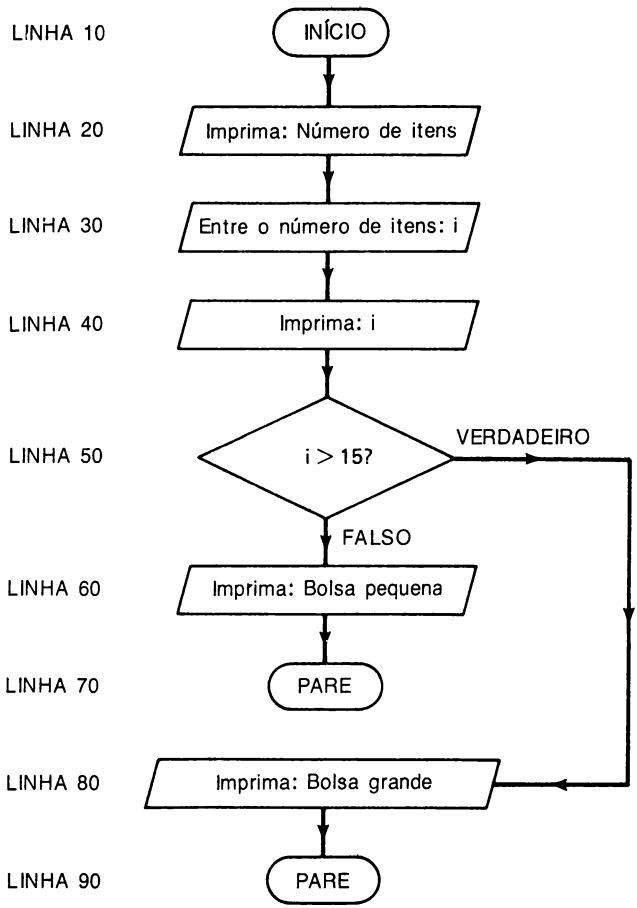
Estudando os programas e fluxogramas a seguir é possível ver como usar esses símbolos. Você verá que eles são incluídos nos comandos IF-THEN, como na unidade anterior.

DIGITE:

```
10  REM  O TAMANHO DE UMA BOLSA DE COMPRAS
20  PRINT  "O numero de itens comprados e ";
30  INPUT  i
40  PRINT  i
50  IF  i > 15  THEN GOTO 80
60  PRINT  "Bolsa pequena"
70  STOP
80  PRINT  "Bolsa grande"
90  STOP
```

Execute o programa diversas vezes com números diferentes e estude o fluxograma a seguir:

Fluxograma do programa anterior:



O próximo é sobre entrar com os nomes de três crianças e os trocados que cada uma recebeu. Depois, o programa calcula a média de dinheiro e compara com 6 e 2 cruzados e imprime a mensagem apropriada.

DIGITE:

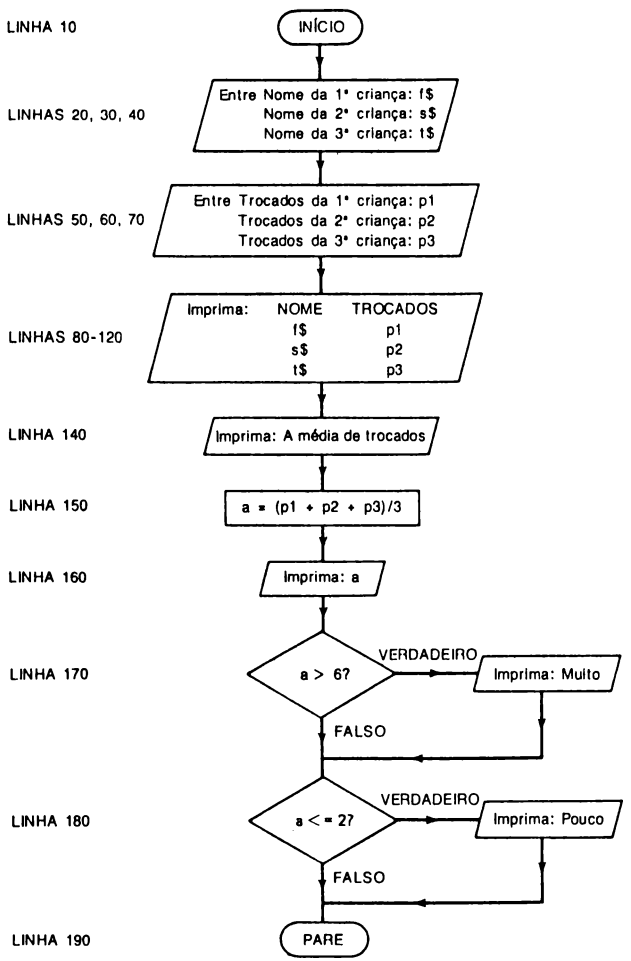
```
 10  REM  TROCADOS
 20  INPUT  "Entre o nome da primeira crianca"; f$
 30  INPUT  "Entre o nome da segunda crianca"; s$
 40  INPUT  "Entre o nome da terceira crianca"; t$
 50  INPUT  "Entre a quantidade de trocados da primeira"; p1
 60  INPUT  "Entre a quantidade de trocados da segunda"; p2
 70  INPUT  "Entre a quantidade de trocados da terceira"; p3
 80  PRINT  "NOME", "TROCADOS"
 90  PRINT  TAB 16; "EM CRUZADOS"
100  PRINT  f$, p1
```

```
110  PRINT  s$, p2
120  PRINT  t$, p3
130  PRINT
140  PRINT  "Media de trocados e";
150  LET  a = (p1 + p2 + p3)/3
160  PRINT  a
170  IF  a > 6  THEN PRINT "Muito dinheiro"
180  IF  a < = 2  THEN PRINT "Pouco dinheiro"
190  STOP
```

Execute o programa diversas vezes, com diversas quantidades de dinheiro.
Veja o fluxograma a seguir.

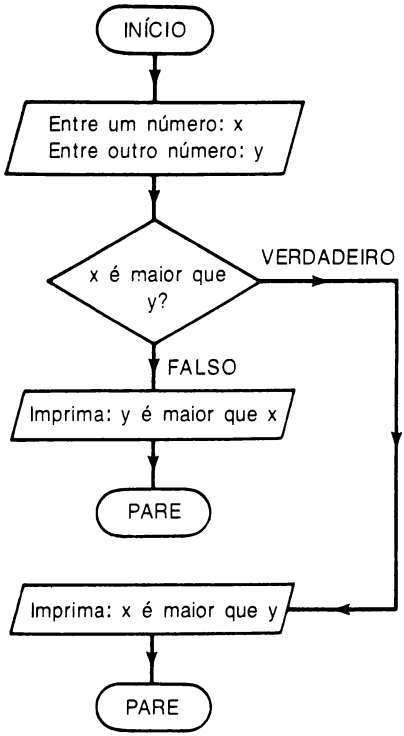
Da linha 20 à 120 apenas recebe as informações necessárias, e as imprime numa tabela cujas colunas são "NOME" e "TROCADOS EM CRUZADOS". A linha 150 calcula a média, e as linhas 170 e 180 a comparam com Cz$ 6,00 e Cz$ 2,00 respectivamente.

Fluxograma do programa anterior

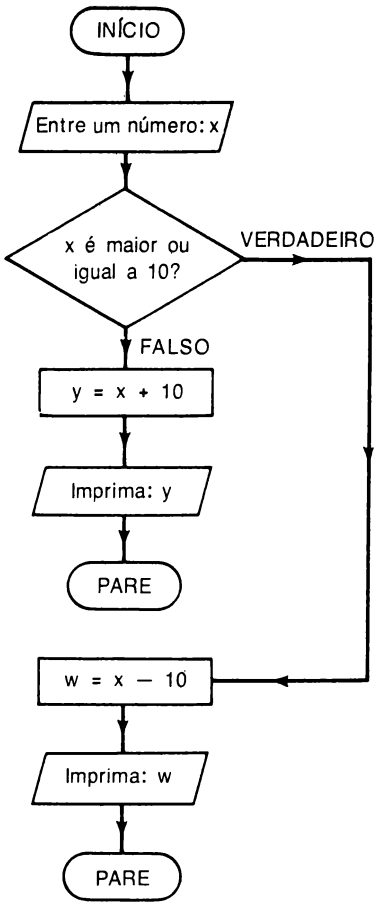


Atividades

3. Escreva um programa para este fluxograma.



4. Escreva um programa para este fiuxograma.



5. Um vendedor recebe um salário semanal de Cz$ 650,00. Se ele vender bens de valor igual a Cz$ 2000,00 ou mais, ganha um bônus extra de Cz$ 100,00. O programa a seguir calcula o salário final usando as seguintes variáveis:

s = salário de Cz$ 650,00
b = bônus de Cz$ 100,00
x = valor dos bens vendidos

Desenhe um fluxograma para este programa:

```
10  REM  SALARIO DE VENDEDOR
20  LET  s = 65
30  LET  b = 10
```

```
 40  PRINT  "Valor vendido pelo vendedor esta semana "
 50  INPUT  "Entre o valor dos bens"; x
 60  PRINT  x; "cruzados"
 70  IF  x > = 200  THEN GOTO 110
 80  PRINT  "Sem bônus adicional. Salário final "; s; " cruzados"
 90  GOTO  120
100  LET  s = s + b
110  PRINT  "Com bonus. Salario final "; s; " cruzados"
120  STOP
```

Liste as variáveis, desenhe os fluxogramas e escreva os programas de 6 a 11. Teste cada programa no computador.

6. Entre um ano (por exemplo, 1983). Se for maior ou igual a 2000, imprima "Próximo século". Se não for, imprima "este século".

7. Entre dois números separados. Se o produto deles for maior que 81, imprima "Muito grande". Se for menor, imprima "Muito pequeno". Se for igual a 81, imprima "Exatamente".

8. Se um garoto ganha Cz$ 50,00 por semana, seu pai dá-lhe mais Cz$ 10,00. Se ele ganha Cz$ 70,00 ou mais, ele dá Cz$ 15,00 para a mãe. Entre o salário do garoto, calcule e imprima o que fica com ele no fim da semana.

9. Um carro leva 33 litros de combustível. Um litro de combustível custa Cz$ 4,50. Um carro gasta 1 litro de combustível para andar 14 km. Um homem põe uma certa quantidade de combustível no carro; calcule o custo e a distância que ele pode percorrer, e imprima. Se ele pode andar 380 km sem reencher o tanque, ele passará o fim de semana em São Paulo, caso contrário irá para o Rio de Janeiro. Imprima onde ele irá.

10. No forno de um padeiro cabem 15 pães, que levam 52 minutos p/assar. Ele enche o forno 4 vezes por dia. Calcule o nº de pães que ele assa por dia e o tempo que leva em horas. Imprima essa informação. O pão é vendido no supermercado local. Se todo o pão for vendido, uma mensagem é enviada ao padeiro, para ele assar mais pães. Imprima essa mensagem se necessário.

11. Um leiteiro vende leite a vários blocos de apartamentos. 18 famílias compram 3 litros por dia, 17 famílias levam 2 litros e 9 famílias ficam com 4 litros por dia. 15 famílias não têm quantidade fixa por dia. Uma caixa de leite contém 12 litros. Calcule o número de litros que ele vende a cada dia e o número de caixas que ele tem de trazer. Imprima essa informação. Se mais de 13 caixas forem necessárias ele precisará voltar ao depósito, caso contrário pode ir para casa. Imprima para onde ele vai.

Algumas vezes precisamos comparar duas ou mais informações no mesmo comando IF-THEN. Por exemplo, "se uma coisa é verdadeira e outra também, então faça isso e aquilo", em BASIC, ficaria:

```
IF  y = 10 AND x = 20 THEN GOTO 200
                          ou
IF  a$ = "John" AND b$ = "McEnroe" THEN PRINT "Craque do tenis"
```

A única palavra que adicionamos ao comando foi AND (shift Y).

DIGITE:

```
10  REM  Usando AND
20  INPUT  "Entre um numero entre 1 e 100 "; n
```

```
 30  INPUT  "Entre outro numero entre 1 e 100 "; n2
 40  IF  n > 50 AND n2 < 50  THEN GOTO 90
 50  PRINT  n + n2
 60  LET  c = 5
 70  PRINT  n * c
 80  STOP
 90  PRINT  n – n2
100  STOP

RUN. Entre 20 e 70
O resultado é:  90
                100
RUN. Entre 70 e 20
O resultado é: 50
RUN. Entre 70 e 70
O resultado é: 140
                350
RUN. Entre 20 e 20
O resultado é:  40
                100
```

A linha 40 testa duas condições, tendo ambas de ser verdadeiras, antes que o computador execute o resto na instrução. Assim, quando n = 20 e n2 = 70, 'n' não é maior que 50 e n2 não é menor que 50, daí o computador ignorar o comando GOTO, e ir para as linhas de 50 a 80. Quando n = 70 e n2 = 20, 'n' é maior que 50 e n2 é menor que 50, logo o GOTO é seguido e o computador salta para a linha 90. Quando n = 70 e n2 = 70, 'n' é maior que 50, mas n2 não é menor que 50, logo uma das condições é verdadeira, a outra é falsa, e o AND produz o valor falso, por isso o computador continua na linha 50 a execução. Quando n é 20 e n2 é 20, 'n' não é maior que 50, apesar de n2 ser menor que 50; assim, novamente, o resultado do AND é falso e a próxima instrução a ser executada é a linha 50.

Atividades

12. Usando o programa anterior, dê os resultados, se os seguintes números fossem entrados:

(a) n = 10, n2 = 20
(b) n = 80, n2 = 40
(c) n = 80, n2 = 60
(d) n = 20, n2 = 10

Teste suas respostas no computador.

AND pode ser usado igualmente bem com as séries de variáveis a serem vistas no próximo programa.

DIGITE:

```
10  REM  Usando AND
20  INPUT  "Entre o primeiro nome de um astro do tenis "; c$
30  INPUT  "Entre o sobrenome do mesmo astro "; s$
40  IF  c$ = "John" AND s$ = "McEnroe"  THEN PRINT "Melhor jogador
do mundo"
```

```
50 IF c$ = "Jimmy" AND s$ = "Connors" THEN PRINT "Segundo
melhor jogador do mundo"
60 STOP
```

RUN. Entre John e McEnroe
O resultado é: Melhor jogador do mundo
RUN. Entre Jimmy Connors
O resultado é: Segundo melhor jogador do mundo
RUN. Entre John e Connors
O resultado é: nenhuma mensagem é impressa.

O princípio de funcionamento deste programa é exatamente o mesmo do anterior. A mensagem será impressa apenas se ambas as condições são verdadeiras.

A palavra OR (shift U) funciona parecido, só que apenas uma das condições precisa ser verdadeira.

DIGITE:

```
10 REM Usando OR
20 INPUT "Entre sua altura em centimetros "; h
30 INPUT "Entre o tamanho de seus sapatos "; s
40 IF h > 165 OR s > 6 THEN PRINT "Muito grande. Voce precisa
diminuir": GOTO 60
50 PRINT "Muito pequeno. Voce precisa crescer"
60 STOP
```

RUN. Entre 155 e 35
O resultado é: "Muito pequeno — Voce precisa crescer"
RUN. Entre 266 e 35
O resultado é: "Muito grande — Voce precisa diminuir"
RUN. Entre 155 e 44
O resultado é: "Muito grande — Voce precisa diminuir."

Quando h = 155 e s = 35 nenhuma das condições era verdadeira e por isso o computador ignorou o resto da linha; mas quando h = 266 e s = 35, uma das condições era verdadeira (266 é maior que 165), daí a mensagem "Muito grande você precisa diminuir". O mesmo acontece com 155 e s = 44.

Atividades

13. Escreva um programa que lê um número que é testado, para ver se é maior ou igual a 100, mas menor ou igual a 200. Se o número satisfaz a condição, então a mensagem "Número entre 100 e 200" deve ser impressa. Caso contrário, a mensagem "Número não está entre 100 e 200" deve ser impressa.

14. Escreva o resultado deste programa se os seguintes números forem entrados na linha 30: (a) n = 40, (b) n = 60, (c) n = 80.

```
10 LET x = 50
20 LET y = 70
30 INPUT "Entre um numero "; n
40 IF n > x AND n < y THEN GOTO 70
```

```
50  PRINT  n + n
60  STOP
70  PRINT  x + y + n
80  STOP
```

15. Escreva o resultado deste programa, se as seguintes idades forem entradas na linha 30: (a) a = 13, (b) a = 21, (c) a = 17, (d) a = 6, (e) a = 19.

```
10  LET  y = 19
20  LET  x = 13
30  INPUT  "Entre sua idade "; a
40  IF  a > = 13 AND a < = 19  THEN PRINT  "Adolescente"
50  IF  a < 13  THEN PRINT  "Criança"
60  IF  a > 19  THEN PRINT  "Adulto"
70  STOP
```

16. Escreva o resultado deste programa se:
(a) w$ = quente, r$ = seco (b) w$ = morno, r$ = úmido
(c) w$ = frio, r$ = seco (d) w$ = morno, r$ = seco

```
10  LET  a$ = "quente"
20  LET  b$ = "frio"
30  LET  c$ = "umido"
40  LET  d$ = "seco"
50  LET  e$ = "morno"
60  INPUT  "O tempo esta quente, morno ou frio? "; w$
70  INPUT  "O tempo esta úmido ou seco? "; r$
80  IF  w$ = a$ OR w$ = b$ OR r$ = c$  THEN PRINT  "Sem passeios
hoje": GOTO 100
90  PRINT "Vamos passear"
100  STOP
```

Unidade 19:

Incrementando um Contador

Como você já viu, um computador pode ser forçado a executar repetidas vezes um programa, pelo uso do GOTO. Algumas vezes não é fácil parar esse tipo de laço, e não é particularmente útil. No entanto, ao incrementarmos um contador, podemos fazer o computador executar um programa um certo número de vezes e então parar. Isso é ilustrado no programa a seguir, que é executado cinco vezes.

DIGITE:

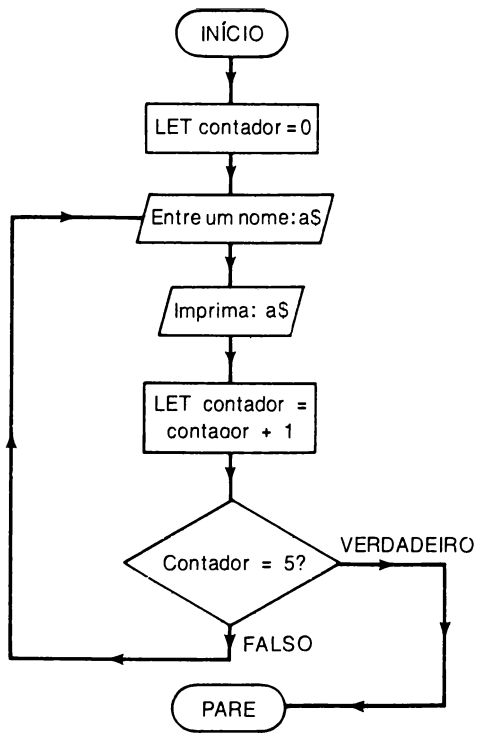
```
10  REM   CONTANDO
20  LET   contador = 0
30  INPUT   "Entre o nome de um carro "; a$
40  PRINT   a$
50  LET   contador = contador + 1
60  IF   contador = 5   THEN GOTO 80
70  GOTO   30
80  STOP
```

RUN. **Não remova do computador.**

A linha 20 inicializa a variável contador com 0. Um nome é lido e impresso. A linha 50 soma 1 ao contador, que agora vale 1. A linha 60 testa se o contador já é igual a 5; como não é, o computador ignora o resto da linha e continua na linha 70, que o desvia de volta à linha 30. Assim, o programa é executado uma segunda vez, e o contador passa a valer 2, que ainda não é igual a 5; logo o computador retorna à linha 30. Isso continua até o contador ser igual a 5, e aí o computador salta para a linha 80 e pára.

Pergunta: Por que a linha 70 diz GOTO 30 e não pára a linha 20?

Fluxograma



Não é necessário estabelecer o número de execuções quando o programa é escrito. Isso pode ser lido pelo programa durante a execução. Modifique o programa anterior da seguinte forma:

```
INCLUA: 15   INPUT "Entre o número de vezes que você quer executar o programa ";
             n
ALTERE: 60   IF  contador = n  THEN GOTO 80
```

RUN. Entre um número não muito grande, caso contrário você terá que dar muitos nomes.

Para economizar tempo de digitação, a variável contador é geralmente abreviada por "c", o que passaremos a fazer.

Um contador não é usado apenas para contar o número de vezes que um programa é executado, podendo ser útil num programa que conte o número de respostas corretas e incorretas. Veja o exemplo a seguir.

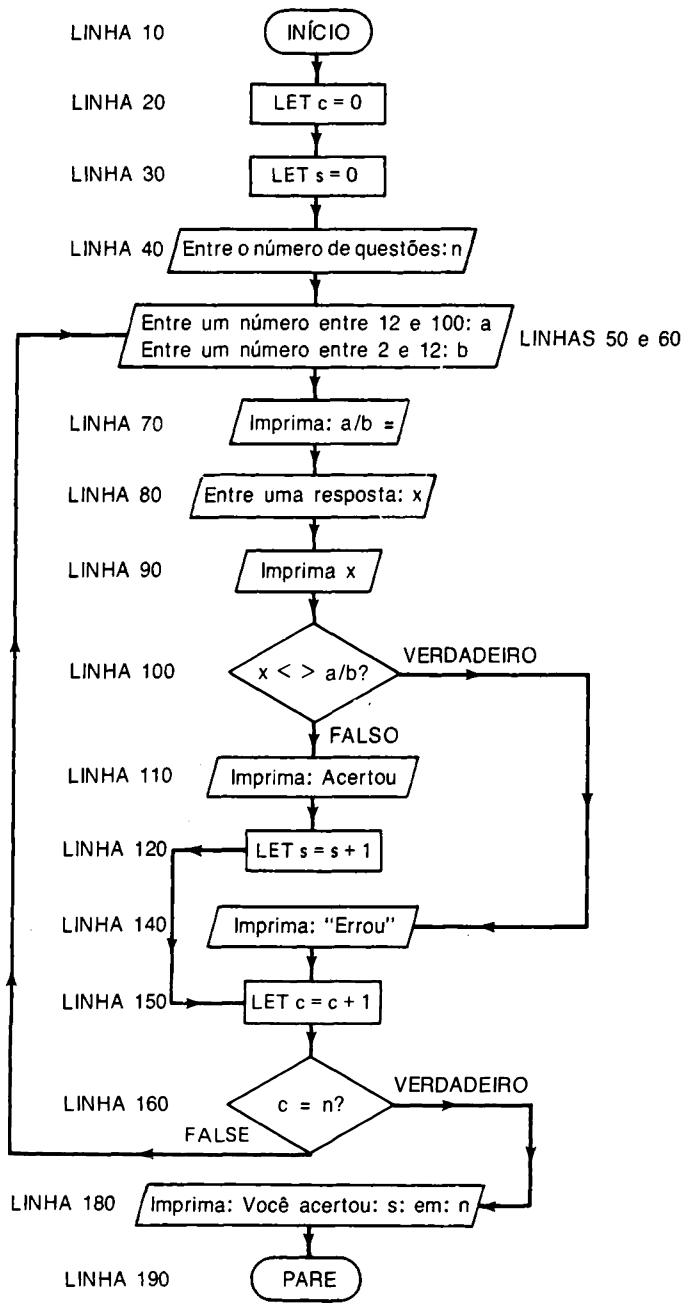
DIGITE:

```
 10  REM   PRATICANDO DIVISAO
 20  LET   c = 0
 30  LET   s = 0
 40  INPUT  "Entre o numero de questoes "; n
 50  INPUT  "Entre um numero entre 12 e 100 "; a
 60  INPUT  "Entre um numero entre 2 e 12 "; b
 70  PRINT  a; "/"; b; "=";
 80  INPUT  "Entre uma resposta "; x
 90  PRINT  x
100  IF   x < > a/b   THEN GOTO 140
110  PRINT  "Acertou"
120  LET   s = s + 1
130  GOTO   150
140  PRINT  "Errou. O certo e "; a/b
150  LET   c = c + 1
160  IF   c = n   THEN GOTO 180
170  GOTO   50
180  PRINT  "Voce acertou "; s; " em "; n
190  STOP
```

RUN

As linhas 20, 40, 150 e 160 inicializam e lidam com o contador, que conta o número de vezes que o programa é executado. A linha 30 inicializa o número de respostas certas. A variável 's' começa em zero, como nenhuma questão foi respondida. Se a primeira divisão for efetuada corretamente, então 1 é somado a 's' na linha 120, e assim 's' passa a ser 1. Se a divisão for efetuada erradamente, então a linha 100 manda o computador para a linha 140, pulando a linha 120, onde 's' seria incrementado de 1. Essa seqüência de eventos é repetida a cada vez que uma nova divisão é efetuada, e a variável 's' é incrementada a cada vez que a divisão for feita corretamente. Quando o programa tiver sido executado o número desejado de vezes, a linha 180 imprime o resultado final. O fluxograma seguinte ajudará você a entender esse programa.

Fluxograma


LINHA 10
INÍCIO
LINHA 20
LET c = 0
LINHA 30
LET s = 0
LINHA 40
Entre o número de questões: n
Entre um número entre 12 e 100: a
Entre um número entre 2 e 12: b
LINHAS 50 e 60
LINHA 70
Imprima: a/b =
LINHA 80
Entre uma resposta: x
LINHA 90
Imprima x
LINHA 100
x < > a/b?
VERDADEIRO
FALSO
LINHA 110
Imprima: Acertou
LINHA 120
LET s = s + 1
LINHA 140
Imprima: "Errou"
LINHA 150
LET c = c + 1
LINHA 160
c = n?
VERDADEIRO
FALSE
LINHA 180
Imprima: Você acertou: s: em: n
LINHA 190
PARE

Atividades

Lembre-se de testar seus programas no computador.

1. Escreva um programa que pergunta pelos nomes de seis flores e imprima uma lista delas.

2. Modifique o programa que você escreveu para o item 1, de tal forma que o número de flores a serem listadas seja decidido durante a execução do programa.

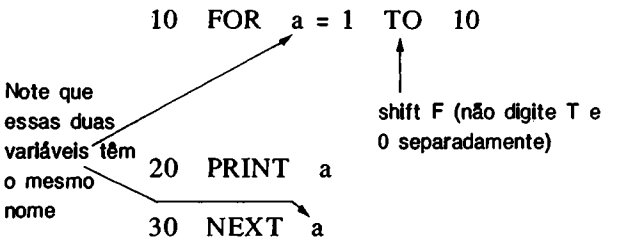
3. Desenhe o fluxograma e escreva um programa em que uma pessoa escolhe um número entre 0 e 20. Se escolher errado, então uma nova tentativa deve ser feita, até ela dar a resposta certa. O computador deve guardar e imprimir o número de tentativas feito. Assegure-se de que a pessoa que irá tentar não veja o número a ser adivinhado sendo entrado!

Unidade 20:

Laços FOR-NEXT

Usar laços FOR-NEXT é uma outra maneira de controlar o número de vezes que um programa ou parte de um programa é executado. FOR é uma palavra-chave encontrada na tecla F e NEXT é outra palavra-chave na tecla N; ambas só podem ser usadas se o computador estiver no modo **palavra-chave K**.

DIGITE:



RUN

Como você pode ver, o resultado desse programa é uma lista com os números de 1 a 10. Na linha 10, o comando FOR faz com que o contador comece em 1 e acabe em 10. A linha 20 imprime o valor de 'a', que na primeira volta é 1. A linha 30 manda o computador de volta ao comando FOR para pegar o próximo (NEXT) valor de 'a', que será 2. Esse laço é repetido até que 'a' tenha o valor 10, e aí ele pára automaticamente. O laço é executado 10 vezes.

DIGITE:

```
10  FOR  b = 11  TO  20
20  PRINT  b
30  NEXT  b

RUN
```

Esse programa também é executado 10 vezes, mas a diferença é que o valor de 'b' começa com 11 e termina em 20.

DIGITE:

```
10  FOR  c = 1  TO  6
20  PRINT  "Ola"
30  NEXT  c

RUN
```

Esse laço FOR-NEXT será executado 6 vezes. A linha 20 não imprime o valor de 'c', mas a palavra 'Ola', e ela aparecerá 6 vezes na tela. Você percebe que quaisquer linhas colocadas entre o FOR e o NEXT são parte do laço, e serão executadas cada vez que o computador passar pelo laço.

Atividades

1. Estude os programas a seguir e escreva o que deveria aparecer na tela se eles fossem executados.

(a)
```
10  FOR  x = 1  TO  12
20  PRINT  x
30  NEXT  x
```

(b)
```
40  FOR  y = 4  TO  16
50  PRINT  y
60  NEXT  y
```

(c)
```
70  FOR  w = 1  TO  4
80  PRINT  "Adeus"
90  NEXT  w
```

(d)
```
100  FOR  c = 1  TO  3
110  PRINT  c
120  PRINT  "Isto e um laco"
130  NEXT  c
```

Se você deseja espaçar as linhas impressas por dois comandos, você pode usar um laço FOR-NEXT para criar o espaço que quiser. Veja o programa abaixo.

DIGITE:

```
10  PRINT  "Eu estou no topo da tela"
20  FOR  a = 1  TO  9
```

```
 30  PRINT
 40  NEXT  a
 50  PRINT  "Eu estou no meio da tela"
 60  FOR  b = 1  TO  9
 70  PRINT
 80  NEXT  b
 90  PRINT  "Eu estou no fim da tela"
100  STOP

RUN
```

Nesse programa, as linhas de 20 a 40 criam um espaço com nove linhas vazias impressas. O mesmo acontece de novo, com o laço das linhas 60 a 80. Isso economiza o esforço de digitar nove comandos PRINT. Por exemplo:

```
10  PRINT  "Eu estou no topo da tela"
20  PRINT
30  PRINT
40  PRINT
    etc.
```

Um laço é criado usando uma variável numérica, tal como

FOR a (variável numérica) = 1 to 10

mas outras variáveis podem ser utilizadas no comando. Veja abaixo.

DIGITE:

```
10  INPUT  "Entre o numero por vezes que
voce quer que o laco seja repetido ";n
20  FOR  x = 1  TO  n
30  PRINT  x
40  NEXT  x
```

Execute várias vezes, com números diferentes, e observe os resultados. Se você entrar com um número maior que 22, a tela ficará cheia e você receberá um pedido de "scroll".

ADICIONE ao programa:

```
 5  INPUT  "Entre o numero em que voce deseja que comece o laco "; s
```

ALTERE a linha 10:

```
10  INPUT  "Entre o numero em que voce
deseja que termine o laco "; n
```

ALTERE a linha 20:

```
20  FOR  x = s TO n
```

Execute o programa, tomando o cuidado de certificar-se de que o primeiro número seja menor que o segundo.

Esse último programa mostra que o FOR por inteiro pode ser composto de variáveis numéricas. Isso permite ao programador entrar números diferentes enquanto o programa estiver sendo executado, de tal forma que o laço possa ser repetido um número qualquer de vezes.

Normalmente, o incremento no FOR-NEXT é de 1, mas pode ser de dois, três, quatro etc., usando a palavra STEP (shift D). O próximo programa ilustra isso.

DIGITE:

```
10  FOR  x = 1 TO 10 STEP 2
20  PRINT  x
30  NEXT  x
```

RUN

O resultado é:

```
1   ↑
3   Apenas os números
5   ímpares são impressos.
7   Note que apesar do laço dever
9   terminar em 10, só é impresso até 9.
    (O próximo já seria 11.)
```

ALTERE a linha 10:

```
10  FOR  x = 3 TO 21 STEP 3
```

RUN

O resultado será:

```
3   ↑
6
9   Esse programa começa em 3 e
12  contado de 3 em 3 até chegar a 21.
15
18
21
```

ALTERE a linha 10:

```
10  FOR  x = 10 TO 1 STEP – 1
```

RUN

O resultado será:

```
10↑
 9
 8 Esse programa conta
 7 decrescentemente do
 6 maior para o menor.
 5 Não podemos escrever
 4 FOR  x = 1 TO 10 STEP – 1
 3
 2
 1
```

ALTERE a linha 10:

```
10  FOR  x = 20 TO − 5 STEP − 5
```

RUN o programa.

O resultado é:

```
 20   ↑
 15   O laço começa em 20 e
 10   decresce de 5 em 5
  5   até − 5.
  0
− 5
```

ALTERE a linha 10:

```
10  FOR  x = − 10 TO − 20 STEP − 2
```

RUN

O resultado é:

```
− 10  ↑
− 12
− 14  Esse programa é parecido,
− 16  mas lembre-se que − 10 é
− 18  maior que − 20
− 20
```

Atividades

2. Estude os programas abaixo e escreva o resultado de cada.

(a)
```
10  FOR  x = 2 TO 12 STEP 2
20  PRINT  x
30  NEXT  x
```

(b)
```
10  FOR  x = 15 TO 25 STEP 4
20  PRINT  x
30  NEXT  x
```

(c)
```
10  FOR  x = 0 TO 100 STEP 20
20  PRINT  x
30  NEXT  x
```

(d)
```
10  FOR  x = 5 TO − 25 STEP − 25
20  PRINT  x
30  NEXT  x
```

(e)
```
10  FOR  x = − 2 TO − 12 STEP − 3
20  PRINT  x
30  NEXT  x
```

(f)
```
10  FOR  x = − 10 TO 26 STEP 5
20  PRINT  x
30  PRINT  "Aumentando"
40  NEXT  x
```

3. Corrija os seguintes programas:

(a)
```
10  FOR  x = 1 TO 10
20  PRINT  x
30  NEXT  y
```

(b)
```
10  FOR  x = 1 TO 10 STEP – 1
20  PRINT  x
30  NEXT  x
```

(c)
```
10  FOR  x$ = 1 TO 6 STEP 2
20  PRINT  x$
30  NEXT  x$
```

(d)
```
10  FOR  x = x TO 10
20  PRINT  x
30  NEXT  x
```

(e)
```
10  FOR  x = 1 TO x
20  PRINT  x
30  NEXT  x
```

(f)
```
10  LET  x = 1 TO 10
20  PRINT  x
30  NEXT  x
```

Variações sobre um tema.

DIGITE:

```
10  FOR  x = 1 TO 6
20  LET  a = x + 5
30  PRINT  x, a
40  NEXT  x
```

RUN

O resultado é:

1	6
2	7
3	8
4	9
5	10
6	11

As linhas 10 e 40 fazem o laço ser repetido 6 vezes. A linha 20 introduz uma nova variável 'a' que é feito ser igual ao valor de 'x' mais 5. A linha 30 imprime as duas variáveis em duas colunas. Na primeira execução do laço, 'x' é igual a 1 e 'a' igual a 6; na segunda vez 'x' é 2, por isso 'a' é 7. Isso continua até o último valor de 'x' ser obtido.

DIGITE:

```
10  FOR  x = 3 TO 12 STEP 3
20  PRINT  x, x * x
30  NEXT  x
```

RUN

O resultado é:

3	9
6	36
9	81
12	144

Explique como esse programa funciona.

Atividades

4. Estude os programas e escreva os resultados:

```
(a) 10  FOR   x = 2 TO 4
    20  PRINT  x, x + 4
    30  NEXT   x

(b) 10  FOR   x = 10 TO 16 STEP 2
    20  LET   a = x + 2
    30  PRINT  x. a
    40  NEXT   x

(c) 10  FOR   x = 12 TO 3 STEP – 3
    20  LET   a = 10
    30  PRINT  x. a
    40  NEXT   x

(d) 10  FOR   x = 100 TO 200 STEP 25
    20  LET   a = x – 10
    30  LET   b = x + 10
    40  PRINT  x:    :a: :b
    50  NEXT   x
```

Usando um laço FOR-NEXT para imprimir tabuadas de multiplicação:

DIGITE:

```
10  REM   TABUADAS DE MULTIPLICACAO
20  FOR   x = 1 TO 12
30  PRINT  x; "vezes 8 ="; x * 8
40  NEXT   x
```

RUN

Execute o programa. O resultado é a tabuada de multiplicação por 8. Esse programa pode ser alterado para dar qualquer tabuada.

INCLUA no programa:

```
15  INPUT  "Entre o número da tabuada que você quer "; n
```

ALTERE a linha 30:

```
30  PRINT  x; "vezes"; n; "="; x * n
```

RUN

Se o número 5 é entrado, a tabela de multiplicação por 5 será impressa; se for o 9, a tabela de multiplicação por 9 será impressa. E assim por diante.

Atividades

Escreva os programas e teste no computador.

5. Imprima todos os números de 13 a 23.

6. Imprima todos os números ímpares de 3 a 37.
7. Imprima os números de 5 a 35, de 5 em 5.
8. Imprima todos os números pares de 10 a −6.
9. Imprima a tabuada de multiplicação por 7.
10. Imprima três sentenças separadas por cinco linhas em branco.
11. Imprima a palavra 'Spectrum' oito vezes.
12. Escreva um programa que imprima todos os anos bissextos deste século. A cada quatro anos ocorre um ano bissexto, e o primeiro neste século foi em 1904.
13. Imprima o valor de x que começa em 10 até 20. Imprima numa coluna separada o valor de x + 3.

Os próximos dois programas mostram que quaisquer instruções podem ser colocadas dentro do laço FOR-NEXT. Podem ser comandos de entrada e comandos de saída, cálculos ou ambos.

DIGITE:

```
10  FOR  x = 1 TO 5
20  INPUT  "Entre o nome de um pais "; c$
30  INPUT  "Entre a população do pais "; p
40  PRINT  "PAIS", "POPULACAO"
50  PRINT  c$, p
60  PRINT
70  NEXT  x
80  STOP
```

RUN. Use a informação a seguir:

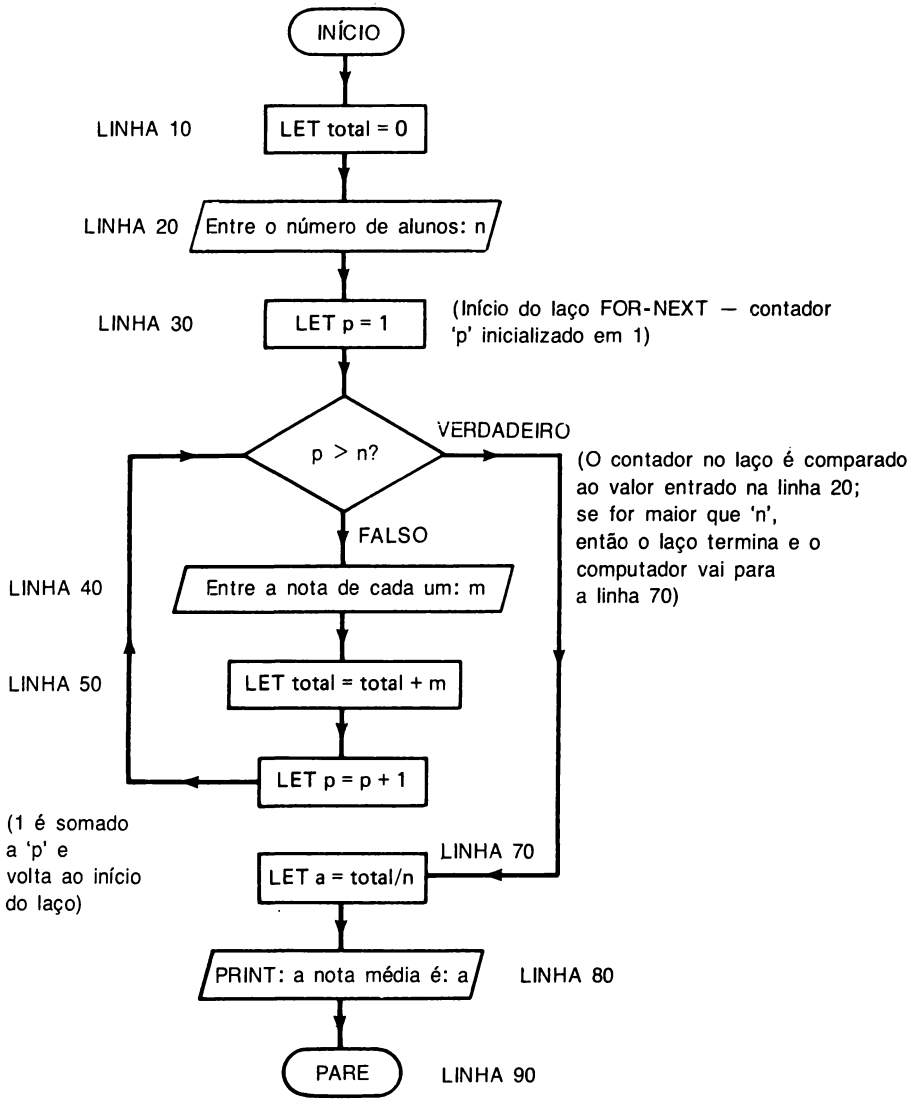
País	**População**	
Dinamarca	5 000 000	(Não se colocam vírgulas
Suécia	8 000 000	entre as séries de zeros.
Grã-Bretanha	56 000 000	Você sabe por que?)
França	53 000 000	
Bélgica	10 000 000	

DIGITE:

```
10  LET  total = 0
20  INPUT  "Entre o numero de alunos "; n
30  FOR  p = 1 TO n
40  INPUT  "Entre as notas de cada um "; m
50  LET  total = total + m
60  NEXT  p
70  LET  a = total/n
80  PRINT  "A nota media e "; a
90  STOP
```

RUN. Varie o número de alunos, na linha 20, e liste as notas.

Fluxograma do programa anterior:



Como você pode ver, o laço FOR-NEXT aparece no fluxograma como um contador comum, e parecido com os diagramas da Unidade 19.

Atividades

Escreva os programas e teste no computador.

14. Entre os nomes de 6 pessoas e suas idades. Imprima em duas colunas com cabeçalho.

15. Usando a informação abaixo entre e imprima os nomes de 4 carros e seus preços. Calcule e imprima o preço médio.

Carro	Preço
Ford Escort	Cz$ 90 000,00
VW Santana	Cz$ 100 000,00
GM Opala	Cz$ 100 000,00
Fiat Uno	Cz$ 65 000,00

16. As taxas de nascimento e morte por 1000 pessoas de vários países, são dadas abaixo. Entre essa informação e calcule a taxa de crescimento natural (nascimentos menos mortes). Imprima essa informação.

País	Taxa de Nascimentos	Taxa de Mortalidade
Índia	43	17
Japão	19	7
Inglaterra	15	12
U.S.A.	16	9
Malawi	49	25
Ghana	47	18
Bolívia	44	19

Unidade 21:

Comandos READ e DATA

Os comandos READ e DATA só podem ser usados quando o computador estiver no modo estendido (pressione CAPS SHIFT e SYMBOL SHIFT juntos). READ está sobre a tecla A e DATA, sobre a tecla D.

READ e DATA permitem uma forma alternativa de entrar informações num programa. No entanto, ao contrário do comando INPUT, onde a informação é alimentada no programa durante a execução, a informação usada no comando DATA é escrita quando o programa é feito, e não pode ser alterada quando o programa é executado. Nesse aspecto, se parece com o comando LET.

Um programa melhor ilustrado determina como esses novos comandos são usados.

DIGITE:

```
10  READ   a, b, c
20  DATA   10, 20, 30
30  PRINT  a, b, c
40  PRINT  b, c, a
50  STOP
```

```
RUN              NÃO APAGUE
O resultado é:10      20
              30
              20      30
              10
```

O comando READ dá o nome das variáveis; nesse caso as variáveis numéricas a, b, c. O comando DATA fornece os valores dessas variáveis; nesse exemplo, os números 10, 20 e 30.

Assim que o computador encontra a primeira variável num comando READ, ele procura a primeira informação no primeiro comando DATA. Assim, o valor 10 é armazenado em 'a'. Quando o computador encontra a segunda variável, procura pela segunda informação; assim 'b' toma o valor 20. E assim por diante para todas as variáveis.

A linha 30 prova que a variável 'a' contém 10, 'b' contém 20, e a variável 'c' contém o valor 30. O conteúdo dessas variáveis não muda, como mostra a linha 40, na qual muda-se a ordem de impressão.

A cada variável de um comando READ, deve corresponder uma informação no comando DATA. Tente o seguinte programa:

Modifique a linha 10.

```
10  READ  a, b, c, d
```

RUN

O resultado será a mensagem de erro:

```
E Out of DATA, 10 : 1
```

Assim, mesmo sem pedirmos para o computador imprimir o valor de 'd' na linha 30, o programa não será executado, pois o computador não conseguiu encontrar o valor correspondente a 'd'.

Variáveis string também podem ser usadas nos comandos READ e DATA.

DIGITE:

```
10  READ a$, b$, c$
20  DATA  "Feliz", "Ano", "Novo"
30  PRINT  a$; b$; c$
40  STOP
```

RUN

O resultado será:

```
FelizAnoNovo
```

Exatamente o mesmo que no programa anterior, só que usamos aspas no comando DATA, como seria de se esperar para variáveis string.

Note que em todos os programas, as variáveis do comando READ e as informações (strings ou números) do comando DATA são separadas por vírgulas (,). Isso é exigência da linguagem BASIC. Comandos READ podem ter ao mesmo tempo variáveis numéricas e string, respeitada a correspondência às informações do comando DATA.

DIGITE:

```
10  READ  a$, b$, c$, a
20  DATA  "Batalha", "de", "Hastings", 1066
30  PRINT  a$;" ";b$;" ";c$;" ";a
40  STOP

RUN
```

O resultado será: Batalha de Hastings 1066

(Note que incluímos espaços na linha 30 para tornar o resultado do PRINT mais legível.)

Pode haver mais de um comando READ e mais de um comando DATA em um programa.

DIGITE:

```
 10  REM  LIVROS
 20  READ  x, a$, b$, c$, d$
 30  READ  e$, f$, y, g$
 40  READ  h$
 50  DATA  2001, "Espaco", "Odisseia",
"Pequena", "Mulher"
 60  DATA  "Grandes", "Expectativas"
, 1984, "de Animais", "Fazenda"
 70  PRINT  "NOME DE LIVRO"
 80  PRINT  "----------"          (tracos, NAO brancos. Pressione shift J)
 90  PRINT  x; " "; a$; " "; b$
100  PRINT  c$; " "; d$
110  PRINT  e$; " "; f$
120  PRINT  y
130  PRINT  g$; " "; h$
140  STOP

RUN                              NÃO APAGUE
```

Como esperado, o resultado é uma lista de livros, mas há vários pontos a serem observados. Temos três READ, mas só dois DATA; o que não importa, desde que as variáveis e valores estejam em correspondência correta. As variáveis g$ (Animal) e h$ (Fazenda) foram impressas junto, mas não foram escritas no mesmo READ; o que importa é que elas correspondem às informações corretas no comando DATA. Esse programa pode ser reescrito, em ordem diferente, para provar que, uma vez que as informações correspondam, o resultado será o mesmo.

Altere as linhas 20 a 40 do programa anterior:

```
20  READ  g$, b$, y, e$, c$
30  READ  a$, h$, f$, d$
40  READ  x

RUN
```

Resultado: mensagem de erro:

C Nonsense in BASIC, 20 : 1

Isso porque as variáveis já não correspondem corretamente; por exemplo, 'g$' tenta receber o valor '2001', o que não é aceitável.

Agora altere as linhas 50 e 60

```
50 DATA "Animal", "Odisseia", 1984, "Grande", "Pequena"
60 DATA "Espaco", "Fazenda", "Expectativas", "Mulher", 2001
```

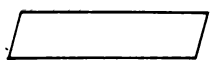
Deixe o resto do programa inalterado.

RUN

A lista de livros reaparece na tela, mostrando que se há compatibilidade de tipos, a ordem não importa.

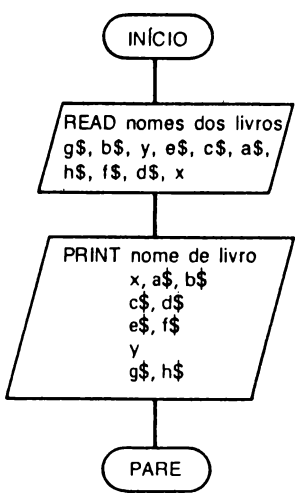
Comandos DATA geralmente seguem os comandos READ, mas podem ser postos no fim do programa.

Num fluxograma, representamos o READ da mesma forma que o INPUT:



apesar de não representarmos o DATA, que fica implícito. O fluxograma a seguir é do programa que acabamos de examinar.

Fluxograma



O programa a seguir mostra como um comando READ pode ser usado várias vezes, dentro de um laço GOTO.

DIGITE:

```
10 REM  READ NUM LACO GOTO
20 LET  c = 0
30 INPUT  "Entre um numero entre 0 e 12"; n
40 READ  a
50 DATA  3, 4, 7, 11, 1, 0, 6, 8, 9, 10
60 DATA  2, 0, 12, 7, 5, 6, 3, 2, 1, 12
70 DATA  1, 5, 3, 7, 9, 3, 11, 2, 4, – 1
```

```
 80  IF  a < 0 THEN GOTO 110
 90  IF  a = n THEN LET c = c + 1
100  GOTO  40
110  PRINT  "O numero de vezes que"; n; " aparece c "; c
120  STOP
```

RUN. Entre diferentes números, executando várias vezes.

O programa lê um número e calcula o número de vezes que ele aparece nos comandos DATA. Há apenas um comando READ, cuja variável muda várias vezes de valor, a cada vez que o laço é computado. Na primeira vez, 'a' recebe o valor 3, que é comparado com 'n' na linha 90. Se 'a' = 'n', então faz c = c + 1; caso contrário, vai para a linha 40, para passar a outro número do comando DATA. Assim por diante, até 'a' valor — 1. Aí, 'a' fica menor que zero, e na linha 80 ocorre o desvio que encerra o laço, passando à linha 110, que imprime o total de vezes que 'n' ocorre, e o programa termina.

Questão: O que aconteceria se retirássemos o valor — 1 do comando DATA? Remova o — 1 da linha 70 e verifique se você está certo.

READ e DATA podem ser usados em laços FOR-NEXT.

DIGITE:

```
10  FOR  x = 1 TO 7
20  READ  y
30  DATA  5, 10, 15, 20, 25, 30, 35
40  PRINT  y
50  NEXT  x
60  STOP
```

RUN

Esse programa passa pelo FOR-NEXT 'x' sete vezes. A cada vez 'y' assume um valor, começando em 5 e percorrendo os valores do comando DATA, até 35. Como o laço é executado sete vezes, deve haver sete informações no comando DATA, cada uma um número, pois 'y' é variável numérica.

Um laço similar pode ser usado para variáveis string.

DIGITE:

```
10  REM  NOMES
20  FOR  x = 1 TO 5
30  READ  y$
40  DATA  "Joao", "Maria", "Ana",
"Paulo", "Henrique"
50  PRINT  y$
60  NEXT  x
70  STOP
```

RUN

Esse programa imprime uma lista de cinco nomes. O comando DATA contém agora strings, logo a variável usada no READ deve também ser string.

Pode-se combinar variáveis string e numéricas num laço, desde que os tipos de dados sejam compatíveis.

Altere as linhas 30, 40 e 50

```
30  READ  y$, w
40  DATA  "Joao", 6, "Maria", 7, "Ana", 3,
"Paulo", 10, "Henrique", 8
50  PRINT  y$, w
```

Não altere o resto do programa.
RUN

O resultado é:

João	6
Maria	7
Ana	3
Paulo	10
Henrique	8

Esse programa tem uma variável string 'y$' seguida de uma variável numérica 'w' no comando READ; daí, no comando DATA, os dados são dispostos correspondentemente, com um string seguido de um número. Como o computador lerá 'y$' primeiro, ele valerá 'João'. A seguir 'w' tomará o valor seguinte no comando DATA, ou seja, '6'. E assim, até a última execução do laço.

Atividades

Reescreva os seguintes programas, corrigindo os erros.

1.
```
10  READ  a$, b, c$
20  PRINT  a$, b, c$
30  DATA  100, 200, 300
```
2.
```
10  READ  x, y, z$
20  PRINT  x, y, z$
30  DATA  gato, cao, peixe
```
3.
```
10  READ  a$, b$, c
20  PRINT  c; a$; b$
30  DATA  3,, "Cego", "ratos"
```
4.
```
10  READ  x$ y b$
20  PRINT  y; b$; x$
30  DATA  garrafas, "10", "verdes"
```
5.
```
10  FOR  n = 1 TO 5
20  READ  x
30  PRINT  x
40  DATA  1, 2, 3, 4
50  NEXT  n
```
6.
```
10  FOR  n = 1 TO 3
20  READ  x
30  PRINT  x
40  DATA  "homem" "mulher" "crianca"
50  NEXT  n
```

7.
```
10  FOR  n = 1 TO 3
20  READ  a$, b
30  PRINT  a$, b
40  DATA  3, "navios", 2, "carros", 1, "trem"
50  NEXT  n
```

8. Qual será o conteúdo das variáveis 'a$', 'b$', e 'c$' ao fim desse programa?
```
10  READ  a$, b$, c$
20  DATA  "Antonio","Jose","Henrique","Jaime","Pedro"
30  PRINT  a$, b$, c$
```

9. Qual o conteúdo das variáveis 'a' e 'b$' ao fim desse programa?
```
10  FOR  n = 1 TO 3
20  READ  a, b$
30  PRINT  a, b$
40  DATA  1, "Franca", 2, "Inglaterra", 3, "Noruega"
50  NEXT  n
```

10. Qual o conteúdo das variáveis 'x', 'y' e 'z' ao fim da terceira execução do laço?
```
10  FOR  n = 1 TO 6
20  READ  x, y, z
30  PRINT  x, y, z
40  DATA  35, 4, 6, 8, 21, 42
50  DATA  10, 3, 17, 9, 12, 23
60  DATA  25, 15, 7, 33, 14, 11
70  NEXT  n
```

Escreva os programas a seguir usando READ e DATA.

11. Imprima uma lista com os 7 dias da semana.

12. Imprima uma lista de 10 números quaisquer entre 35 e 64.

13. Entre a frase "O gato do rei de Roma fugiu da Babilonia"; use uma variável para cada palavra e imprima frases diferentes com essas palavras.

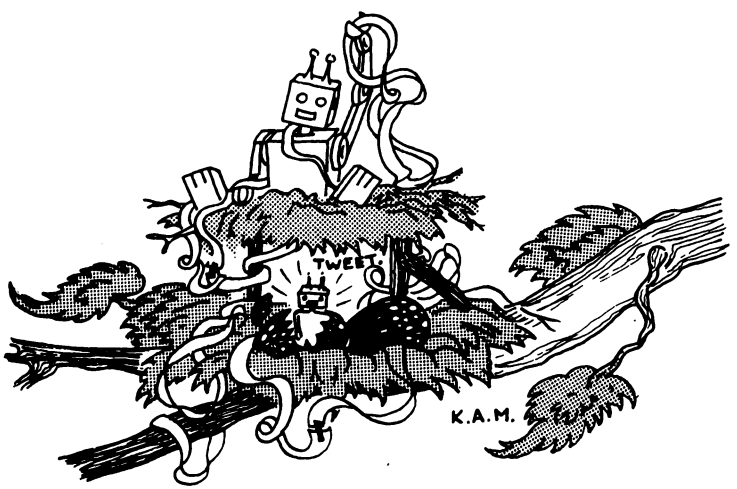
14. Entre e imprima uma lista com os seguintes alimentos e seus preços (em cruzados):

pao 3,60, sopa 19,00, geleia 5,80, feijao 17,00, ovos 4,20, arroz 8,80

Calcule e imprima o custo total desses alimentos.

Unidade 22:

Laços Aninhados



Essa unidade também trata de laços FOR-NEXT, mas nos casos em que um laço é colocado, ou aninhado, dentro de outro.

DIGITE:

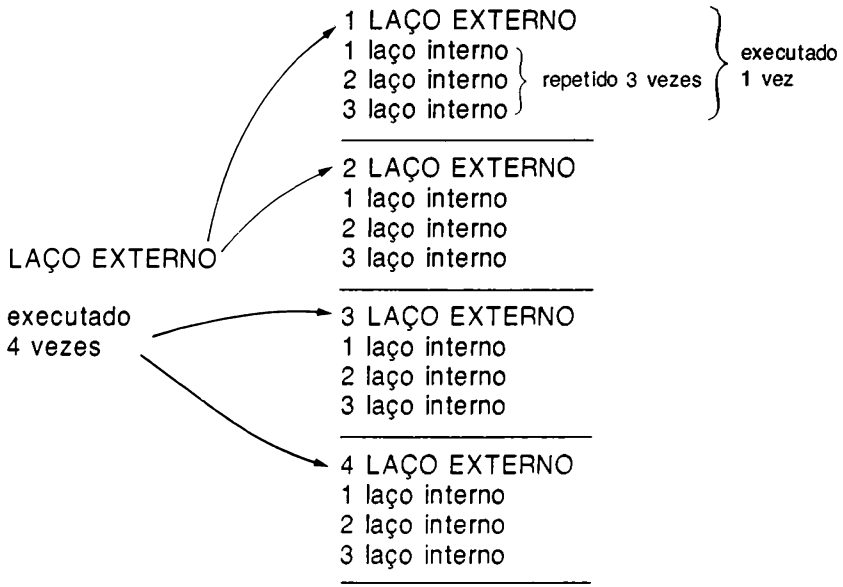
```
10  REM  LACOS ANINHADOS
20  FOR  n = 1 TO 4
30  PRINT  n; " LACO EXTERNO"
40  FOR  x = 1 TO 3
50  PRINT  x; " LACO INTERNO"
60  NEXT  x
70  PRINT  "- - - - - - - - - -"
80  NEXT  n  (Traços - NÃO espaços)
90  STOP     (shift J)
```

LAÇO EXTERNO
Repetido
4 vezes

Laço Interno ou Aninhado. Dentro do laço externo. Repetido 3 vezes toda vez que o laço exterior é executado.

RUN

O resultado é:



Note que as linhas 30 e 70 só são impressas 4 vezes por fazerem parte do laço externo.

Um laço interno ou aninhado tem que estar inteiramente contido no laço externo. Você não pode escrever:

```
20  FOR  n = 1 TO 4
30  FOR  x = 1 TO 3
40  PRINT  x
50  NEXT  n
60  NEXT  x
```

Laços deslocados — não é permitido

O programa seguinte enche a tela de números.

DIGITE:

```
    10  REM  TELA DE NUMEROS
    20  FOR  n = 1 TO 22
    30  FOR  x = 1 TO 20
    40  PRINT  x;
    50  NEXT  x
    60  PRINT
    70  NEXT  n
    80  STOP
```

Laço externo

Laço interno ou aninhado

RUN

O laço interno preenche com valores de x de 1 a 20 as colunas da linha da tela (linhas 30 a 50). Já o laço externo repete isso para as 22 linhas da tela. O ponto-e-vírgula (;) garante que os números sejam impressos um ao lado do outro. A linha 60 garante que a cada vez que o laço externo é executado uma nova linha começa — veja o que acontece se você suprimir a linha 60.

O próximo programa mostra como uma tabuada de multiplicações pode ser impressa por completo usando laços aninhados.

DIGITE:

```
        10  REM  TABUADAS DE MULTIPLICACAO
E  ---> 20  FOR  n = 2 TO 12
X       30  PRINT  n; "  TABUADA DE MULTIPLICACAO"
T       40  PRINT  "-- - - - - - - -"      (Traços — shift J, não espaços)
E       50  FOR  x = 1 TO 12  <---+
R       60  PRINT  x; "*"; n; "="; x * n  |  I
N       70  NEXT  x  ------------------+  N
O  ---> 80  NEXT  n                       T
        90  STOP                          E
                                          R
                                          N
                                          O
```

RUN.
Você terá que "scroll" várias vezes.

O laço externo 'n' dá o número da tabuada de multiplicação: a primeira será por 2, a segunda por 3, até por 12. O laço interno 'x' dá o número a ser multiplicado; se x = 1, será impresso 1 × 2 = 2, se x = 2, teremos 2 × 2 = 4, e assim por diante, terminando com 12 × 2 = 24. Daí, voltamos ao laço externo, para n = 3, e começando novamente com x = 2, e a impressão de 1 × 3 = 3, 2 × 3 = 6 etc.

Podemos agora aplicar laços aninhados na solução de um problema. Uma loja oferece financiamento especial, se for dado um sinal de 15% do preço a ser pago. O cliente pagará o restante, então, em prestações, com planos de 6, 9 e 12 meses. O programa calcula os depósitos requeridos para itens variando de Cz$ 50,00 a Cz$ 500,00, de Cz$ 50,00 em Cz$ 50,00, e depois calcular o valor das prestações mensais:

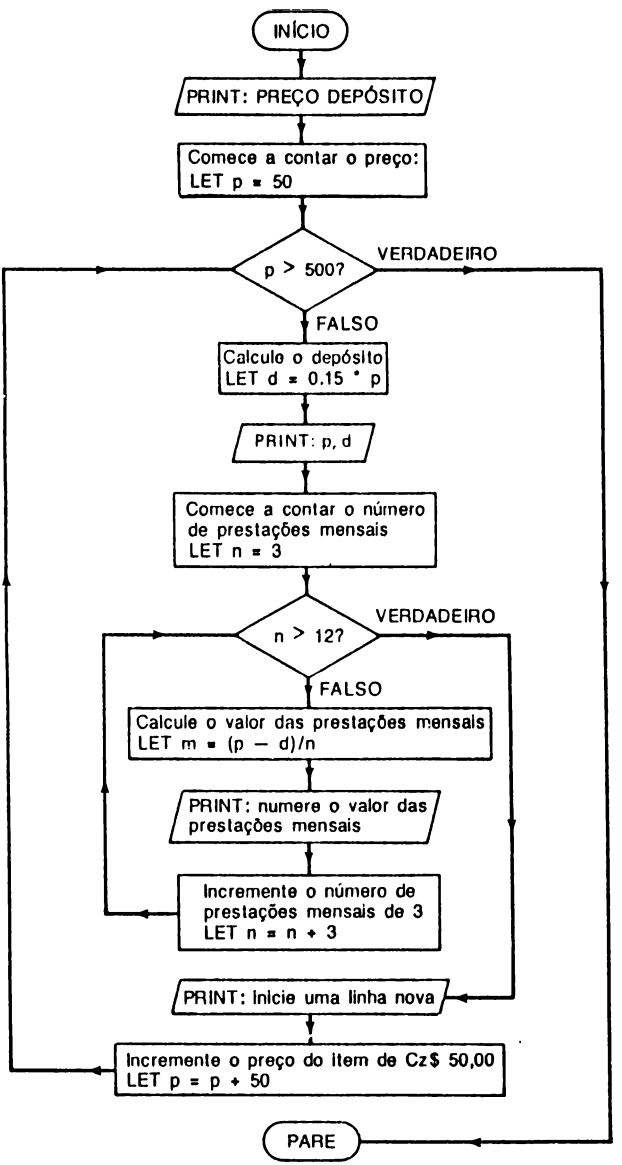
p = preço
d = depósito
n = número de prestações
m = valor das prestações.

DIGITE:

```
                10  REM  TERMOS DO CREDITO
                20  PRINT  "PRECO", "DEPOSITO"
        ------> 30  FOR  p = 50 TO 500 STEP 50
                40  LET  d = 0.15 * p <---------- (fórmula para calcular 15%)
E               50  PRINT  p, d
X   I   ------> 60  FOR  n = 6 TO 12 STEP 3  (3 meses de intervalo)
T   N           70  LET  m = (p-d)/n  (fórmula para calcular pagamentos mensais)
E   T
R   E           80  PRINT  "Os pagamentos mensais durante ";
N   R       n; "  meses e "; m; "  Cz$"
O   N   <------ 90  NEXT  n
    O          100  PRINT
        ------>110  NEXT  p
               120  STOP
```

O laço externo, usando 'p' para preço, calcula e imprime o depósito necessário, a partir de Cz$ 50,00, e de Cz$ 50,00 em Cz$ 50,00 (STEP 50) até Cz$ 500,00. A linha 40 calcula o depósito (0.15 * p). O laço interno, linhas 60 a 90, usa 'n' para o número de meses das prestações. O número de meses varia entre 3, 6 e 12 meses. STEP 3 é usado por isso. A linha 70 calcula o valor a ser pago, em cada caso, a cada mês — o preço menos o depósito, isto dividido pelo número de meses (p — d)/n, o qual é impresso na linha 80. O computador passa por esse laço interno 3 vezes, uma vez para cada plano de pagamento.

Fluxograma para o programa anterior



Atividades

Use laços aninhados para escrever os seguintes programas. Teste-os no computador.

1. Imprima 10 linhas com 10 asteriscos (*).

2. Entre os nomes de 5 pessoas e a quantidade de dinheiro que elas têm em contas no banco. Imprima essa informação. Calcule a quantidade de juros que elas teriam ganho se as taxas de juros fossem 6%, 8% e 10%. Imprima o valor dos juros e o total para cada caso. Use as seguintes variáveis:

s = soma poupada n$ = nome da pessoa

r = taxa de juros

$$i = \text{juros ganhos} = \frac{\text{soma poupada} \times \text{taxa de juros}}{100}$$

n = contador para 5 pessoas.

3. Imprima cinco retângulos de asteriscos (*). O comprimento e largura dos retângulos deverão entrar enquanto o programa for executado. (Esse programa necessitará de 3 laços FOR-NEXT, um dentro do outro.)

Unidade 23:

Gráficos Usando Comandos PRINT, Gráficos com Movimento e a Cores

Até agora controlamos a posição de impressão usando ponto-e-vírgula, vírgula e comando TAB. Aprendemos também como deixar espaços num programa e como começar uma nova linha, apenas usando a palavra PRINT (por exemplo, 50 PRINT). No entanto, podemos controlar melhor a posição de impressão usando PRINT AT. A palavra AT fica na tecla I e é obtida com SYMBOL SHIFT. Se você olhar o diagrama 23.1, você verá que a tela é dividida em 22 linhas e 32 colunas (de 0 a 21 e de 0 a 31, respectivamente).

Usando esses números, primeiro linhas, seguidos de colunas, o computador imprimirá onde quisermos, na tela. Por exemplo:

```
10  PRINT  AT  10, 16; "*"
```

imprimirá um asterisco quase no centro da tela. Note que os dois números são separados por uma vírgula e que um ponto-e-vírgula separa o segundo número do resto da instrução. O programa abaixo imprime asteriscos em diversas partes da tela.

DIGITE:

```
10  REM  ASTERISCOS
20  INPUT  "Entre um numero entre 0 e 21"; a
```

```
30  INPUT  "Entre outro numero entre 0 e 31"; b
40  PRINT  AT a, b; "*"
50  GOTO  20
```

Execute esse programa algumas vezes, com valores diferentes. Para parar o programa use STOP, caso contrário ele seguirá indefinidamente.

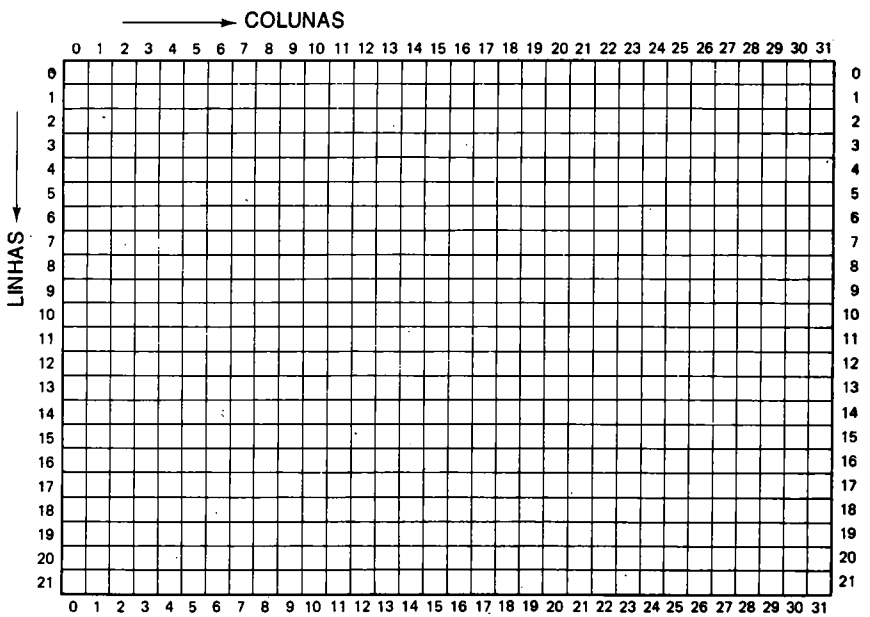
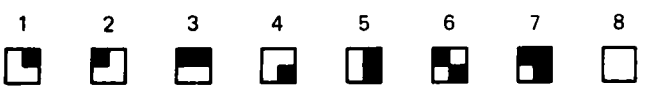


Diagrama 23.1 O número de linhas e colunas na tela do TK90X, para uso do comando PRINT AT.

Atividade

1. Com uma cópia do diagrama 23.1, desenhe uma figura simples com asteriscos e depois use o programa anterior para reproduzi-la na tela. Quando a figura estiver completa, entre STOP.

Agora, ao desenharmos figuras não vamos querer ficar limitados a asteriscos. Vamos então ver os caracteres gráficos. Eles estão nas teclas de 1 a 8:



Apenas as partes em negrito aparecerão na tela; assim, por exemplo, o número 8 produzirá um espaço em branco na tela. Para usar esses caracteres gráficos devemos passar para o **modo gráfico;** isto é, mudar o cursor de **L** para **G.** Isso é feito apertando a tecla CAPS SHIFT e então a tecla 9, na qual está escrito GRAPHICS. Para sair do **modo gráfico** basta apertar novamente a tecla 9.

DIGITE:

```
10  PRINT  AT 10, 9;"
```

RUN NÃO APAGUE

Esses não são os únicos caracteres que podemos produzir. Se, no **modo gráfico,** apertarmos CAPS SHIFT e qualquer número de 1 a 8, teremos o caractere reverso. Ou seja, o que era branco fica negro e o negro fica branco. Por exemplo:

Inclua no programa anterior:

```
20  PRINT  AT 12, 9; "
```

RUN

Você pode agora usar esses caracteres para desenhar figuras. O programa a seguir desenha um ônibus.

DIGITE:

```
10  REM  Desenhando um onibus
20  PRINT  AT 10, 4; "
```

```
30  PRINT  AT 11, 4; "
```

```
40  PRINT  AT 12, 4; "
```

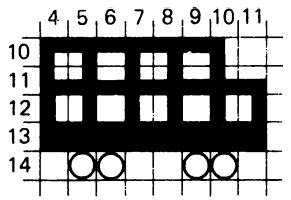
```
50  PRINT  AT 13, 4; "
```

```
60  PRINT  AT 14,5; "OO"
70  PRINT  AT 14, 9; "OO"
```

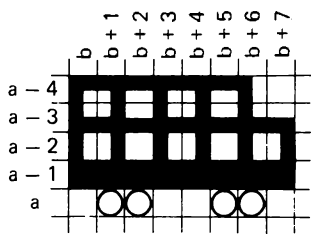
(O maiúsculo)

RUN NÃO APAGUE

Você pode achar que ficou um ônibus feio, mas dá a idéia de como você pode construir uma figura usando caracteres gráficos. É possível modificar esse programa de forma a imprimir ônibus em qualquer lugar da tela. Veja o diagrama a seguir:



As rodas estão na linha 14. (Linhas 60 e 70 do programa); assim, se chamarmos a linha 14 de 'a', a linha 13 será 'a — 1', a 12 será 'a — 2', e assim por diante. Da mesma forma, o canto esquerdo do ônibus está na coluna 4, que podemos chamar 'b', e a linha 5 será 'b + 1', e assim por diante. Isto é ilustrado no diagrama a seguir:



Podemos então mudar as linhas 20 a 70 no nosso programa do ônibus, para usar as variáveis 'a' e 'b'.

Altere o programa anterior:

```
10  REM  Desenhando um onibus
20  PRINT  AT a— 4, b
30  PRINT  AT a— 3, b
40  PRINT  AT a— 2, b
50  PRINT  AT a— 1, b
60  PRINT  AT a, b+1
70  PRINT  AT a, b+5
```

O resto dessas linhas continua inalterado

Agora tudo o que temos a fazer é entrar os números que 'a' e 'b' representam.

Adicione ao programa:

```
12  INPUT  "Entre uma linha"; a
15  INPUT  "Entre uma coluna"; b
80  GOTO  12
```

Execute o programa, de tal forma que nove ônibus apareçam na tela, sem se sobrepor ou serem truncados. Você precisará do diagrama 23.1. Se você passar da margem, toda sorte de coisas estranhas acontecerá. Se você fizer isso com sucesso, faça os ônibus sobreporem-se propositadamente. O que acontece?
NÃO REMOVA.

Para colorir a figura você pode usar os comandos:

INK (abaixo da tecla X) } usadas no **modo estendido**
PAPER (abaixo da tecla C) } com **symbol shift**
BORDER (palavra-chave na tecla B) – usada no modo **palavra-chave**

As cores que podem ser usadas estão escritas acima das teclas numéricas, pelas quais são obtidas; por exemplo: azul está no número 1, vermelho no número 2, e assim por diante. Podemos dizer ao computador para colorir a margem ou a tela, ou ambos. A margem é a parte da tela que não é impressa, exceto para comandos INPUT.

Adicione ao programa anterior:

```
16  BORDER  1
```

Apague a linha 80.

Execute o programa. A margem continuará azul, até que outro comando seja dado.

Mude a linha 16:

```
16  BORDER  2
```

RUN.

Tente quantas cores quiser, mas escolha uma que você goste de usar.

NÃO REMOVA.

Para colorir o ônibus precisamos mudar a tinta. Faremos o ônibus vermelho e as rodas pretas.

Adicione ao programa:

```
19  INK  2
55  INK  0
```

RUN NÃO APAGUE

Finalmente, podemos mudar a cor do fundo, ao mudar a cor do papel.

Adicione ao programa:

```
17  PAPER  6
```

RUN o programa.

Experimente com INK e PAPER, variando as cores.

NÃO APAGUE

Também é possível fazer o ônibus se mover. Faremos isso usando um laço FOR-NEXT.

Altere o programa:

Apague a linha 10

Adicione:

```
5  FOR  r = 0 TO 31
6  PRINT  AT 12, r; " ■ "
7  NEXT  r
```

Isso imprime uma estrada para o ônibus passar

Edite a linha 12:

```
12  LET  a = 11
```

Apague a linha 15.
Adicione:

```
18  FOR  b = 1 TO 23
80  NEXT  b
```

Edite as linhas 20 a 70

```
10  REM  Desenhando um onibus
20  PRINT  AT a– 4, b; "
30  PRINT  AT a– 3, b; "
40  PRINT  AT a– 2, b; "
50  PRINT  AT a– 1, b; "
60  PRINT  AT a, b+1; "OO"
70  PRINT  AT a, b+5; "OO"
```

O resto das linhas permanece inalterado, mas com a adição de um espaço antes do primeiro caractere gráfico em cada linha

RUN NÃO APAGUE

O ônibus agora correrá na tela, da esquerda para a direita. Podemos tornar esse movimento mais lento usando o comando PAUSE na tecla M. Por exemplo:

PAUSE 50 significa esperar 1 segundo
PAUSE 150 significa esperar 3 segundos (você poderia usar PAUSE 50 * 3)

Nós não queremos retardar o ônibus demais, por isso usaremos, apenas, parte do segundo.

Adicione:

```
75  PAUSE  10
```

RUN NÃO APAGUE

O ônibus agora passará na tela (meio desconjuntadamente). O laço FOR-NEXT é o responsável pelo movimento. 'b' recebe o valor 1, e na coluna 1 é colocado o canto esquerdo do ônibus; na próxima execução do laço, 'b' receberá 2, passando o ônibus uma coluna mais para a direita, e assim por diante até a coluna 22. À medida que o ônibus se move, a coluna anterior mais à esquerda é apagada pelo espaço em branco que adicionamos nas linhas 20 a 70. Se você quiser alterar a velocidade do ônibus, mude a linha 75 — lembre-se, PAUSE 50 é um segundo.

Edite a linha 75:

```
75  PAUSE  0
```

RUN — o ônibus não se moverá até que você aperte quaisquer teclas, exceto as duas teclas shift.

REMOVA O PROGRAMA.

Atividade

2. (a) Escreva um programa que imprima uma das seguintes figuras: carro, trem, navio, avião. Use cores diferentes para margem, papel e tinta. Faça seu veículo move-se na tela.

(b) Altere seu programa para que seu veículo atravesse a tela 6 vezes. Cada vez que ele atravessa a tela deve ser impresso numa cor diferente. (Faça a cor do PAPEL igual a branco.)

Vamos agora observar os objetos em movimento em maior detalhe. O próximo programa move um quadrado negro pelos cantos da tela. A única parte complicada é fazer com que o quadrado vá sendo apagado por um espaço, assim que o próximo quadrado seja impresso. As duas variáveis usadas são 'a' para linhas e 'b' para colunas. Cada parte do programa começa com um comando PRINT, o qual informa por onde o quadrado está passando. As linhas 40, 130, 200 e 280 usam um novo comando; CLS (tecla V) que remove o que estiver escrito na tela, antes do quadrado passar por ali. O comando PAUSE, antes de cada uma dessas linhas, assegura que a mensagem fique por 2 segundos na tela, o suficiente para que você possa ler, antes de ser apagada. (Não confunda CLEAR (tecla X) com CLS. CLS significa "CLEAR Screen", em Inglês, ou seja, limpar a tela; já CLEAR, que significa em Inglês 'limpar', apaga a memória do computador.)

DIGITE:

```
 10  REM  Movendo quadrados
 20  PRINT  "Quadrado movendo-se para baixo
e pela esquerda"
 30  PAUSE  100
 40  CLS
 50  LET  b=0
 60  FOR  a=0 TO 18
 70  PRINT  AT a, b; " "
 80  PRINT  AT a+1, b; "■"
 90  PAUSE  10
100  NEXT  a
110  PRINT  "Quadrado movendo-se pelo canto
inferior da esquerda p/direita"
120  PAUSE  100
130  CLS
140  FOR  b=0 TO 30
150  PRINT  AT  a, b; " ■"
160  PAUSE  10
170  NEXT  b
180  PRINT  "Quadrado movendo-se para cima
pela direita"
190  PAUSE  100
```

```
200  CLS
210  FOR  a=19 TO 1 STEP – 1
220  PRINT  AT  a– 1, b; "■"
230  PRINT  AT a, b; " "
240  PAUSE  10
250  NEXT  a
260  PRINT  "Quadrado movendo-se pelo canto
superior da direita p/esquerda"
270  PAUSE  100
280  CLS
290  FOR  b=30 TO 0 STEP – 1
300  PRINT  AT  a, b; "■ "
310  PAUSE  10
320  NEXT  b
330  STOP
```

RUN

Mover o quadrado para baixo é feito adicionando-se 1 ao valor de 'a' (as linhas) toda vez que o laço FOR-NEXT 'a' é executado. O valor de 'b' (as colunas) permanece inalterado, enquanto o quadrado não muda de coluna. A linha 70 imprime o espaço em branco que seguirá o quadrado, apagando o anterior, enquanto ele se move para baixo. O primeiro espaço é impresso em 0, 0 (a = 0 e b = 0); assim, o primeiro quadrado é impresso em 1, 0 (a + 1 = 1, b = 0). Na segunda vez que o laço é executado o espaço será impresso em 1, 0 (a = 1, b= 0) e o quadrado em 2, 0 (a + 1 = 2, b = 0).

Mover o quadrado na tela da esquerda para a direita já foi discutido no programa do ônibus. Para fazer o quadrado andar no sentido de baixo para cima, basta inverter o laço; 'a' começa em 19, decrescendo de 1 em 1, e não crescendo (STEP — 1 na linha 210). O espaço deve continuar seguindo o quadrado, e, por isso, é impresso abaixo do quadrado, na posição 19, 30 (a = 19, b = 30, enquanto o quadrado se movia da coluna 0 para 30 na parte anterior do programa), e o primeiro quadrado na posição 18, 30 (a — 1 = 18, b = 30). Finalmente, o quadrado se move da direita para a esquerda aplicando-se a mesma inversão, só que agora no valor de 'b', que começa em 30. A linha não se altera. O espaço continua seguindo o quadrado, agora no lado direito.

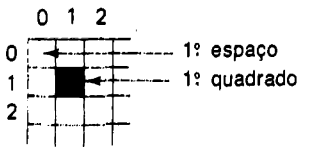
O próximo programa move o quadrado diagonalmente do canto superior esquerdo para perto do canto inferior direito.

DIGITE:

```
10  REM  Movendo o quadrado diagonalmente
20  FOR  n=0 TO 20
30  PRINT  AT n, n; " "
40  PRINT  AT n+1, n+1; "■"
50  PAUSE  10
60  NEXT  n
70  STOP
```

RUN

Agora, linhas e colunas são representadas por 'n' e começam em 0. O primeiro espaço impresso fica em 0, 0 (n, n) e o quadrado, em 1, 1 (n + 1, n + 1). Veja o diagrama a seguir:



Na segunda vez que o laço 'n' é executado, 'n' tem valor 1, o espaço é impresso em 1, 1 (n, n) e o quadrado é impresso em 2, 2 (n + 1, n + 1).

Atividades

3. Estude o programa abaixo e descubra de onde e para onde o quadrado está se movendo. Desenhe um diagrama para mostrar onde os primeiros, espaço e quadrado, serão impressos.

```
10  FOR  n = 0 TO 19
20  PRINT  AT n, 20– n; " "
30  PRINT  AT n+1, 19– n; " ■"
40  PAUSE  10
50  NEXT  n
60  STOP
```

4. Escreva um programa que mova um quadrado negro diagonalmente na tela do próximo do canto inferior direito para o canto superior esquerdo. Comece na linha 20 e coluna 20.

O próximo programa permite que você desenhe uma figura movendo um quadrado na direção que quiser. Ele usa as teclas numéricas 5, 6, 7 e 8 para mostrar a direção do movimento pelas setas desenhadas acima dessas teclas. Para isso o programa usa a função INKEY$ (sobre a tecla N – modo estendido). Essa função permite averiguar por programa que tecla foi pressionada.

DIGITE:

```
 10  REM  Desenhando um quadrado
 20  LET  a=11  } (a = linhas, b = colunas)
 30  LET  b=16  }
 40  PRINT  AT a, b; " ■"
 50  IF  INKEY$="5" THEN LET b=b–1
 60  IF  b < 0 THEN LET b = 0
 70  IF  INKEY$ = "6" THEN LET a = a + 1
 80  IF  a > 21 THEN LET a = 21
 90  IF  INKEY$ = "7" THEN LET a = a – 1
100  IF  a < 0 THEN LET a = 0
110  IF  INKEY$ = "8" THEN LET b = b + 1
120  IF  b > 31 THEN LET b = 31
130  GOTO  40
```

RUN

A única forma de parar o programa é pressionar CAPS SHIFT e BREAK ao mesmo tempo, seguido por ENTER que retorna o programa para a tela.

Esse programa traça uma linha negra contínua porque nós não apagamos os quadrados à medida que fomos avançando. O primeiro quadrado é impresso em 11, 16, que são os valores iniciais de 'a' e 'b'. Esses valores vão mudando à medida que as teclas são apertadas. Se '5' é pressionado, 'b' se torna 'b — 1', e o quadrado se move para a esquerda; 'a' continua inalterado, pois não se muda de linha. Se '6' é apertado, o quadrado desce 1 posição, a linha cresce de 1, com 'a' = a + 1. Se '7' é pressionado, 'a' passa a valer 'a — 1', e o quadrado sobe 1 posição. Se '8' é apertado, 'b' passa a 'b + 1' e o quadrado se move para a direita. As linhas 80 e 100 asseguram que 'a' se mantenha entre 0 e 21, e as linhas 60 e 120 que 'b' fique entre 0 e 31. Isso impede uma condição de erro, mantendo o quadrado nos limites da tela.

Atividade

5. Altere o programa de tal forma que o quadrado vá sendo apagado à medida que se mover. Você terá que pôr espaços em branco ao redor do quadrado, de tal forma que ele seja apagado em qualquer direção que vá. Além disso, cuidado para não tentar imprimir um espaço fora da tela, o que causará mensagens de erro.

A idéia de mover um objeto através da tela é comum em muitos jogos de computador.

Para concluir esta unidade, vamos atentar para três outros comandos:

FLASH (tecla V) BRIGHT (tecla B) INVERSE (tecla M)	Todos usados no modo estendido com SYMBOL SHIFT

FLASH significa que os caracteres impressos ficarão piscando na tela, como os cursores piscantes usados pelo computador. O número 1 faz os caracteres piscarem e o número 0 pára isso.

DIGITE:

```
10  FLASH 1
20  PRINT  "Estes caracteres estao pulsando"
30  FLASH  0
40  PRINT  "Estes caracteres nao estao pulsando"
50  STOP
```

RUN

BRIGHT significa que os caracteres aparecerão com brilho. Funciona da mesma forma que o FLASH.

DIGITE:

```
10  BRIGHT  1
20  PRINT  "Eu estou brilhante"
30  BRIGHT  0
40  PRINT  "Eu estou opaco"
50  STOP
```

RUN

INVERSE inverte as tintas, da mesma forma que o FLASH, mas sem piscar de uma para outra. Novamente, 0 e 1 são usados da mesma forma.

DIGITE:

```
10  INVERSE  1
20  PRINT  "Meus caracteres sao branco num
fundo negro — estou invertido"
30  INVERSE  0
40  PRINT  "Meus caracteres estao normais"
50  STOP
```

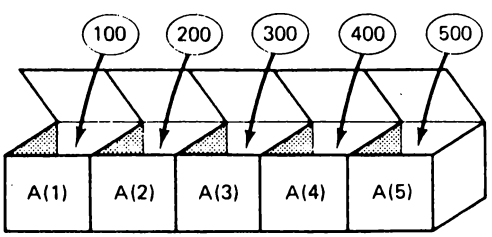
RUN

Se FLASH, BRIGHT e INVERSE não são desligados no fim do programa continuam influenciando tudo que se seguir (por exemplo, a listagem do programa).

Unidade 24:

Arrays - Comando DIM

Um array é um conjunto ou grupo de espaços na memória, todos com o mesmo nome. O nome de um array é apenas uma letra do alfabeto (maiúscula ou minúscula). É parecido com uma variável simples, só que ao invés de um único espaço na memória, temos vários, como uma caixa com vários compartimentos.



Array numérico com 5 elementos

(1) N.T.: **Array** às vezes é traduzido como **vetor.** No entanto, um array pode representar um vetor, uma matriz bidimensional etc. Por isso, no intuito de não induzir o leitor à falsa idéia de que arrays são simplesmente vetores, e pelo grau de divulgação do termo no meio técnico, preferimos manter o termo inglês.

Cada espaço de armazenamento de um array é chamado um elemento, e é referenciado por um número. Por exemplo, se nós usarmos um vetor chamado 'A' para armazenar 5 números, então o primeiro número será armazenado no primeiro elemento ou espaço, chamado A(1), o segundo número no A(2), e assim até o último número ser armazenado no A(5). Teremos então um array numérico com 5 elementos — ele conterá 5 variáveis numéricas.

O array ilustrado na página anterior mostra que o número 100 é armazenado em A (1), o número 200 em A (2), e assim por diante. O computador pode acessar qualquer um desses números simplesmente referenciando o nome e o número do espaço de armazenamento. Por exemplo, digite PRINT A (1) e 100 aparecerá na tela.

DIGITE:

```
20  LET   A (1) = 100
30  LET   A (2) = 200
40  LET   A (3) = 300
50  LET   A (4) = 400
60  LET   A (5) = 500
70  PRINT   A (1). . A (2). . A (3). . A (4). . A (5)
```

RUN!

Você ficará surpreso ao perceber que o programa só funciona se a cada elemento do array tiver sido atribuído um valor. A mensagem de erro

2 variable not found, 20 : 1

apareceu na tela. Isso porque, antes de um array poder ser usado, o computador precisa ser informado do nome e do tamanho do vetor, para saber quanto espaço de memória precisará. Isso é feito através do comando DIM, que é uma abreviação para **dimensão.** DIM está na tecla D.

DIGITE:

```
10  DIM A(5)
      |  └ tamanho — número de elementos
      nome do array
```

RUN

NÃO APAGUE

O resultado agora é o que você esperava:

```
100
200
300
400
500
```

No entanto, você pode, provavelmente, estar achando que o mesmo resultado poderia ser obtido usando 5 variáveis simples. Por exemplo:

```
20  LET   A = 100
30  LET   B = 200
```

```
40  LET  C = 300
         etc.
70  PRINT  A,, B,, C.....
```

Assim, o programa seguinte usa o mesmo vetor, mas apenas imprime o número pedido pela pessoa durante a execução do programa, o que não poderia ser feito com variáveis numéricas ordinárias.

Deixe as linhas 10 a 60 como estão.
Apague a linha 70.
DIGITE:

```
70  INPUT  "Entre um numero de 1 a 5__"; n
80  PRINT  A(n)
90  STOP
```

Execute (RUN) o programa diversas vezes, com números diferentes.

Como você pode ver, o número 'n' entrado na linha 70 é usado com o nome do array 'A' na linha 80. Assim, se o número 2 fosse entrado, então A(n) seria na verdade A(2) e 200 seria impresso.

Outro uso para um array é o de entrar números no array enquanto o programa é executado.

DIGITE:

```
10  DIM  b(7)
20  FOR  n = 1 TO 7
30  INPUT  "Entre um numero "; b(n)
40  PRINT  b(n)
50  NEXT  n
60  PRINT  b(1)
70  PRINT  b(3)
80  PRINT  b(7)
```

RUN. Verifique se as linhas 60 a 80 imprimem os números que você deseja.
NÃO APAGUE.

Esse programa usa um array chamado 'b' o qual tem 7 elementos ou espaços de armazenamento. As linhas 20 a 50 formam um laço FOR-NEXT 'n' que será executado 7 vezes. Na linha 30, na primeira execução, o número será armazenado na posição 'b(1)', já que 'n' vale 1. Na segunda vez, 'n' = 2 e o número entrado será armazenado na posição b(2). E assim por diante até o sétimo número ser armazenado na posição b(7) e o array ficar completo. A linha 40 simplesmente imprime os valores de cada elemento ou variável, enquanto as linhas 60 a 80 provam que realmente as variáveis 'b(1)', 'b(2)' etc. existem.

É possível tratar os elementos de um array como variáveis comuns e usá-los como nós quisermos. Se nós estendermos o programa anterior, podemos imprimir os valores dos elementos em qualquer ordem.

DIGITE:

```
100  PRINT
110  FOR  s = 7 TO 1 STEP – 1
120  PRINT  b(s)
130  NEXT  s
140  PRINT
150  FOR  r = 1 TO 7 STEP 2
160  PRINT  b(r)
170  NEXT  r
180  STOP
```

Se entrarmos os núrneros

11, 22, 33, 44, 55, 66, 77

você consegue escrever o resultado do programa, antes de executá-lo?
Execute o programa e verifique o resultado.

Inclua instruções para que os números 77, 44 e 11 sejam impressos nessa ordem. (Lembre-se de retirar o comando STOP da linha 180 primeiro.)

Execute o programa e verifique o resultado.

Podemos também armazenar números num array através dos comandos READ e DATA. Isso é ilustrado no próximo programa.

DIGITE:

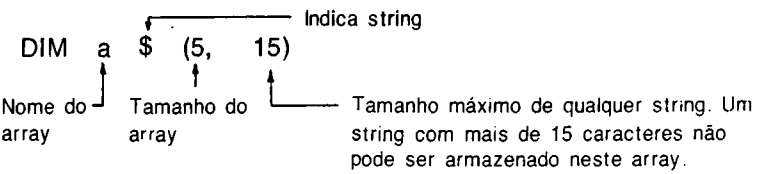
```
10  DIM  d(10)
20  FOR  g = 1 TO 10
30  READ  d(g)
40  DATA  11, 12, 13, 14, 15, 16,
17, 18, 19, 20
50  PRINT  d(g)
60  NEXT  g
70  STOP
```

RUN

Usar os comandos READ e DATA num laço FOR-NEXT é muito mais rápido do que armazenar os números num array da forma tradicional:

```
LET  d(1) = 11
LET  d(2) = 12
LET  d(3) = 13
       etc.
```

Até aqui usamos, somente, arrays numéricos, mas podemos obviamente ter arrays de strings. Novamente, o nome deverá ter uma letra apenas, seguida do cifrão ($). O tamanho do array deve ser dado como antes, e também o tamanho máximo da string a ser armazenada. Por exemplo:



DIGITE:

```
10  DIM  a$ (5, 15)
20  FOR  n = 1 TO 5
30  READ  a$ (n)
40  DATA  "Adeus", "Ola", "Feliz Natal",
"Feliz Ano Novo", "Durma bem"
50  NEXT  n
60  INPUT  "Entre com o numero da mensagem
que voce deseja que seja impressa "; n
70  PRINT  a$ (n)
80  STOP
```

Execute o programa diversas vezes, entrando números diferentes.

O comando DIM cria um array chamado 'a$' com 5 elementos, e o maior string usado terá 15 caracteres, que é "Feliz Ano Novo". (Lembre-se que o espaço em branco conta como 1 caractere.) O laço FOR-NEXT 'n' lê os dados e armazena nos 5 elementos do array. A mensagem a ser impressa é decidida pelo número fornecido na linha 60.

O próximo programa chamado "JOGO DAS FRUTAS" usa dois arrays de strings e muitas das técnicas aprendidas nas últimas unidades.

DIGITE:

```
 10  REM  Jogo das Frutas
 20  DIM  a$ (10, 7)
 30  DIM  b$ (10, 7)
 40  FOR  n = 1 TO 10
 50  READ  a$ (n)
 60  NEXT  n
 70  FOR  m = 1 TO 10
 80  READ  b$ (m)
 90  NEXT  m
100  DATA  "laranja", "limao", "laranja", "morango",
"maca", "laranja", "morango", "limao", "maca", "laranja",
"morango", "maca", "limao", "morango", "laranja",
"maca", "limao", "laranja", "maca", "limao"
110  LET  s = 0
120  PRINT  "JOGO DAS FRUTAS"
130  PRINT  "REGRA — voce nao pode entrar
o mesmo par de numeros duas vezes"
140  PRINT
150  FOR  a = 1 TO 5
160  INPUT  "Entre um numero entre
1 e 10"; n
170  PRINT  a$ (n); "    ";
```

```
180  INPUT  "Entre um numero entre
1 e 10 "; m
190  PRINT  b$ (m)
200  IF  a$ (n) < > b$ (m)
THEN PRINT "Voce perdeu — nenhum ponto
marcado": GOTO 230
210  IF  a$ (n) = b$ (m) THEN PRINT "voce
venceu — 5 pontos marcados"
220  LET  s = s + 5
230  PRINT
240  NEXT  a
250  PRINT
260  PRINT
270  PRINT  "Voce marcou "; s; " pontos"
280  STOP
```

RUN

As linhas de 20 a 100 armazenam as informações nos dois arrays chamados 'a$' e 'b$'. Cada array com 10 elementos, cada elemento com tamanho máximo de 7 caracteres. Dois laços FOR-NEXT 'n' e 'm', linhas 40 a 60 e 70 a 80, alimentam a informação usando READ e DATA. Note que todos os strings estão contidos num único comando DATA, o que é possível porque o primeiro laço 'n' toma os 10 primeiros elementos e o segundo laço 'm' toma os 10 restantes, armazenando em a$(1), a$(2), ... etc., e em b$(1), b$(2), ... etc., respectivamente.

O contador 's', para guardar os pontos, é inicializado na linha 110. O título e a regra do jogo são impressos nas linhas 120 e 130. Outro laço FOR-NEXT, usando a variável 'a', proporciona ao jogador 5 tentativas. Os números 'n' e 'm' são entrados nas linhas 160 a 190, e se referem a dois espaços de memória diferentes, um para cada array. O conteúdo desses elementos é impresso, 'a$(n)' e 'b$(m)'. As linhas 200 e 210 comparam o conteúdo de 'a$(n)' e de 'b$(m)'; se não forem iguais, o jogador perde. Se forem, ele ganha 5 pontos, os quais são acumulados a cada tentativa, na variável 's', na linha 220. Isso, para 5 tentativas. O programa termina imprimindo os pontos ganhos no final.

Outra coisa importante a ser observada sobre arrays de strings é que o computador completa com brancos à direita, automaticamente, qualquer string armazenado num elemento, cujo comprimento seja menor que o declarado no comando DIM. Veja o exemplo a seguir.

Se criarmos um array de string, DIM a$ (6, 7), então a memória do computador se parecerá com o seguinte.

Nome do array e número do elemento	Comprimento do string						
	1	2	3	4	5	6	7
a$(1)							
a$(2)							
a$(3)							
a$(4)							
a$(5)							
a$(6)							

Esse array espera ser preenchido com 6 strings, cada um com 7 caracteres. Se os seguintes strings forem entrados

cão, girafa, camelo, peixe, rato, galinha

O array ficará assim (um traço representa um espaço em branco).

Nome do array e número do elemento	Comprimento do string						
	1	2	3	4	5	6	7
a$(1)	c	ã	o	–	–	–	–
a$(2)	g	i	r	a	f	a	–
a$(3)	c	a	m	e	l	o	–
a$(4)	p	o	l	x	e	–	–
a$(5)	r	a	t	o	–	–	–
a$(6)	g	a	l	i	n	h	a

O que não for ocupado é preenchido com brancos, que são caracteres. Assim todos os strings ficam com o mesmo comprimento. É preciso levar isso em consideração. Se você estiver escrevendo um programa que compare duas palavras num array, elas terão de ser do mesmo tamanho. Por exemplo, considere o diagrama anterior e o programa a seguir:

```
50  INPUT  "Entre o nome de um animal"; b$
60  IF  b$ = a$(1) THEN PRINT "Au, au"
```

Mesmo que a palavra 'cão' tenha sido entrada em b$, o programa ignorará o resto da linha 60, pois o 'cão' de b$ tem 3 caracteres e o 'cão' de a$(1) tem 7, sendo considerados diferentes. No 'Jogo das Frutas', comparamos strings de 2 arrays diferentes, mas isso foi possível por ambos terem o mesmo comprimento. Caso contrário, não funcionaria. Veja o exemplo a seguir.

DIGITE:

```
10  DIM  a$ (3, 5)
20  LET  b$ = "oi"
30  LET  a$ (1) = b$
40  IF  a$ (1) = b$ THEN PRINT "O K"
```

RUN NÃO APAGUE

Nada acontece, exceto a mensagem 0 OK, 40 : 1. Isso porque não é possível para a$(1) conter a palavra 'oi', que tem 2 caracteres, já que a$ tem elementos de 5 caracteres cada.

Altere a linha 20

```
20  LET  b$ = "oi———"
```

RUN NÃO APAGUE

O computador agora imprime 'OK', pois b$ contém agora 5 caracteres, compatível com a$(1). No entanto, não é a melhor forma de resolver o problema; é muito mais conveniente que b$ seja um array, cujo comprimento máximo seja também de 5 caracteres.

Modifique o programa.

```
15  DIM  b$ (1, 5)
20  LET  b$ (1) = "oi"
30  LET  a$ (1) = b$ (1)
40  IF  a$ (1) = b$ (1) THEN PRINT "O K"
```

RUN

'b$(1)' tem automaticamente 5 caracteres, sendo acrescentados 3 brancos à palavra 'oi', e assim 'OK' é impresso. Vejamos se isso funciona num programa mais complicado.

DIGITE:

```
 10  REM  COMPUTADORES
 20  DIM  a$ (12, 9)
 30  DIM  x$ (1, 9)
 40  PRINT  "COMPUTADORES"
 50  INPUT  "Entre o nome de um computador ";
 x$ (1)
 60  FOR  n = 1 TO 12
 70  READ  a$ (n)
 80  IF  x$ (1) = a$ (n) THEN PRINT "Sim, existe
 um computador chamado "; x$ (1): GOTO 120
 90  NEXT  n
100  PRINT  "Nao, nao existe um computador
chamado "; x$ (1)
110  DATA  "Spectrum", "Oric", "B.B.C",
"Apple", "Electron", "Atari", "ZX81",
"Research", "Acorn", "Commodore", "I.B.M.",
"Dragon"
120  STOP
```

RUN

O nome entrado na linha 50 e armazenado no array x$(1, 9) é comparado com o conteúdo do array 'a$(12, 9)'. Isso é possível porque ambos os arrays contêm variáveis string que têm nove caracteres de comprimento. Como o array 'a$' tem 12 elementos, o conteúdo de x$(1) é comparado com cada um, a partir de a$(1). O comando GOTO na linha 80 faz o computador saltar fora do laço FOR-NEXT 'n', o que é algo que não tínhamos tentado antes. Você pode entender por que isso é necessário?

Um programa pode conter arrays numéricos e string, mas separadamente. Não é possível misturar variáveis numéricas e string no mesmo array.

DIGITE:

```
10  REM  MONTANHAS
20  DIM  a$ (6, 11)
30  DIM  x (6)
40  FOR  n = 1 TO 6
50  READ  a$ (n)
60  DATA  "Everest", "Kilimanjaro", "Table",
"Blanc", "Matterhorn", "Ben Nevis"
70  NEXT  n
80  FOR  m = 1 TO 6
90  READ  x (m)
100 DATA  8848, 5895, 1087, 4807, 4478, 1347
110 NEXT  m
120 PRINT  "MONTANHA", "ALTURA EM METROS"
130 PRINT
140 FOR  Y = 1 TO 6
150 PRINT  a$ (y), x (y)
160 NEXT  y
170 STOP
```

RUN NÃO APAGUE

As linhas 20 e 30 criam os dois arrays, a$(6, 11) para strings e x(6) para números. O primeiro laço FOR-NEXT 'n' entra o nome das montanhas nos seis elementos do array string. O segundo laço FOR-NEXT 'm' entra as alturas em metros de cada montanha no array x(6). Note que não há ligação entre os dois arrays. A linha 120 imprime os cabeçalhos das duas colunas. O terceiro laço FOR-NEXT 'y' imprime as informações em duas colunas.

Por que a variável 'y' é usada com a$ e x no último laço?

Atividades

1. (a) Inclua um outro array de strings no programa anterior para guardar os nomes dos países onde as montanhas se situam. Use as informações abaixo.

Montanha	**País**
Everest	Nepal-Tibet
Kilimanjaro	Tanzânia
Table	África do Sul
Blanc	França
Matterhorn	Suíça
Ben Nevis	Escócia

(b) Adicione uma outra parte ao programa, para imprimir essa informação em duas colunas. Cada coluna deve ter um cabeçalho. Execute seu programa.

2. Escreva um programa que recebe 8 números e os imprime na ordem inversa.

3. Escreva um programa que recebe 12 números e os imprime salteados; um sim, outro não.

4. (a) Escreva um programa que armazena os nomes dos dias da semana e então imprime o nome do dia, conforme o número entrado pelo usuário.

(b) Modifique o programa para que, se o número entrado for menor que 1, apareça a mensagem "número muito pequeno" e se for maior que 7, apareça a mensagem "número muito grande" e o computador volte para a linha que lê o número.

(c) Modifique o programa para que ele rode quatro vezes antes de parar.

5. Escreva um programa que recebe uma palavra e verifica se se trata de nome de um mês. Se for, o programa imprime o número do mês (por exemplo, maio é mês 5). Se a palavra não for um mês, então a mensagem "Não é mês" deverá aparecer. (Você vai precisar de dois arrays, um para os nomes dos meses e outro para as palavras entradas.)

6. Escreva um programa que armazene e imprima a seguinte informação.

COMO A ELETRICIDADE É PRODUZIDA NA GRÃ-BRETANHA

Fonte de eletricidade	**Porcentagem do total produzido**
Carvão	79
Óleo	8
Nuclear	11
Água	2

Teste seus programas no computador.

Todos os arrays usados até aqui são chamados **arrays unidimensionais.** Isso porque eles contêm apenas um conjunto ou lista de elementos.[1] No entanto, é possível ter **arrays bidimensionais.** Por exemplo, DIM a(6, 3) significa que o array chamado 'a' tem 18 elementos, dispostos em seis linhas e três colunas. Os nomes dos elementos no array são:

	Coluna 1	**Coluna 2**	**Coluna 3**
Linha 1	a (1, 1)	a (1, 2)	a (1, 3)
Linha 2	a (2, 1)	a (2, 2)	a (2, 3)
Linha 3	a (3, 1)	a (3, 2)	a (3, 3)
Linha 4	a (4, 1)	a (4, 2)	a (4, 3)
Linha 5	a (5, 1)	a (5, 2)	a (5, 3)
Linha 6	a (6, 1)	a (6, 2)	a (6, 3)

Note que o primeiro índice entre parênteses se refere à linha e o segundo indica a coluna:

```
DIM a    (6 ,     3)
     |    |       |
Nome do  Número de  Número de
array    linhas     colunas

6 × 3 = um vetor de18 elementos
```

[1] N.T. Seria melhor dizer que o nome **unidimensional** vem do fato de que usamos apenas **uma** coordenada para localizar um elemento qualquer no conjunto. Num array **bidimensional** precisamos de **duas** coordenadas (linha e coluna), e assim por diante.

O próximo programa usa um array numérico bidimensional.

DIGITE:

```
 10  REM  HORARIO DO TREM
 20  DIM  a (6, 3)
 30  FOR  n = 1 TO 6
 40  FOR  n = 1 TO 3
 50  READ  a (n, m)
 60  DATA  7.01, 7.48, 8.04
 70  DATA  7.14, 8.03, 8.19
 80  DATA  7.32, 8.19, 8.33
 90  DATA  7.45, 8.32, 8.49
100  DATA  8.01, 8.54, 9.07
110  DATA  8.13, 9.03, 9.19
120  NEXT  m
130  NEXT  n
140  PRINT  "BRIGHTON        E. CROYDON
VICTORIA"
150  PRINT  "SAIDA           CHEGADA
CHEGADA"
160  FOR  x = 1 TO 6
170  FOR  y = 1 TO 3
180  PRINT  a (x, y); "
190  NEXT  y
200  PRINT
210  NEXT  x
220  STOP
RUN
```

Pontos a observar:

Linha 20 — cria o array numérico bidimensional.
Linha 30 — conta o número de linhas — 1 a 6.
Linha 40 — conta o número de colunas — 1 a 3.
Linha 50 — lê a informação e coloca nos elementos do array

```
  a     (n,     m)
nome, linha, coluna
```

Linhas 60-110 — fornece o horário do trem. Na primeira execução do laço, o elemento a(1, 1) recebe o valor 7.01; o laço interno é então repetido e o segundo elemento a(1, 2) recebe o valor 7.48; o laço interno é repetido novamente, e a(1, 3) recebe o valor 8.04; a primeira linha do array fica completa. O laço externo é repetido, 'n' recebe 2 e 'm' volta a 1, e o próximo elemento é a(2, 1) que recebe 7.14. Esse padrão se repete até que o laço externo é executado seis vezes e então todos os elementos conterão números.
Linhas 140 e 150 — imprime os cabeçalhos das três colunas.
Linhas 160 a 210 — dois laços aninhados imprimem a informação em colunas e linhas. O laço 'x' conta as linhas, e o laço 'y', as colunas, assim cada elemento é referido como a(x, y).

O próximo programa armazena os nomes de seis alunos e suas notas nos exames de Inglês, Francês e Matemática. Ele imprime essa informação numa tabela e também calcula e imprime as notas totais e média dos alunos.

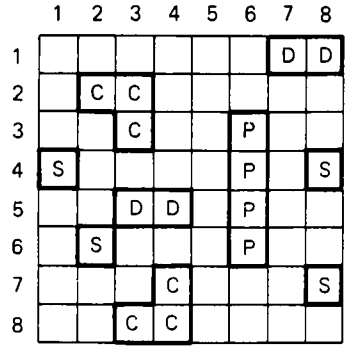
DIGITE:

```
 10  REM  NOTAS DOS ALUNOS
 20  DIM  a (6, 3)
 30  DIM  n$ (6, 7)
 40  FOR  n = 1 TO 6
 50  FOR  m = 1 TO 3
 60  READ  a (n, m)
 70  DATA  42, 52, 61, 12, 23, 34, 67, 72,
63, 78, 82, 91, 53, 62, 58, 73, 69, 85
 80  NEXT  m
 90  NEXT  n
100  FOR  z = 1 TO 6
110  READ  n$ (z)
120  DATA  "Joao", "Jose", "Carlos",
"Ana", "Luiza", "Pedro"
130  NEXT  z
140  PRINT  "NOME        INGLES        FRANCES
MATEMATICA"
150  FOR  x = 1 TO 6
160  PRINT  n$ (x); "    ";
170  LET  t = 0
180  LET  v = 0
190  FOR  y = 1 TO 3
200  PRINT  a (x, y); "          ";
210  LET  t = t + a (x, y)
220  NEXT  y
230  LET  v = t/3
240  PRINT
250  PRINT  "NOTA TOTAL "; t
260  PRINT  "MEDIA "; v
270  PRINT
280  NEXT  x
290  STOP
```

Execute o programa.
Estude o resultado e o programa.
Escreva uma descrição de como o programa funciona.

Podemos agora escrever um programa para jogar 'batalha naval'. 'Batalha Naval' consiste num tabuleiro com 8 colunas, cada coluna com 8 quadrados onde estão secretamente colocados 4 submarinos, 2 destroyers, 2 cruzadores e 1 porta-aviões, como o mostrado no primeiro diagrama da página 125. Repare bem no formato dos navios, pois isso será importante na hora de jogar.

A posição desses navios precisa ser armazenada num array numérico, por isso é preciso trocar as letras por numéros. Assim submarinos serão 1, destroyers 2, cruzados 3 e o porta-aviões 4. Qualquer quadrado sem navio será zero. O diagrama fica como apresentado no segundo diagrama da página 125.



S = submarino
D = destroyer
C = cruzador
P = porta-aviões

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	2	2
2	0	3	3	0	0	0	0	0
3	0	0	3	0	0	4	0	0
4	1	0	0	0	0	4	0	1
5	0	0	2	2	0	4	0	0
6	0	1	0	0	0	4	0	0
7	0	0	0	3	0	0	0	1
8	0	0	3	3	0	0	0	0

0 = sem navio
1 = submarino
2 = destroyer
3 = cruzador
4 = porta-aviões

Essa informação pode agora ser armazenada num array numérico bidimensional, com 64 elementos — 8 colunas e 8 linhas, e nós o chamaremos de 'a(8, 8)'. Isso é feito por laços aninhados, o externo 'p' conta as linhas e o interno 'q' conta as colunas. O interno será executado 8 vezes para cada execução do externo. Enquanto essa informação estiver sendo lida para o array, é preciso contar o número de quadrados diferentes de 0; isso nos dá o número de quadrados que contém navios, o qual será necessário mais tarde para saber quando todos os navios forem afundados e o jogo terminar. A variável 'n' é usada para isso, e cada elemento do array 'a(q, p)' será comparado com 0; se não for igual a zero então 1 é somado ao contador 'n'. Podemos agora escrever a primeira parte do jogo.

DIGITE:

```
 10  REM  BATALHA NAVAL
 20  DIM  a (8, 8)
 30  LET  n = 0
 40  FOR  p = 1 TO 8
 50  FOR  q = 1 TO 8
 60  READ  a (p, q)
 70  IF  a (p, q) < > 0 THEN LET n = n + 1
 80  NEXT  q
 90  NEXT  p
100  DATA  0, 0, 0, 0, 0, 0, 2, 2
110  DATA  0, 3, 3, 0, 0, 0, 0, 0
120  DATA  0, 0, 3, 0, 0, 4, 0, 0
130  DATA  1, 0, 0, 0, 0, 4, 0, 1
140  DATA  0, 0, 2, 2, 0, 4, 0, 0
150  DATA  0, 1, 0, 0, 0, 4, 0, 0
160  DATA  0, 0, 0, 3, 0, 0, 0, 1
170  DATA  0, 0, 3, 3, 0, 0, 0, 0
```

Agora o tabuleiro para o jogo está armazenado no array 'a(8, 8)'; por exemplo, o número 0 está em a(3, 1), e o número 4, em a(3, 6). O contador 'n' armazenou o número de quadrados com navios, agora o computador precisa saber o que significa 1, 2, 3 e 4, pois se você acertar um navio você vai precisar saber que navio acertou,

e um número como resposta não é bom o suficiente. Assim, vamos usar um array de string unidimensional, a$(4, 12), para armazenar essa informação. O array tem quatro elementos consistindo de 10 caracteres cada.

DIGITE:

```
180  DIM  s$ = (4, 12)
190  FOR  b = 1 TO 4
200  READ  s$ (b)
210  DATA  "submarino", "destroyer",
"cruzador", "porta-avioes"
220  NEXT  b
```

O computador agora sabe que s$(1) é um submarino, s$(2) é um destroyer, s$(3), um cruzador e s$(4), um porta-aviões.

Para jogar batalha naval é necessário entrar 2 números, que serão chamados 'x' e 'y' no programa. O primeiro número 'x' refere-se a colunas e 'y', a linhas. Os números devem estar entre 1 e 8; nem menor que 1 nem maior que 8 é permitido. O computador verificará isso, imprimindo a mensagem apropriada quando necessário. Se os números 5 e 6 são entrados então 'a(5, 6)' é um 0 (veja o próximo diagrama), assim o computador avisa que você errou e nós precisamos de outra jogada. Se por outro lado, você entrar 2 e 6 você será avisado de que acertou um submarino (veja o próximo diagrama). (Quando estiver jogando é bom ter uma cópia em branco do tabuleiro, de tal forma que você possa anotar os acertos e erros.)

	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	2	2
2	0	3	3	0	0	0	0	0
3	0	0	3	0	0	4	0	0
4	1	0	0	0	0	4	0	1
5	0	0	2	2	0	4	0	0
6	0	X	0	0	X	4	0	0
7	0	0	0	3	0	0	0	1
8	0	0	3	3	0	0	0	0

a(5, 6) = 0 logo um erro é contado
a(2, 6) = 1 logo um acerto é contado

Vamos escrever essa parte do programa.

DIGITE:

```
230  INPUT  "Atire num navio entrando dois
numeros entre 1 e 8. Pressione ENTER depois de
cada numero"; x, y
240  IF  x < 1 OR x > 8 OR y < 1 OR y >
8 THEN PRINT "Numeros nao permitidos": GOTO
230
250  IF  a (x, y) = 0 THEN PRINT "Voce
errou": GOTO 230
260  PRINT  "Voce acertou um "; s$ (a(x,
y))
```

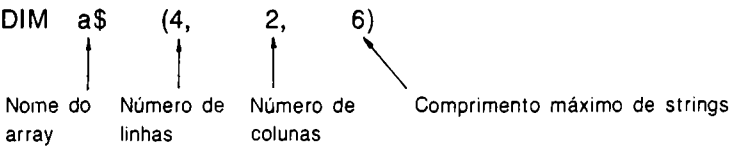
Se você acertar um navio, tudo o que precisa ser feito é tirar 1 do contador 'n' o qual está armazenando o número de quadrados com navios neles. O contador precisa ser então comparado com 0, para ver se há outros navios a serem afundados. Uma vez que 'n' seja igual a 0 o jogo acaba com uma mensagem final.

DIGITE:

```
270  LET  n = n — 1
280  IF  n > 0 THEN GOTO 230
290  PRINT
300  PRINT  "Muito bem, todos os navios
afundados"
310  STOP
```

Agora jogue; você não deverá lembrar-se onde os navios estão com exatidão, mas não olhe nenhum dos diagramas.[1]

Finalmente, vamos nos voltar para arrays de strings bidimensionais. Por exemplo:



É usado da mesma forma que um array numérico bidimensional com laços aninhados para alimentar a informação.

DIGITE:

```
10  REM  NOMES
20  DIM  a$ (4, 2, 6)
30  PRINT  "GAROTAS", "GAROTOS"
40  FOR  n = 1 TO 4
50  FOR  m = 1 TO 2
60  READ  a$ (n, m)
70  DATA  "Ana", "Paulo", "Lucia", "Roberto",
"Sara ou Luiza", "Jaime", "Susana", "Marcos"
80  PRINT  a$ (n, m) ,
90  NEXT  m
```

---

(1) N.T.: Obviamente trata-se de um programa "furado", permitindo, por exemplo, que você ganhe o jogo apenas repetindo seguidamente um par (x, y) que tenha acertado. Por exemplo, entrando (4, 1) dezoito vezes você ganha o jogo, sem, na verdade, ter acertado mais que um submarino. Para impedir isso, poderia acrescentar-se um array numérico bidimensional 'B(8, 8)' que armazenasse as posições tentadas; assim acrescentaríamos as seguintes instruções:

```
21  DIM  B (8, 8)
61  LET  B(p, q) = 1
241  IF  B(x, y) = 0 THEN GOTO 249
242  PRINT  "Voce ja jogou nesta posicao. Tente outra vez"
243  GOTO  230
249  LET  B(x, y) = 1
```

```
100  PRINT
110  NEXT  n
120  PRINT
130  PRINT  "Exemplos do que alguns elementos contem"
140  PRINT  "a$ (1, 1) ="; a$ (1, 1)
150  PRINT  "a$ (3, 2) ="; a$ (3, 2)
160  PRINT  "a$ (4, 1) ="; a$ (4, 1)
170  STOP
```

RUN

Como existem duas colunas neste array, o laço FOR-NEXT 'm' será executado duas vezes, pegando os dois primeiros nomes, toda vez que o laço 'n' for executado. O laço 'n' é executado 4 vezes, já que existem 4 linhas no array. Os nomes são impressos (linha 80) à medida que os laços são executados, mas apenas para mostrar que os nomes estão armazenados no array, enquanto as linhas 140 a 160 imprimem três exemplos.

Atividades

7. Usando o programa anterior, que nomes estão nos elementos a$(1, 2), a$(3, 1) e a$(4, 2)?

8. Usando um array de string bidimensional imprima em duas colunas, com cabeçalhos, a informação dada abaixo:

Nome	Esporte
J. Connors	Tênis
S. Cram	Atletismo
S. Davis	Sinuca
L. Piggott	Hipismo
F. Bruno	Boxe

9. Usando um array numérico bidimensional imprima em três colunas, com cabeçalhos, a informação abaixo:

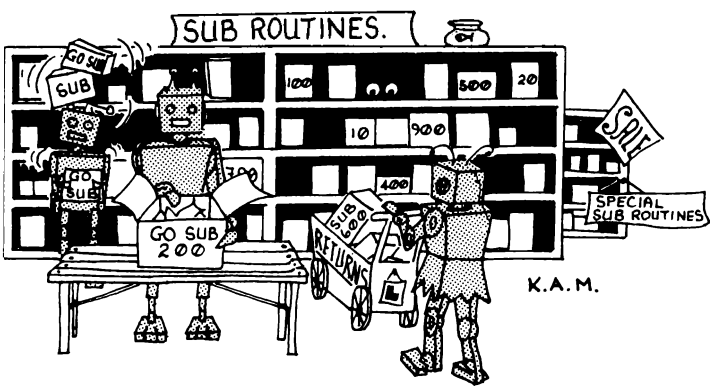
Divisão da riqueza mundial (porcentagem)

Ano	Países desenvolvidos	Países em desenvolvimento
1950	68	32
1960	79	21
1970	82	18

Teste seus programas no computador.

Unidade 25:

GOSUB e RETURN



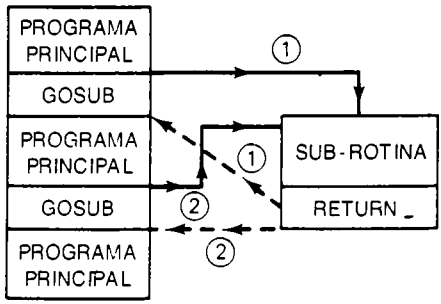
À medida que os programas tendem a ficar mais longos, pode haver diversas linhas do programa que são necessárias mais de uma vez. Não é preciso digitar essas linhas repetidamente, pois elas podem ser agrupadas em **sub-rotinas**, ou **subprogramas**. Uma sub-rotina é, portanto, constituída de várias linhas de programa, formando uma seção do programa principal, a qual pode ser chamada mais de uma vez durante a execução do programa. Para usar ou chamar uma sub-rotina é usado o comando GOSUB,(1) na tecla H. Por exemplo,

```
GOSUB  200
```

Isso significa ir para a sub-rotina que começa na linha 200. Ao final das linhas da sub-rotina deve sempre ser colocado o comando RETURN, na tecla Y. Isso faz o computador voltar para a instrução seguinte ao último GOSUB. Por exemplo, se GOSUB 200 estava na linha 50 do programa, então o comando RETURN fará o compu-

(1) N.T.: O BASIC na verdade não tem o conceito de "sub-rotina", pois não há controle de passagem de parâmetros, como em outras linguagens, tais como PASCAL, ALGOL e até mesmo FORTRAN.

tador voltar o fluxo de execução para a instrução seguinte à linha 50. Veja a ilustração a seguir:



Portanto, quando o computador é mandado a outra parte do programa por um comando GOSUB, ele se lembrará de onde veio e retornará para esse lugar. Sub-rotinas normalmente são colocadas no final do programa principal. Não há palavra especial para indicar o início de uma sub-rotina, por isso é uma boa prática incluir um comando REM nesse ponto. Isso foi feito no programa a seguir.

DIGITE:

```
 10  REM  Uso de sub-rotinas
 20  INPUT  "Entre seu nome "; n$
 30  PRINT  n$
 40  GOSUB  200
 50  INPUT  "Entre a primeira linha do
 endereco "; a$
 60  INPUT  "Entre a segunda linha do
 endereco "; b$
 70  INPUT  "Entre a terceira linha do
 endereco "; c$
 80  PRINT  a$,, b$,, c$
 90  GOSUB  200
100  INPUT  "Entre a idade em anos (so
em numero) "; a
110  PRINT  "Idade é"; a; "anos"
120  GOSUB  200
130  INPUT  "Entre a data do nascimento";
d$
140  PRINT  "A data do nascimento e "; d$
150  GOSUB  200
160  STOP
200  REM  Sub-rotina p/tracar uma linha
210  FOR  j = 1 TO 32
220  PRINT  "—";  <-- (traço, e não espaço)
230  NEXT  j
240  PRINT
250  RETURN
```

RUN, entrando a informação necessária.

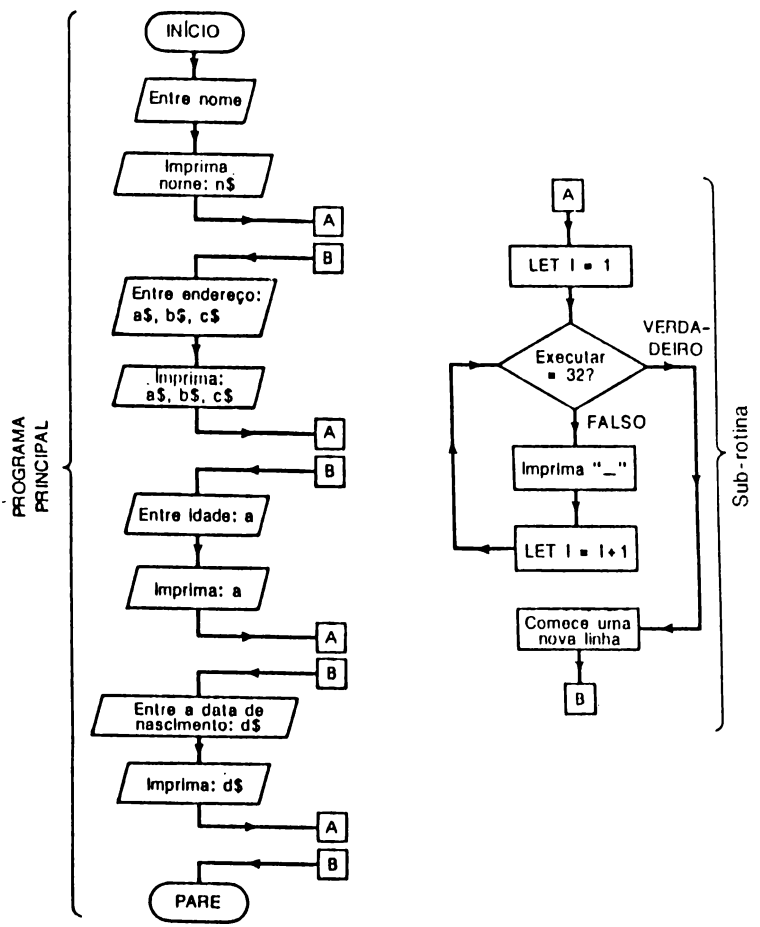
Este programa imprime o nome, endereço, idade e data de nascimento de uma pessoa, mas separa cada parte da informação por uma linha tracejada. Como essa linha é necessária várias vezes, foi criada uma sub-rotina começando na linha 200. A linha 40 chama essa sub-rotina a primeira vez, e quando ela acaba de ser executada, o computador volta para a linha 50. A linha 90 chama a sub-rotina uma segunda vez e dessa vez o computador continua a execução a partir da linha 100. A sub-rotina ainda é chamada 2 vezes, pelas linhas 120 e 150. O comando STOP, na linha 160, é muito importante porque, se não fosse ele estar nessa posição, o computador executaria mais uma vez a sub-rotina, e ao encontrar o comando RETURN ele se confundiria, por não saber para que linha voltar, já que não houve comando GOSUB.

Apague a linha 160.
RUN

O resultado é uma linha tracejada extra, pois o computador cai mais uma vez na sub-rotina e gera a seguinte mensagem de erro.

```
7  RETURN  without GOSUB, 250 : 1
```

O fluxograma do programa mostra que a sub-rotina é desenhada como uma seção separada.



O próximo programa é sobre uma máquina que é operada por um temporizador; liga e desliga a cada 5 segundos. O programa dá uma mensagem que avisa se a máquina está ligada ou desligada. Os intervalos de 5 segundos são controlados por um FOR-NEXT que é escrito numa sub-rotina. O programa principal é executado cinco vezes; isso é contado pela variável 'c'.

DIGITE:

```
 10  REM  Maquina ligada ou desligada
 20  LET  c = 0
 30  GOSUB  100
 40  PRINT  "MAQUINA LIGADA"
 50  GOSUB  100
 60  PRINT  "MAQUINA DESLIGADA"
 70  LET  c = c + 1
 80  IF  c = 5 THEN GO TO 140
 90  GOTO  30
100  REM  Sub-rotina para intervalos
110  FOR  x = 1 TO 900
120  NEXT  x
130  RETURN
140  STOP
```

RUN

Atividades

1. Desenhe o fluxograma para o programa anterior.

Algumas vezes a sub-rotina é chamadá, apenas, se certas condições forem verdadeiras e ignorada caso contrário. O próximo programa é um exemplo disso.

DIGITE:

```
 10  REM  Sub-rotina usada se n for
 4 ou 7
 20  INPUT  "Entre um numero entre
 1 e 10 "; n
 30  IF  n = 4 OR n = 7 THEN GOSUB 100
 40  LET  n = n + 2
 50  PRINT  n
 60  STOP
100  REM  Sub-rotina
110  LET  n = n * n
120  RETURN
```

Antes de executar o programa tente responder:

2. O que será impresso na linha 50 quando os seguintes números forem entrados: (a) 4, (b) 2, (c) 7. Verifique suas respostas executando o programa.

Atividades

3. Qual o valor de x depois de esse programa ser executado se: (a) n = 5, (b) n = 2.

```
10  INPUT  "Entre um numero "; n
20  IF  n < 5 THEN GOTO 40
30  GO SUB 70
40  LET  x = n * n
50  PRINT  x
60  STOP
70  REM  Sub-rotina
80  LET  n = n + z
90  RETURN
```

Desenhe um fluxograma para esse programa.

4. Escreva um programa que pergunte aos usuários quatro questões sobre sua casa. A pergunta e a resposta deverão ser impressas na tela e separadas das seguintes por uma linha de asteriscos, a qual deve ser impressa por sub-rotina.

5. Entre um número entre 1 e 100, 'n'. Se 'n' for menor que 50, mande o computador para uma sub-rotina onde 25 é somado a 'n'. Na volta para o programa principal os números de 0 a 'n' devem ser impressos de cinco em cinco.

6. Escreva um programa de adivinhação de números. Entre um número entre 1 e 100 sem o jogador ver. Se o jogador acertar a tentativa, uma mensagem apropriada deve ser enviada. Caso contrário, o programa deve voltar ao início para nova tentativa. Se a tentativa for muito alta o computador deve ir para uma sub-rotina que imprima esse fato e proponha ao jogador uma nova tentativa. Se, por outro lado, a tentativa for muito baixa, o computador deve ir para uma outra sub-rotina, similar à anterior. O programa principal deve formar um laço infinito.

7. Um robô tem dois conjuntos de instruções para abrir uma garrafa de vinho: um para garrafas com rolha e outro para garrafas com tampa. Entre o tipo de garrafa que o robô deve abrir e mande-o para o conjunto de instruções respectivo. As instruções devem ser impressas na tela. (Duas sub-rotinas são requeridas, uma para cada conjunto de instruções.) Depois de a garrafa ter sido aberta, o computador deve retornar ao programa principal e imprimir as seguintes mensagens:

"Garrafa aberta"
"Posso servir o seu copo?"
"Delicia!!"

Se o tipo errado de garrafa for entrado, o computador deve imprimir uma mensagem apropriada e voltar a pedir o tipo da garrafa.

8. Gasta-se 0.15 litro de tinta para pintar um metro quadrado de parede. Escreva um programa que:

(a) receba como entrada o número de paredes a serem pintadas
(b) receba como entrada a altura e largura de cada parede
(c) calcule a área de cada parede
(d) calcule a área total a ser pintada
(e) calcule a quantidade de tinta necessária — a área total multiplicada por 0.15
(f) imprima a quantidade de tinta necessária.

Infelizmente, todas as medidas são dadas em pés, e a conversão para metros é feita por uma sub-rotina:

metros = pés/3.208

Em BASIC fica m = f/3.208

As seguintes variáveis devem ser usadas:
r = litros de tinta necessários para pintar 1 m²
n = número de paredes a serem pintadas
f = comprimento da parede em pés } também usada na sub-rotina
f = altura da parede em pés
a = área de cada parede (1 * h)
t = área total de todas as paredes (t + a)
m = comprimento da parede em metros } usada na sub-rotina
m = altura da parede em metros
l = comprimento da parede em metros } usada no programa principal
h = altura da parede em metros
p = quantidade de tinta requerida (t * r)
w = variável para contar o número de paredes.

O exemplo final desta unidade mostra como até o comando RETURN pode estar sob condicionantes. Na linha 320, o computador retorna ao programa principal apenas se c$(1) = t$(x). Caso contrário, o comando RETURN é ignorado, e uma outra parte da sub-rotina é executada, até o computador encontrar um outro comando RETURN na linha 350.

DIGITE:

```
 10  REM  Distancias entre cidades
 20  DIM  t$ (7, 10)
 30  DIM  d (7, 7)
 40  DIM  c$ (1, 10)
 50  FOR  n = 1 TO 7
 60  READ  t$ (n)
 70  DATA  "London", "Taunton", "Plymouth",
"Nottingham", "Liverpool", "Leeds", "Hull"
 80  NEXT  n
 90  FOR  m = 1 TO 7
100  FOR  s = 1 TO 7
110  READ  d (m, s)
120  DATA  0, 144, 211, 122, 197, 190, 168, 144, 0,
74, 180, 203, 237, 247, 211, 74, 0, 254, 278, 312, 321,
122, 180, 254, 0, 97, 67, 73, 197, 203, 278, 97, 0, 73,
128, 190, 237, 312, 67, 73, 0, 55, 168, 247, 321, 73,
128, 55, 0
130  NEXT  s
140  NEXT  m
150  INPUT  "Entre a cidade para onde voce
vai "; c$ (1)
160  GOSUB  300
170  IF  x = 0 THEN PRINT "Sem informacoes
sobre "; c$ (1): GOTO 150
180  LET  y = x
```

```
190  INPUT  "Entre a cidade de onde voce
vem "; c$ (1)
200  GOSUB  300
210  IF  x = 0 THEN PRINT "Sem informacoes
sobre "; c$ (1): GOTO 190
220  PRINT  "A distancia entre "; t$ (y);
" e "; t$ (x); " é "; d (y, x); " milhas"
230  STOP
300  REM  Sub-rotina
310  FOR  x = 1 TO 7
320  IF  c$ (1) = t$ (x) THEN RETURN
330  NEXT  x
340  LET  x = 0
350  RETURN
```

Execute o programa diversas vezes, tentando diferentes cidades.

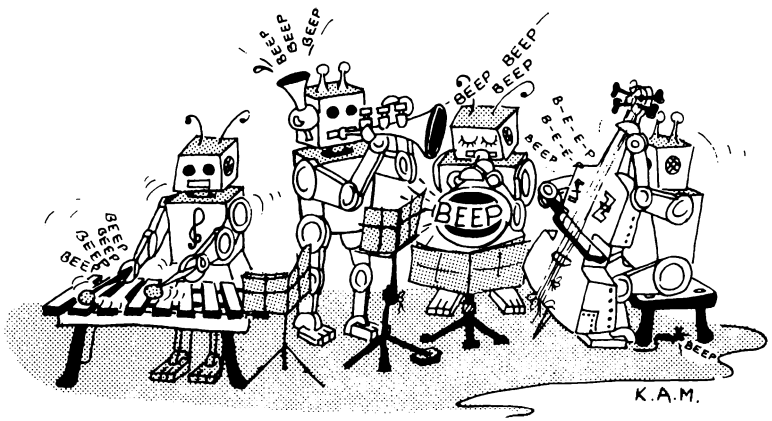
Todas as técnicas usadas nesse programa, exceto para os comandos GOSUB e RETURN, já foram usadas nas unidades anteriores; portanto estude cuidadosamente o programa e:

Atividade

9. Escreva uma explicação de como o programa acima funciona.

Unidade 26:

Tocando Música



Como você pode estar sabendo, o TK90X tem um alto-falante embutido. Pode-se produzir uma nota simples usando o comando BEEP, abaixo da tecla Z (modo estendido e symbol shift). O programa a seguir lhe dá a oportunidade de ouvir o alto-falante em ação.

DIGITE:

```
10  FOR  n = 50 TO – 30 STEP – 1
20  BEEP  0.25, n
30  NEXT  n
```

RUN NÃO REMOVA

Repare a tecla 20. O comando BEEP, que produz a nota musical, é seguido de dois números. O primeiro é a **duração** da nota; 1 significa um segundo. No caso 0.25 produzirá uma nota com um quarto do segundo de duração. O segundo número significa a nota **dentro da escala**; aqui vai de 50, que é muito aguda, a – 30, que é muito grave. Assim, quão mais alto é o número, mais aguda a nota. O programa toca 80

notas ao todo, cada uma por 0.25 segundo, descendo na escala uma nota por vez. Pode-se facilmente inverter o programa, subindo as notas na escala.

Altere a linha 10

```
10  FOR  n = -30 TO 50
```

RUN NÃO REMOVA

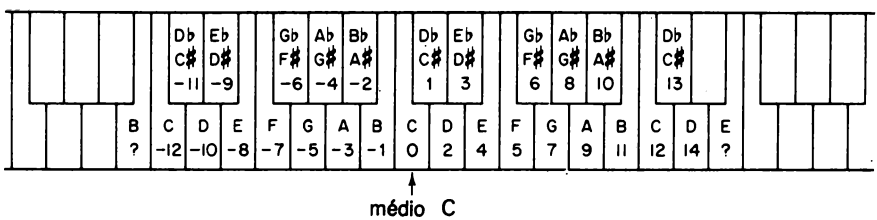
Pode-se também tocar as notas alternadamente, usando o comando STEP com valores diferentes.

Altere a linha 10

```
10  FOR  n = -30 TO 50 STEP 2
```

RUN. Experimente outros valores.

Até aqui produzimos séries de notas mas não sabemos que notas estamos tocando. Para isso precisamos observar as teclas do piano para ver que números as notas têm. Não temos espaço para mostrar todas as teclas, assim o diagrama mostra aquelas próximas do C médio, a nota que está quase no centro do teclado.



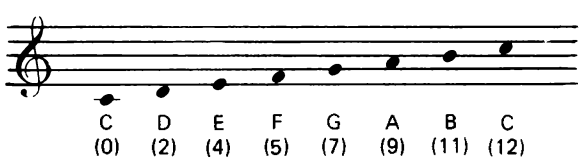
♭ = bemol

♯ = sustenido

Como você pode ver, o C médio é 0; as notas acima têm números positivos, e as abaixo têm números negativos. (Os números positivos têm sinal de mais, embora não seja usual escrevê-lo, e os números negativos levam sinal de menos.)

Atividade

1. Que números o E mais acima e o B mais abaixo no diagrama deveriam receber?



Escala do C maior

É bom ter os números nas teclas do piano, mas se você não for músico não saberá onde estão essas notas numa partitura. O diagrama do final da página anterior deverá ajudar. Ele mostra a tecla de C maior (sem bemol ou sustenido) começando no **C** médio e terminando no próximo C. A escala consiste de 8 notas, 1 oitava.

O nome da nota está escrito abaixo da nota e o respectivo número para o computador está escrito entre parênteses. O programa seguinte toca essa escala, cada nota com meio segundo de duração.

DIGITE:

```
    10  REM  Escala do C maior
    20  BEEP  .5, 0: BEEP .5, 2: BEEP
.5, 4: BEEP .5, 5: BEEP .5, 7: BEEP
.5, 9: BEEP .5, 11: BEEP .5, 12
    30  STOP
```

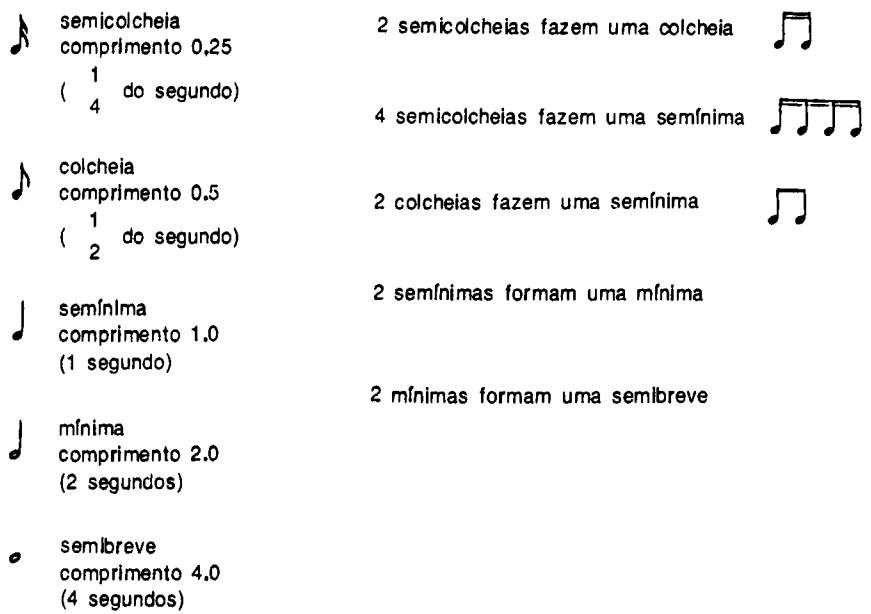
RUN NÃO REMOVA

Note que cada comando BEEP é separado do próximo por dois-pontos (:), já que cada um é um novo comando.

Atividades

2. Altere o programa para ele tocar a escala três vezes.

Diagrama mostrando diferentes durações de notas



Um ponto após a nota significa uma duração de uma vez e meia à original. Por exemplo:

3. Adicione uma pausa para ter um intervalo de 3 segundos entre cada vez que a escala é tocada.

NÃO REMOVA

Estamos quase prontos para realmente tocar algum trecho de música, mas ainda estamos lidando com notas com a mesma duração, o que raramente acontece. Agora precisamos conhecer notas de diferentes durações, daí ser necessária uma outra lição de música. Veja o diagrama da página anterior, que mostra os diversos comprimentos de notas (há ainda outras adicionais, mas não precisamos conhecê-las neste momento).

Retorne ao programa da escala C maior.

Altere a linha 20

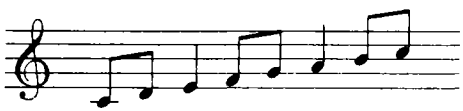
```
20  BEEP  .5, 0: BEEP .25, 2: BEEP
.5, 4: BEEP .25, 5: BEEP .5, 7: BEEP
.25, 9: BEEP .5, 11: BEEP .25, 12
```

RUN NÃO REMOVA

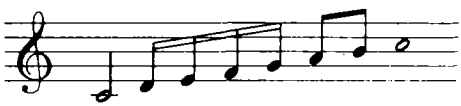
Agora o programa toca a primeira nota como colcheia, a segunda como semicolcheia, a terceira como colcheia, e assim por diante.

Atividade

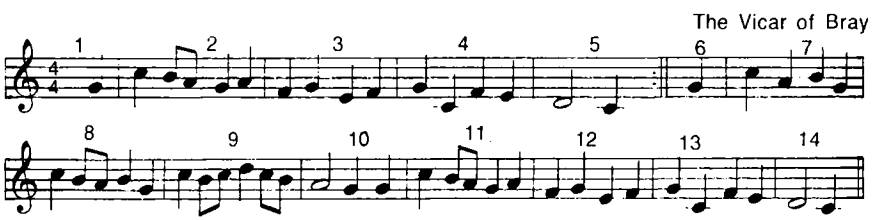
4. Usando o programa da escala C maior:
(a) toque todas as notas na escala em semicolcheia
(b) toque a escala como a seguir:



(c) toque a escala como a seguir:



Agora vamos tentar tocar uma música — **The Vicar of Bray.**(1) Está escrita na escala de C maior, mas usa uma nota extra, D maior, a qual, se você olhar novamente o diagrama das teclas do piano, é a número 14. Cada compasso foi numerado e corresponde a uma linha no programa, de tal forma que você pode ver o que está acontecendo. Por exemplo, o compasso 3 está na linha 230, e o 9, na linha 70. Os dois pontos no final do compasso 5 significam voltar ao início.



(1) N.T.: Mantido no original.

Usando o diagrama da escala C maior da página 137 podemos escrever o número de cada nota.

Compasso 1	Compasso 2	Compasso 3	Compasso 4	Compasso 5	Compasso 6
7	12, 11, 9, 7, 9	5, 7, 4, 5	7, 0, 5, 4	2, 0	7

← Repita →

Compasso 7	Compasso 8	Compasso 9	Compasso 10	Compasso 11
12, 9, 11, 7	12, 11, 9, 11, 7	12, 11, 12, 14, 12, 11	9, 7, 7	12, 11, 9, 7, 9

← ←

Compasso 12	Compasso 13	Compasso 14
5, 7, 4, 5	7, 0, 5, 4	2, 0

— Mesmo que nos compassos 1 a 5 →

O próximo ponto a ser notado é que uma seção da partitura é tocada três vezes, compassos 1 a 5. Esses compassos são tocados duas vezes no início e então se você olhar com cuidado verá que da última nota do compasso 10 até o fim do compasso 14, é a mesma coisa que os compassos 1 a 5. Isso significa que essa seção pode ser colocada numa sub-rotina, logo digitada uma única vez. Nós precisamos agora calcular o comprimento de cada nota e colocar isso na nossa partitura musical para o computador.

	Compasso 1	Compasso 2	Compasso 3	Compasso 4	Compasso 5	Compasso 6
P	7	12, 11, 9, 7, 9	5, 7, 4, 5	7, 0, 5, 4	2, 0	7
D	1	1, .5, .5, 1, 1	1, 1, 1, 1	1, 1, 1, 1	2, 1	1

	Compasso 7	Compasso 8	Compasso 9	Compasso 10	Compasso 11
P	12, 9, 11, 7	12, 11, 9, 11, 7	12, 11, 12, 14, 12, 11	9, 7, 7	12, 11, 9, 7, 9
D	1, 1, 1, 1	1, .5, .5, 1, 1	1, .5, .5, 1, .5, .5	2, 1, 1	1, .5, .5, 1, 1

	Compasso 12	Compasso 13	Compasso 14
P	5, 7, 4, 5	7, 0, 5, 4	2, 0
D	1, 1, 1, 1	1, 1, 1, 1	2, 1

P = Nota
D = Duração (comprimento)

DIGITE:

```
                10  REM  The Vicar of Bray
                20  GOSUB  200
                30  GOSUB  200
Compasso 6      40  BEEP  1, 7
Compasso 7      50  BEEP  1, 12: BEEP 1, 9: BEEP 1, 11: BEEP 1, 7
Compasso 8      60  BEEP  1, 12: BEEP .5, 11: BEEP .5, 9: BEEP 1, 11:
                BEEP 1, 7
Compasso 9      70  BEEP  1, 12: BEEP .5, 11: BEEP .5, 12: BEEP 1, 14: BEEP
                .5, 12: BEEP .5, 11
```

```
Compasso 10     80  BEEP  2, 9: BEEP 1, 7     (apenas duas notas, pois a outra G
                90  GOSUB  200                 (7) está no começo da sub-rotina)
               100  STOP
               200  REM  Sub-rotina
Compasso 1     210  BEEP  1, 7
Compassos 2    220  BEEP  1, 12: BEEP .5, 11: BEEP
e 11           .5, 9: BEEP 1, 7: BEEP 1, 9
Compassos 3    230  BEEP  1, 5: BEEP 1, 7: BEEP 1,
e 12           4: BEEP 1, 5
Compassos 4    240  BEEP  1, 7: BEEP 1, 0: BEEP 1,
e 13           5: BEEP 1, 4
Compassos 5    250  BEEP  2, 2: BEEP 1, 0
e 14           260  RETURN
```

RUN

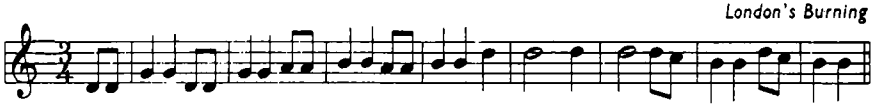
A velocidade pode parecer um pouco lenta. Isso é porque nós tomamos uma semínima

para ser de um segundo de duração por conveniência; poderia ser um pouco menor, provavelmente 0.8 ao invés de 1. No entanto, é mais simples manter a semínima com o 1, pois facilita o cálculo dos valores das outras notas.

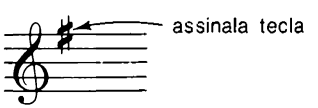
Atividade

5. Escreva um programa para tocar **London's Burning.**[1] A música é fornecida abaixo. Note que cada pequena frase é repetida. Assim, para evitar digitação repetida, use laços FOR-NEXT.

London's Burning



Vamos ver uma tecla musical que usa sustenido (#). Muitas músicas são escritas na tecla G maior que tem F sustenido; qualquer nota escrita nessa tecla se parecerá com



Essa tecla começa e termina em um G e F # é tocada ao invés da F ordinária. Se você voltar a olhar o diagrama das teclas do piano, verá que os seguintes números tocariam essa escala começando no primeiro G abaixo do C médio.

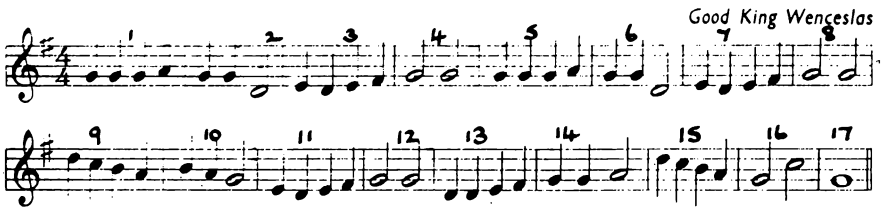
```
                ↓ C médio
G   A   B  C  D  E  F#G
–5, –3, –1, 0, 2, 4, 6,  7
```

(1) N.T.: Mantido no original.

Atividades

6. (a) Digite um programa para tocar a escala do G maior.
(b) Escreva os números da próxima oitava da escala começando no primeiro G acima do C médio.
(c) Adicione ao seu programa a segunda oitava da escala.

7. Escreva um programa para **Good King Wenceslas**[1]; a música é dada a seguir. Note a tecla assinalada. Apenas as notas usadas na atividade 6 são necessárias. Note que os primeiros quatro compassos são repetidos e compassos 9 e 15 são também o mesmo. Você consegue um jeito de evitar digitar isso de novo?



Digitar uma frase musical é chato principalmente por causa de todos os comandos BEEP necessários. O próximo programa usa comandos READ e DATA num laço FOR-NEXT que elimina todos os BEEP's exceto um!

DIGITE:

```
10  REM  Nursery rhyme
20  FOR  n = 1 TO 38
30  READ  a, b
40  BEEP  a, b
50  NEXT  n
60  DATA  1, 0, 1, 0, 1, 7, 1, 7, .5, 9, .5, 11,
. 5, 12, .5, 9, 2, 7, 1, 5, 1, 5, 1, 4, 1, 4, 1, 2,
1, 2, 2, 0, 1, 7, .5, 7, .5, 7, 1, 5, .5, 5, .5, 5,
1, 4, .5, 4, .5, 4, 1 . 5, 2, .5, 2, 1, 7, .5, 7,
.5, 7, .25, 5, .25, 7, .25, 9, .25, 5, 1, 4, .5, 2,
.5, 2, 2, 0
70  STOP
```

RUN. Você sabe as notas? NÃO APAGUE

São 38 notas, e por isso o computador executa o laço FOR-NEXT 'n' 38 vezes, tocando notas diferentes a cada vez. O comando READ pega os primeiros dois valores do comando DATA, que são 1 e 0, e assim a primeira nota é uma semínima no C médio. Da segunda vez, os valores são 1 e 0, o terceiro e quarto valor no comando DATA. Da terceira vez que o laço é executado, 1 e 7 são lidos. Assim, a cada vez dois valores novos são pegos no comando DATA, produzindo uma nota. Isso continua até que todos os DATA sejam usados.

---

(1) N.T.: Mantido no original.

Adicione ao programa:

```
15  FOR  m = 1 TO 2
65  NEXT  m
```

RUN. O que acontece e por quê?

Uma mensagem de erro resulta, pois o computador tentou ler mais dados do que havia no comando DATA, pois foram esgotados p/m = 1. É possível fazer o computador tocar a música duas vezes, usando o comando RESTORE, acima da tecla S (modo estendido).

ADICIONE:

```
63  RESTORE
```

Execute o programa novamente. NÃO APAGUE.

RESTORE é uma instrução que diz ao computador para começar a ler (READ) do início da lista do comando DATA. Podemos separar dois versos de uma rima usando o comando PAUSE. (Lembre-se que PAUSE 50 é 1 segundo.)

Adicione ao programa:

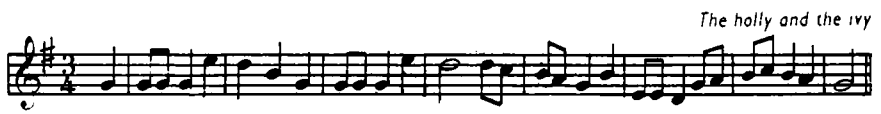
```
64  PAUSE  3 * 50
```

RUN

Temos agora um intervalo de 3 segundos entre dois versos.

Atividade

8. Escreva um programa usando READ e DATA para quatro versos do **The Holly and the Ivy**,[1] com intervalo de 2 segundos entre os versos.



Usando a função INKEY$, como fizemos na Unidade 23, podemos fazer o teclado do micro funcionar como se fosse um teclado de piano. O programa final usa as teclas 1 a 8 para as notas da escala de C maior, começando com C na tecla 1, D na tecla 2, e assim por diante. O comprimento das notas continua o mesmo, assim a variável 'a' é usada para armazenar esse valor.

DIGITE:

```
10  REM  Piano
20  LET  a = .25
30  IF  INKEY$ = "1" THEN BEEP a, 0
40  IF  INKEY$ = "2" THEN BEEP a, 2
```

[1] N.T.: Mantido no original.

```
 50  IF  INKEY$ = "3" THEN BEEP a, 4
 60  IF  INKEY$ = "4" THEN BEEP a, 5
 70  IF  INKEY$ = "5" THEN BEEP a, 7
 80  IF  INKEY$ = "6" THEN BEEP a, 9
 90  IF  INKEY$ = "7" THEN BEEP a, 11
100  IF  INKEY$ = "8" THEN BEEP a, 12
110  GOTO  30
```

RUN. Aperte qualquer tecla entre 1 e 8. Se você segurar a tecla, a nota apenas se repetirá, mas não produzirá uma nota mais longa. Você consegue tocar alguma música?

Tente esta: 1, 3, 2, 4, 3, 5, 3, 1
1, 3, 2, 4, 3, 1
1, 3, 2, 4, 3, 5, 3, 1
6, 2, 4, 3, 1

A única forma de parar o programa é apertar CAPS SHIFT e BREAK.

Unidade 27:

Algumas Funções Úteis - RND, RAND e INT

Random, abreviada como RND, é uma função que produz uma seleção aleatória de números dentre uma lista de 65536 números.

DIGITE:

```
10  REM  Numeros Randomicos
20  FOR  n = 1 TO 10
30  PRINT  RND
40  NEXT  n
50  STOP
```

RUN.

Anote os três primeiros números que aparecem na tela. NÃO APAGUE.

O resultado é uma lista de 10 números, variando de 0 a 0.99999999; você nunca conseguirá o número 1.

Execute o programa novamente — a lista de números deve ser diferente da anterior —, compare com os números que você anotou.

Outra lista foi produzida e, novamente, nenhum número está abaixo de 0 e nenhum acima de 0.99999999. Os números parecem bastante aleatórios, mas na verdade o computador está seguindo uma seqüência, embora seja difícil determinar qual seja. No entanto, podemos, através da função RANDOMIZE, abreviada por RAND (tecla T), fazer com que o computador pegue sempre o mesmo grupo, começando sempre do mesmo ponto da seqüência. Por exemplo, RAND 35 começará a pegar os números a partir do 35º da seqüência.

Adicione ao programa:

```
15  RAND  35 (note que a palavra é escrita por inteiro)
```

RUN. Anote os 3 primeiros números.
Execute novamente. Verifique que os números são os mesmos.

Você pode ver que RAND mantém fixo o ponto de partida da seqüência. Se RND é usado sozinho, o computador começará a seqüência de um lugar qualquer.

Você pode estar pensando que os números de 0 a quase 1 não são muito úteis e raramente são usados, e que normalmente um intervalo muito maior de números é necessário. Podemos produzir números maiores multiplicando RND por outro número. Por exemplo:

```
 6 * RND gera números de 0 a 5.999999
64 * RND gera números de 0 a 63.999999
```

DIGITE:

```
10  REM  Numeros Randomicos
20  PRINT  "6 * RND", "64 * RND"
30  FOR  n = 1 TO 10
40  PRINT  6 * RND, 64 * RND
50  NEXT  n
60  STOP
```

Execute o programa diversas vezes. Note que o número 6 nunca aparecerá na primeira coluna, e o número 64 nunca aparecerá na segunda coluna.

Até aqui todos os números que surgiram na tela pareceram complicados por causa das diversas casas decimais. Usualmente trabalhamos apenas com números inteiros, daí precisamos de uma função que trunque as casas decimais desnecessárias. A função que usaremos é INT, escrita acima da tecla R (modo estendido). INT é abreviatura para **inteiro**, de números inteiros. INT trunca todas as casas decimais. Por exemplo:

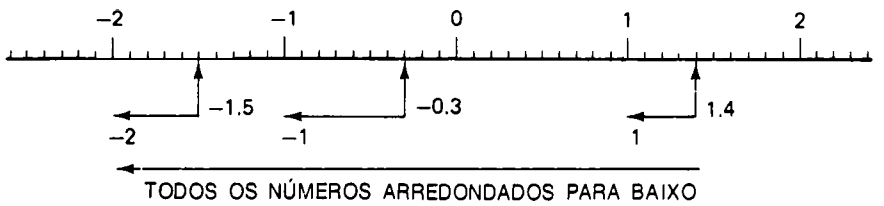
```
INT  1.4809143  fica 1
INT  5.9621567  fica 5
INT  62.5132789 fica 62
```

Note que todos os números foram arredondados p/baixo, mesmo os que estavam muito próximos do inteiro mais acima; 5.9621567 se torna 5, apesar de estar mais próximo de 6. Isso se aplica também p/números negativos.

```
INT  −1.2678957   se torna  −2
INT  −10.6723142  se torna  −11
```

Não pense que esses números foram arredondados p/cima, pois lembre-se de que – 2 é menor que – 1 e – 11 é menor que – 10.

Veja o eixo numérico abaixo para se convencer de que todos esses números foram arredondados para baixo.



O próximo programa prova este fato:

DIGITE:

```
10  REM  Inteiros
20  PRINT  INT  5.6
30  PRINT  INT  62.9
40  PRINT  INT  1.4
50  PRINT  INT  – 1.2
60  PRINT  INT  – 10.6
70  STOP
```

RUN

O resultado é:

```
 5
62
 1
–2
–11
```

Atividade

1. Quais são os valores produzidos:

   (a) INT 4.5
   (b) INT 103.9
   (c) INT 99.1
   (d) INT – 5.6
   (e) INT – 87.1
   (f) INT – 0.87
   (g) INT 0.23

Vamos usar agora INT E RND juntos para produzir números inteiros randômicos.

DIGITE:

```
10  REM  Numeros inteiros randomicos
20  PRINT  "INT (– 10 * RND)", "INT
(10 * RND)"
30  FOR  n = 1 TO 10
40  PRINT  INT (– 10 * RND), INT (10
* RND)
50  NEXT  n
60  STOP
```

Execute o programa diversas vezes. Note que 0 nunca pode aparecer na primeira coluna de números negativos e que 10 nunca pode aparecer na segunda coluna de números positivos.

Produzir números inteiros randômicos pode ser útil para escrever programas de jogos. Nas unidades 19 e 25 você escreveu um programa para adivinhar números, onde o número a ser adivinhado devia ser colocado enquanto o programa era executado. Mas seria muito melhor se o programa gerasse, sem ninguém tomar conhecimento, seu próprio número. Usando INT e RND vamos incrementar esse jogo.

DIGITE:

```
10  REM  Adivinhar um numero
20  LET  c = 1
30  LET  x = INT (21 * RND)
40  PRINT  "Adivinhe o numero entre 0 e 20"
50  INPUT  "Entre um numero "; n
60  IF  n = x THEN GOTO 100
70  LET  c = c + 1
80  PRINT  n; " Errou, tente outra vez"
90  GOTO  50
100 PRINT  "Acertou, voce fez tentativas "; c; " tentativas"
110 STOP
```

Execute o programa diversas vezes.

A linha 30 produz um número entre 0 e 20 o qual é então comparado com 'n', o número entrado na linha 50.

Atividade

2. Por que a linha 30 não foi escrita como?

```
30  LET  x = INT (20 * RND)
```

Algumas vezes, num conjunto de números randômicos, 0 não é desejado, mas o próximo número maior sim. Por exemplo, só os números 1 a 6 são obtidos aleatoriamente, como no jogo de dados, INT (6 * RND) produz de 0 a 5 e INT (7 * RND) produz de 0 a 6, nenhum dos dois sendo o ideal. Para superar esse inconveniente, usamos INT (6 * RND) + 1; adicionando-se 1, passamos o 0 a 1 e o 5 a 6, obtendo o intervalo de 1 a 6.

DIGITE:

```
10  REM  Numeros randomicos de 1 a 6
20  FOR  n = 1 TO 10
30  PRINT  INT (6 * RND) + 1
40  NEXT  n
50  STOP
```

Execute o programa diversas vezes, observando que os números gerados ficam entre 1 e 6.

O próximo programa usa essa idéia para contar o número de vezes que o número 3 ocorre quando um dado é jogado 100 vezes.

DIGITE:

```
 10  REM  Jogando um dado 100 vezes
 20  LET  c = 0
 30  FOR  x = 1 TO 10
 40  FOR  y = 1 TO 10
 50  LET  d = INT (6 * RND) + 1
 60  PRINT  d; " ";
 70  IF  d = 3 THEN LET c = c + 1
 80  NEXT  y
 90  PRINT
100  NEXT  x
110  PRINT
120  PRINT  "O numero de vezes que 3 apareceu foi "; c
130  STOP
```

RUN. Verifique a resposta.
NÃO REMOVA.

Nenhuma explicação para esse programa é necessária, já que todas as técnicas aplicadas foram usadas antes.

Atividades

3. Modifique o programa anterior de tal forma que os usuários possam entrar o número que eles desejarem que seja contado.

4. Escreva um programa que imprima uma lista de 20 números inteiros aleatórios entre 0 e 100.

5. Escreva um programa que imprima uma lista de 15 números inteiros randômicos entre 1 e 20.

6. Escreva um programa que gere sempre a mesma seqüência de 10 números aleatórios entre 0 e 10.

7. Escreva um programa que imprima 100 números inteiros aleatórios de 1 a 5 em 10 colunas de 10 linhas e que calcule e imprima quantas vezes cada número ocorre.

8. Escreva um programa que teste a tabuada de multiplicação entre 2 e 12. A multiplicação deve ser gerada aleatoriamente. Os usuários devem ser avisados se acertaram; se erraram, a resposta certa deve ser mostrada.

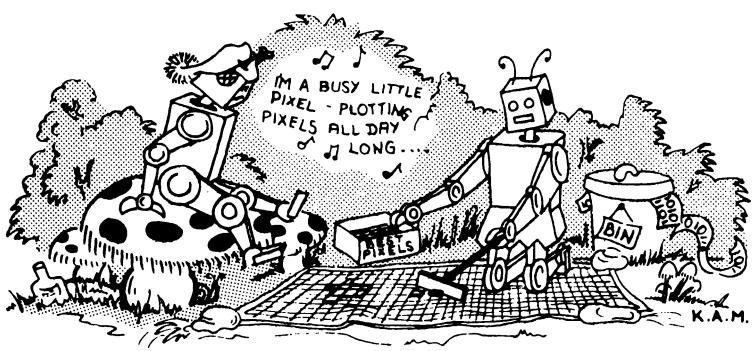
9. (a) Escreva um programa para jogar dois dados 50 vezes e imprimir o número total de vezes em que ocorre o 7.

(b) Adicione um outro laço de tal forma que o programa seja repetido 5 vezes e a cada vez o total é acumulado para fornecer ao final um total geral. O programa deve terminar imprimindo o número médio de vezes que o 7 apareceu.

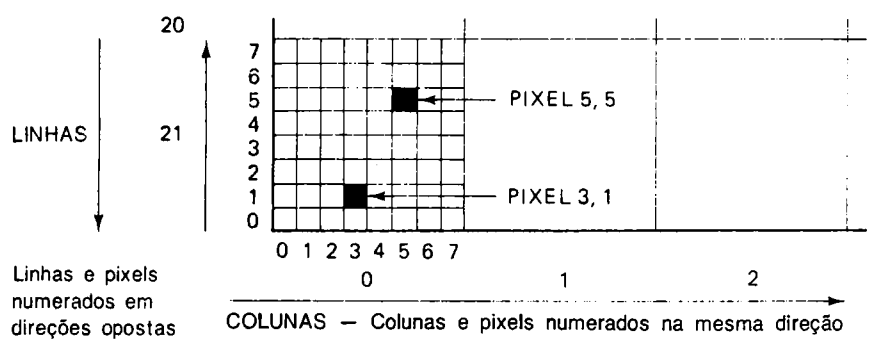
10. Escreva um programa que conte o número de vezes que caras e coroas aparecem quando uma moeda é jogada 80 vezes. Faça cara igual a 0 e coroa igual a 1. O resultado deve ser impresso.

Unidade 28:

Gráficos de Alta-Resolução - PLOT, DRAW, CIRCLE e INVERSE



Gráficos com alta-resolução envolvem o uso de **pixels**. Um pixel é um ponto pequeno.(1) Cada caractere na tela é composto por oito pixels de largura e oito de altura. O diagrama a seguir, o qual foi aumentado, mostra a posição 21,0 dividida



(1) N.T.: PIXEL — Mantido no original em Inglês — é a menor unidade endereçável da tela. É um termo comum no meio técnico, usado para designar cada ponto da tela que possa ser ativado.

em 8 × 8 quadrículas — cada quadrícula sendo um pixel. Dois exemplos de pixels foram marcados.

Existem 32 (0-31) colunas e assim existem 32 × 8 = 256 posições pixel horizontais na tela; estas são numeradas na mesma direção das colunas. Existem 22 (0-21) linhas, logo 22 × 8 = 176 pixels verticais na tela. Estes são numerados na direção contrária às linhas, pois a posição 0 é no final da tela. Veja o diagrama 28.1, o qual pode ser usado como guia para localizar pixels. Quando referenciar um pixel, use coordenadas 'x' e 'y', como ao plotar pontos num gráfico cartesiano. 'x' representa a posição horizontal, lembrando que o 0 está no lado esquerdo da tela. 'y' representa a posição segundo o eixo vertical, lembrando que o 0 está no final (embaixo) da tela. A coordenada 'x' é sempre dada primeiro, logo os pixels desenhados no diagrama anterior estão em 3,1 e 5,5. Obviamente as quadrículas não são visíveis na tela, assim antes de plotar pixels ou desenhar linhas é preciso ter uma cópia do diagrama 28.1.

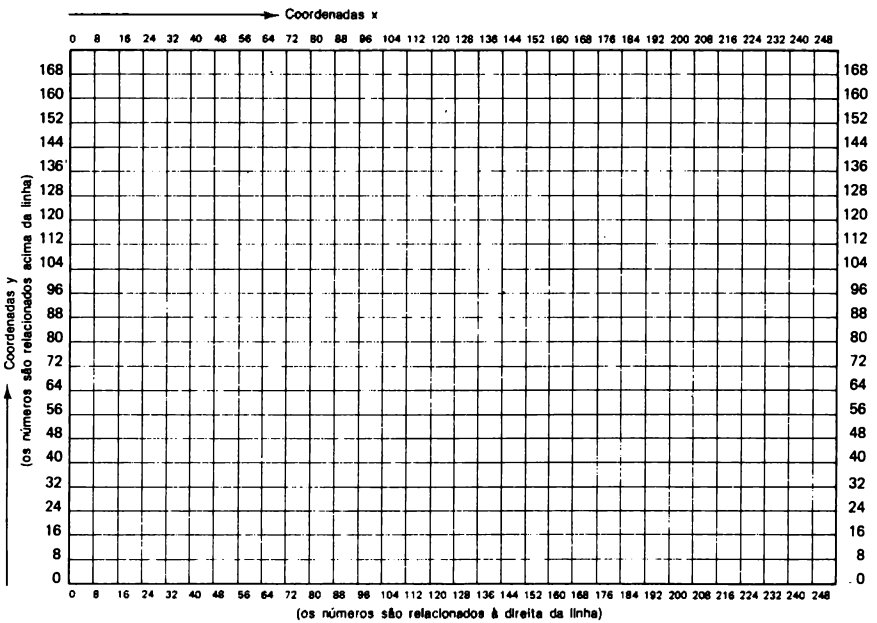
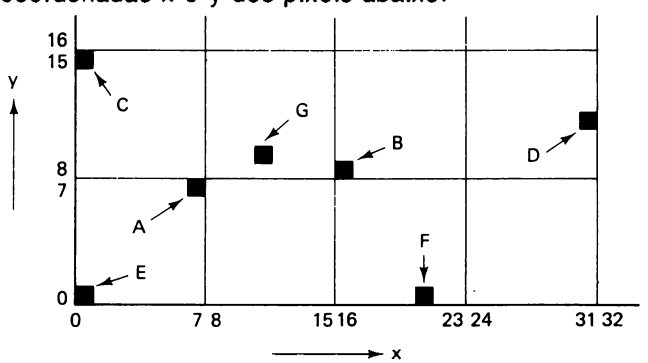


Diagrama 28.1 Diagrama mostrando a numeração dos pixels.

Atividade

1. Dê as coordenadas x e y dos pixels abaixo:



Para ativar pixels na tela usamos o comando PLOT (tecla Q).

DIGITE:

```
10  REM  Ativando pixels
20  INPUT  "Entre a coordenada x (0 a 255) "; x
30  INPUT  "Entre a coordenada y (0 a 175) "; y
40  PLOT  x, y
50  GOTO  20
```

RUN.

Entre números e veja os pixels serem ativados. Note como são pequenos.

Se você entrar uma coordenada x maior que 255 ou uma coordenada y maior que 175, o programa produzirá a mensagem de erro:

B Integer out of range, 40 : 1

para impedir isso, adicione

```
33  IF  x > 255 THEN LET x = 255
36  IF  y > 175 THEN LET y = 175
```

Isso impedirá o pixel de cair fora da tela.
Repare que você não pode ter coordenadas negativas.
Entre STOP p/parar o programa.

No próximo programa o computador ativa 200 pixels aleatoriamente na tela; é como uma pessoa tonta.

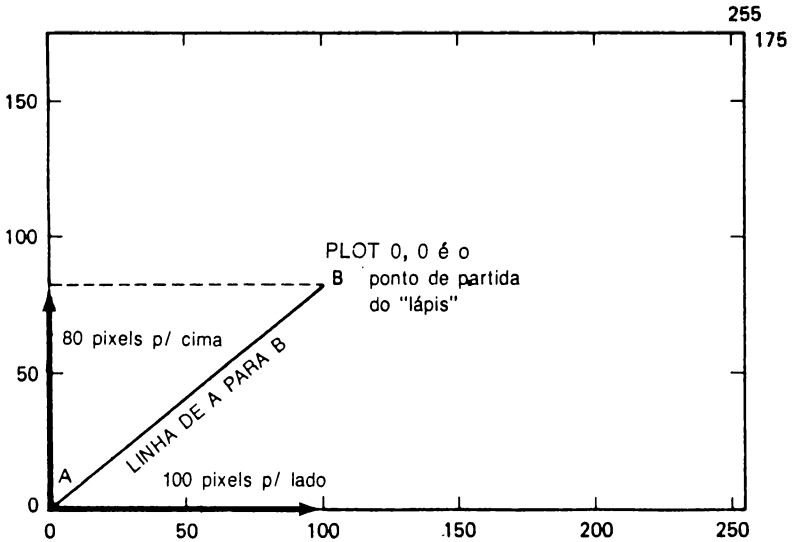
DIGITE:

```
10  REM  Desenhando ao acaso
20  FOR  p = 1 TO 200
30  LET  x = INT (256 * RND)
40  LET  y = INT (176 * RND)
50  PLOT  x, y
60  NEXT  p
70  STOP
```

RUN

Usando PLOT apenas não é muito útil, a menos que estejamos desenhando um gráfico, mas precisamos entender o comando para podermos desenhar linhas. Por exemplo, quando desenhamos uma linha precisamos ativar o ponto onde o 'lápis' começará a traçar. O comprimento e direção da linha são dados incrementando-se os números dos pixels através da tela, e usando o comando DRAW (tecla W).

Exemplo 1: Desenhando uma linha de A para B

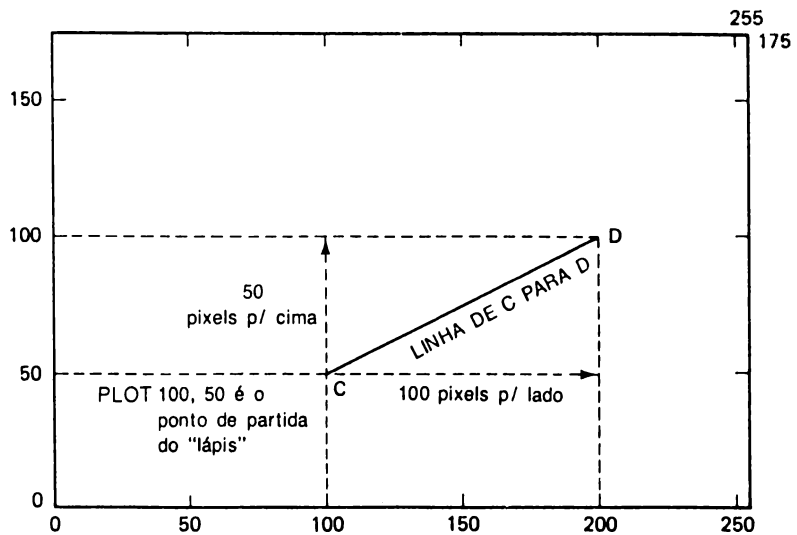


DIGITE:

```
10  PLOT  0, 0
20  DRAW  100, 80
```

RUN

Exemplo 2: Desenhando uma linha de C para D

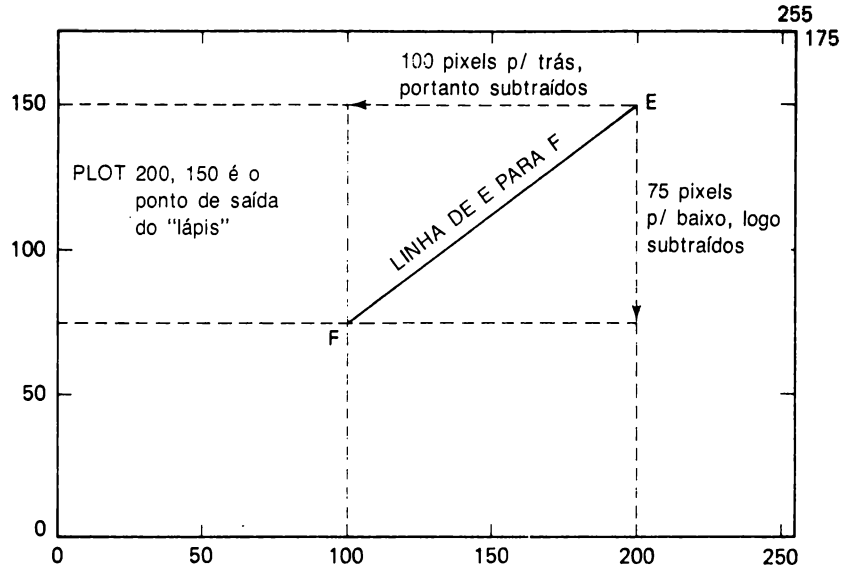


DIGITE:

```
10  PLOT  100, 50
20  DRAW  100, 50 (note que este não é o ponto final da linha, o qual se
                   ativado seria 200, 100)
```

RUN

Exemplo 3: Desenhando a linha de E para F



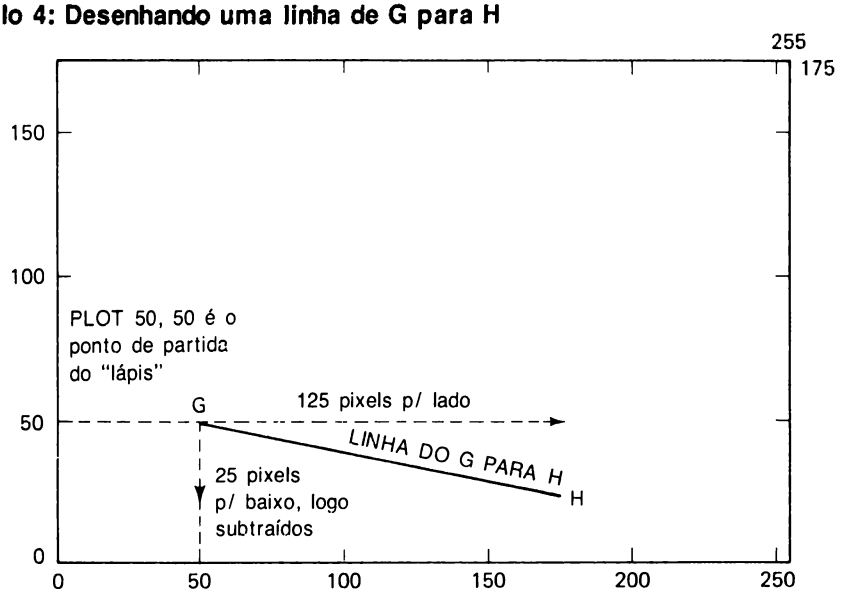
DIGITE:

```
10  PLOT  200, 150
20  DRAW  -100, -75
```

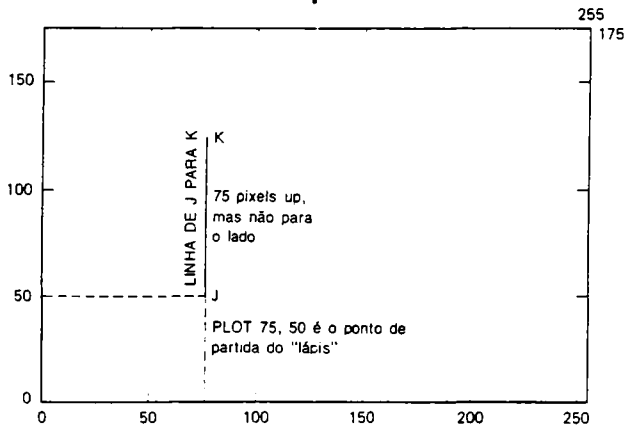
RUN

Este programa mostra que, ao contrário do comando PLOT, o comando DRAW pode usar números negativos.

Exemplo 4: Desenhando uma linha de G para H



Exemplo 5: Desenhando uma linha de J para K



DIGITE:

```
10  PLOT  75, 50
20  DRAW  0, 75
```

RUN

Atividade

2. (a) O diagrama 28.2 mostra uma tela com 6 linhas, A, B, C, D, E e F. As setas mostram a direção na qual as linhas foram desenhadas.

Escreva um programa que desenhe essas linhas nas posições corretas aproximadamente.

(b) Usando PRINT AT, marque cada linha com uma letra no local apropriado (lembre-se que com PRINT AT as linhas são numeradas diferentemente e que a linha, e não a coluna, é dada primeiro — veja a Unidade 23).

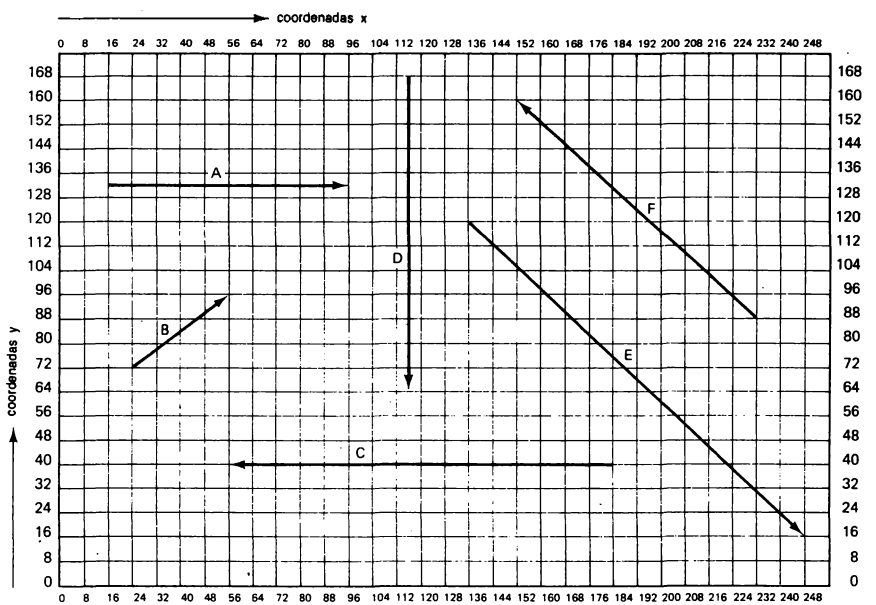
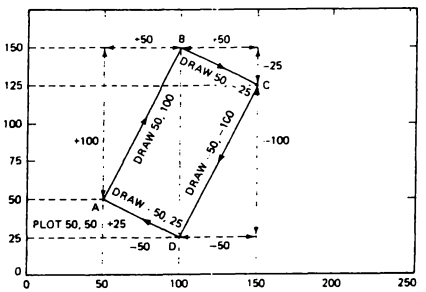


Diagrama 28.2

Se quisermos desenhar um retângulo não é necessário ativar o ponto inicial do "lápis" no começo de cada lado. Após o primeiro lado do retângulo ter sido desenhado, o "lápis" continua no fim daquela linha, de tal forma que podemos desenhar automaticamente o próximo lado a partir daquele ponto; tudo que precisamos é outro comando DRAW. O único ponto que precisa ser ativado é o inicial, chamado de **A** no diagrama a seguir:



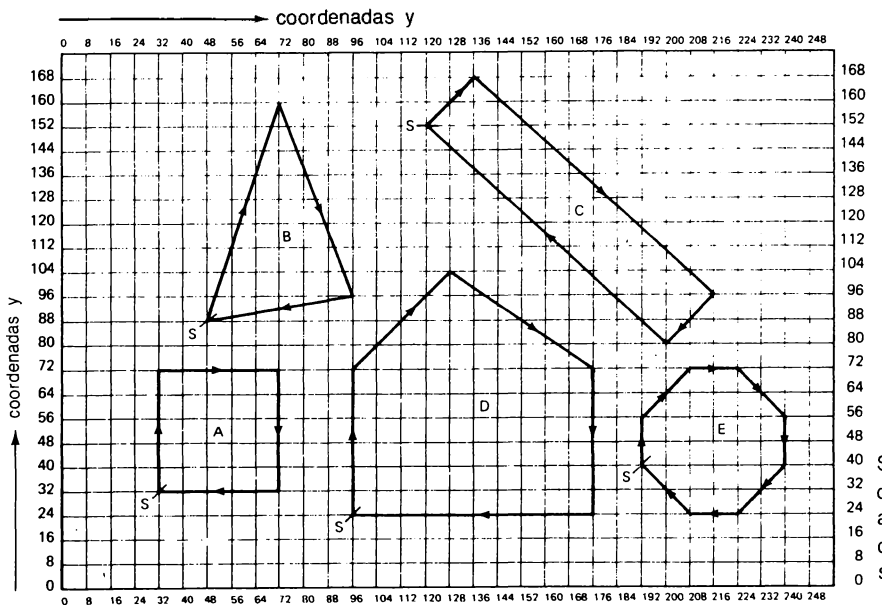
DIGITE:

```
10  REM  Retangulo ABCD
20  PLOT  50, 50          (inicio do desenho — ativa ponto de partida do lapis)
30  DRAW   50, 100        (desenha AB, lapis esta agora em B)
40  DRAW   50, — 25       (desenha BC, lapis esta agora em C)
50  DRAW   — 50, — 100    (desenha CD, lapis agora em D)
60  DRAW   — 50, 25       (desenha DA, lapis agora de volta em A)
70  STOP
```

RUN

Atividade

3. Escreva programas que desenhem as figuras dadas no diagrama 28.3. Imprima a letra correta no centro de cada figura.



S = Início direção na qual as figuras foram desenhadas nas soluções

Diagrama 28.3

Nós podemos desenhar quaisquer figuras, colocando as informações necessárias no programa enquanto estiver sendo executado. A única informação necessária é o ponto de partida do "lápis" e a altura e largura do retângulo. O próximo programa permite ao usuário desenhar 5 retângulos.

DIGITE:

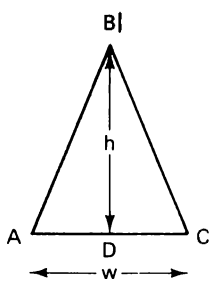
```
 10  REM  Desenhando qualquer retangulo
 20  FOR  n = 1 TO 5
 30  INPUT  "Entre a coordenada x "; x
 40  IF  x < 0 OR x > 255 THEN PRINT "Inaceitável": GOTO 30
 50  INPUT  "Entre a coordenada y "; y
 60  IF  y < 0 OR y > 175 THEN PRINT "Inaceitavel": GOTO 50
 70  INPUT  "Entre a llargura do retangulo "; w
 80  IF  x + w > 255 THEN PRINT "Muito largo": GOTO 70
 90  INPUT  "Entre a altura do retangulo "; h
100  IF  y + h > 175 THEN PRINT "Muito alto": GOTO 90
110  CLS
120  PLOT  x, y
130  DRAW  0, h
140  DRAW  w, 0
150  DRAW  0, – h
160  DRAW  – w, 0
170  NEXT  n
180  STOP
```

RUN
Forneça a informação pedida.

Linhas 40 e 60 impedem que x e y caiam fora da tela. Linhas 80 e 100 verificam se o retângulo cabe na tela. Por exemplo, se x é 200 e então o usuário entra 100 para largura 'w', é óbvio que x + w será maior que 255 e não caberá na tela. A linha 110 limpa quaisquer mensagens ou desenhos anteriores da tela. O retângulo é desenhado pelas linhas 120 e 160.

Atividades

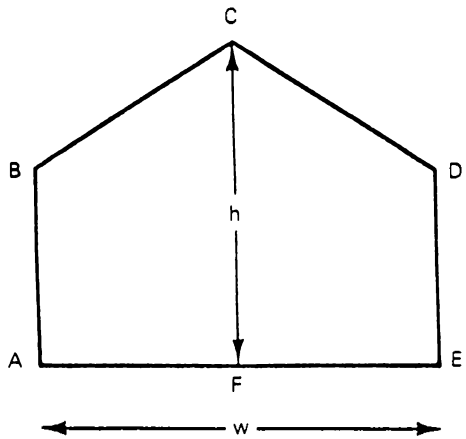
4. Escreva um programa que permita ao usuário desenhar triângulos isósceles de diversos tamanhos. (Um triângulo isósceles tem dois lados iguais.) Use o diagrama a seguir.



AB = BC
AD = DC
h = altura
w = largura

O programa deverá verificar se o triângulo cabe na tela. Comece o desenho em A.

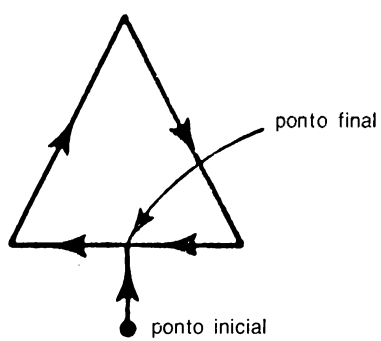
5. Adicione uma seção ao seu programa que peça ao usuário para escolher entre triângulos ou pentágonos. Se a resposta for triângulos, o computador deve ser mandado para o programa do exercício anterior; se for pentágonos, então o computador deve executar uma nova seção de seu programa que desenhe pentágonos. O diagrama abaixo será útil.



AB = DE
CB = CD
AF = FE
AB e DE são iguais à metade da altura
h = altura
w = largura

Note que a informação necessária é exatamente igual a antes (o ponto de partida, a altura e a largura), assim essa parte do programa será exatamente a mesma de antes, não precisando ser digitada duas vezes.

6. Escreva um programa que desenhe a figura mostrada no diagrama 28.4. Todas as árvores são do mesmo tamanho, logo podem ser desenhadas usando um laço FOR-NEXT. Os únicos números diferentes a serem entrados são as coordenadas necessárias para o ponto de partida de cada árvore, o que pode ser colocado num comando DATA. As árvores podem ser desenhadas com uma linha única contínua, como mostra o diagrama a seguir:



As casas podem ser desenhadas de forma similar, pois são também todas do mesmo tamanho. As ruas são um pouco mais "macetosas", mas podem ser feitas com uma linha contínua, e portanto o ponto inicial só precisa ser ativado uma vez.

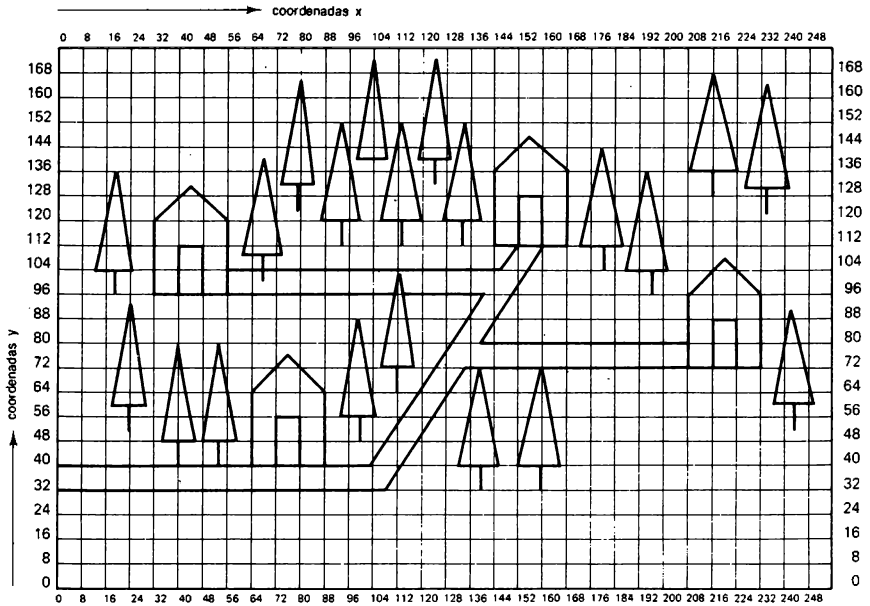
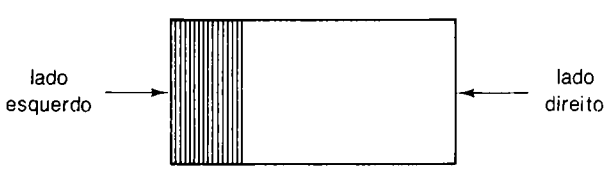


Diagrama 28.4

7. Tente escrever um programa para um dos seguintes:
   (a) uma figura sua usando apenas linhas retas
   ou
   (b) uma figura usando vários traços sobrepostos.

Sombrear uma figura

É possível sombrear ou colorir a área interna de uma figura. É mais fácil visualizar isso com um retângulo. O que nós precisamos fazer é desenhar todas as linhas verticais que existem entre o lado esquerdo e o direito do retângulo. Veja o diagrama a seguir:



Se essas linhas verticais estiverem próximas darão a impressão de colorido.

DIGITE:

```
10  REM  Sombreando um retangulo
20  INPUT  "Entre a coordenada x "; x
30  INPUT  "Entre a coordenada  y "; y
40  INPUT  "Entre a largura "; w
50  INPUT  "Entre a altura "; h
60  PLOT  x, y
70  DRAW  0, h
80  DRAW  w, 0
90  DRAW  0, – h
```

```
100  DRAW   — w, 0
110  REM  Sombreando
120  FOR  n = 1 TO w — 1
130  PLOT  x + n, y
140  DRAW  0, h — 1
150  NEXT  n
160  STOP
```

(número de linhas verticais à tração)
(inicie o "lápis" a cada execução do laço — x é incrementado de 1, e assim o "lápis" se move de 1 posição)

RUN.

Seja cuidadoso ao entrar números adequados, pois omitimos deliberadamente as linhas de crítica de tamanho, para tornar o programa menor.

NÃO APAGUE

Este programa produz um quadrado negro, pois as linhas verticais estão tão próximas que parecem se tocar. No entanto, para se convencer de que a sombra consiste de linhas verticais, tente isto:

Edite a linha 120

```
120  FOR  n = 1 TO W — 1 STEP 2
145  PAUSE  50
```

RUN

Você agora pode ver as linhas verticais sendo desenhadas. O retângulo poderia ter sido sombreado com linhas horizontais, começando por baixo.

Atividade

8. Tente a sugestão feita acima — edite o programa anterior ao invés de fazer tudo de novo.

Desenhando círculos

Para círculos usamos o comando CIRCLE, abaixo da tecla H (modo estendido e depois symbol shift). Por exemplo

```
CIRCLE  120,  88,  20
```

x e y: coordenadas para o centro do círculo

raio do círculo — número de pixels (distância do centro à borda do círculo)

DIGITE:
```
10  CIRCLE  120, 88, 20
```
RUN

Isso lhe dará um círculo cujo centro estará próximo ao centro da tela. Podemos alterar o programa para traçar círculos de diversos tamanhos em diversos lugares, testando sempre se cabem na tela.

DIGITE:

```
 10  REM  Circulos
 20  INPUT  "Entre a coordenada x para o centro do circulo "; x
 30  IF  x < 0 OR x > 255 THEN PRINT "Inaceitavel": GOTO 20
 40  INPUT  "Entre a coordenada y para o centro do circulo "; y
 50  IF  y < 0 OR y > 175 THEN PRINT "Inaceitavel": GOTO 40
 60  INPUT  "Entre o raio do circulo"; r
 70  IF  x + r > 255 OR x - r < 0 THEN PRINT "Sobra dos lados": GOTO
 60
 80  IF  y + r > 175 OR y - r < 0 THEN PRINT "Sobra acima ou abaixo";
 GOTO 60
 90  CLS
100  CIRCLE  x, y, r
110  GOTO  20
```

RUN.

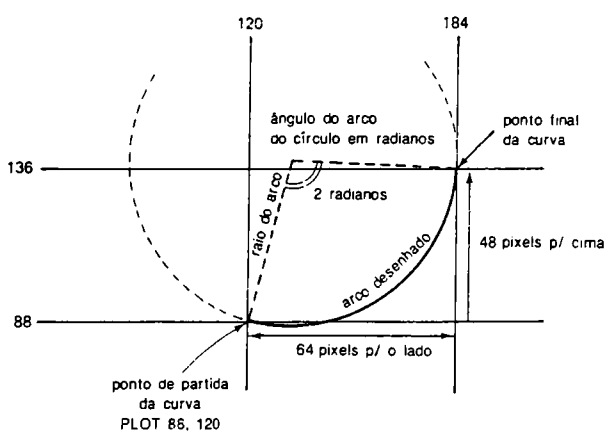
Entre vários números diferentes, algumas vezes tente alguns que você já saiba que não serão aceitos, e verifique todas as partes do programa. Use o diagrama 28.1 como guia.

Desenhando curvas ou arcos

Curvas e arcos são partes de círculos. Não são tão simples de desenhar quanto círculos. Usamos PLOT e DRAW. Por exemplo:

```
PLOT  120,  88 — dá o ponto de partida do lápis.
DRAW  64,    48,   2, ← ângulo do arco do círculo em
        ↑     ↑          radianos, NÃO em graus.
    número  número
      de      de
    pixels  pixels
    p/o lado p/cima
```

Veja o diagrama abaixo:



```
DIGITE: 10  PLOT   120, 88
        20  DRAW   64, 48, 2
```

RUN

É difícil visualizar um ângulo em radianos, mas se você pensar que 5 radianos são quase 360°, isso dará quase um círculo completo.

```
DIGITE: 10  PLOT   120, 88
        20  DRAW   64, 48, 5
```

RUN

Você não vai achar fácil julgar o tamanho e formato de um arco, portanto é melhor praticar mais com a idéia.

Atividade

9. Escreva um programa que desenhe a figura do navio Viking mostrada no diagrama 28.5

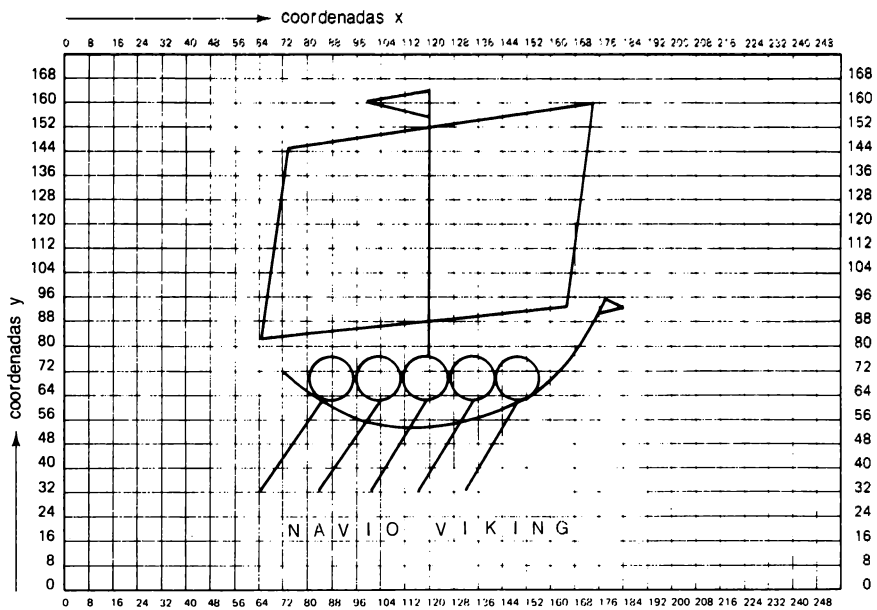


Diagrama 28.5

Apagando as linhas

Podemos remover ou apagar uma linha ou círculo usando o comando INVERSE. INVERSE (tecla M — modo estendido e depois SYMBOL SHIFT) é usado para inverter as cores da tinta e do papel na Unidade 23 em gráficos de baixa resolução. INVERSE 0 produzirá pontos de tinta normais, mas INVERSE 1 mudará os pontos de tinta para pontos de papel (branco), causando o efeito de apagar os pontos.

DIGITE:

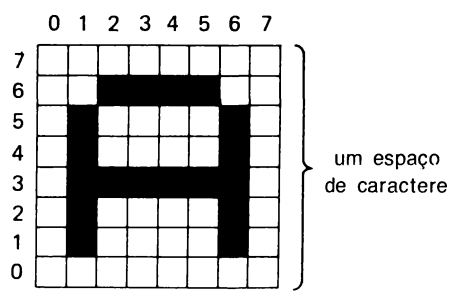
```
10  REM  Apagando
20  PLOT  24, 40
30  DRAW  INVERSE  0; 180, 112
40  PAUSE  30
50  DRAW  INVERSE  1; − 180, − 112
60  CIRCLE  INVERSE  0; 128, 88, 50
70  CIRCLE  INVERSE  1; 128, 88, 50
80  STOP
```

RUN

Unidade 29:

Caracteres Definidos Pelo Usuário

Existem 21 caracteres no TK90X que você pode alterar e fazer deles novos caracteres. As teclas que podem ser alteradas são as letras A a U. Como você sabe, cada caractere é formado por um espaço de 8 × 8 pixels. Por exemplo, a letra A pode ser composta como no diagrama abaixo:



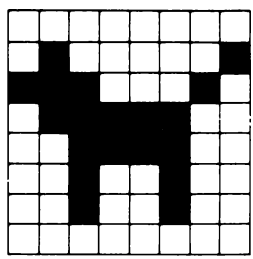
De forma a entender este caractere, o computador reconhece os espaços em branco como **pontos do papel**, atribuindo a eles valor 0, e os espaços negros com

os **pontos de tinta**, atribuindo a eles o valor 1. Assim para o computador o caractere A se parece com:

0	0	0	0	0	0	0	0
0	0	1	1	1	1	0	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	1	1	1	1	1	1	0
0	1	0	0	0	0	1	0
0	1	0	0	0	0	1	0
0	0	0	0	0	0	0	0

Um matemático saberia que números compostos por seqüências de zeros e uns são números **binários.** Nós não precisamos entender aritmética binária, mas a palavra abreviada por 'BIN' aparecerá no próximo programa.

Para fazer nosso próprio caractere precisamos, primeiro, desenhá-lo. O diagrama a seguir mostra um cachorro!



Traduzido para números binários, o caractere se parece com:

0	0	0	0	0	0	0	0
0	1	0	0	0	0	0	1
1	1	1	0	0	0	1	0
0	1	1	1	1	1	0	0
0	0	1	1	1	1	0	0
0	0	1	0	0	1	0	0
0	0	1	0	0	1	0	0
0	0	0	0	0	0	0	0

Agora precisamos pôr isso na memória do computador. Usaremos três comandos para isso:

POKE (tecla O), o que significa colocar algo na memória.
USR (acima da tecla L — modo estendido), que diz ao computador que o usuário irá construir um caractere.
BIN (acima da tecla B — modo estendido) que dá ao computador os números binários que ele entende.

A única coisa é decidir qual letra entre A e U nós iremos usar para nosso caractere. Escolhamos a letra D.

DIGITE:

```
 10  REM  Cachorro
 20  POKE  USR  "D", BIN 00000000
 30  POKE  USR  "D" + 1, BIN 01000001
 40  POKE  USR  "D" + 2, BIN 11100010
 50  POKE  USR  "D" + 3, BIN 01111100
 60  POKE  USR  "D" + 4, BIN 00111100
 70  POKE  USR  "D" + 5, BIN 00100100
 80  POKE  USR  "D" + 6, BIN 00100100
 90  POKE  USR  "D" + 7, BIN 00000000
100  PRINT  "D"      Entre 'D' no modo gráfico
```

RUN

Esse programa imprime um cão. Para imprimir vários, devemos alterar o programa.

Edite a linha 100:

```
100  FOR  n = 1 TO 20
```

Adicione:

```
100  FOR  n = 1 TO 20
110  FOR  m = 1 TO 15
120  PRINT  "D"; " ";
130  NEXT  m        Entre 'D' no modo gráfico.
140  PRINT          (O caractere do cão aparecerá)
150  NEXT  n
160  STOP
```

RUN

Pontos a ressaltar:

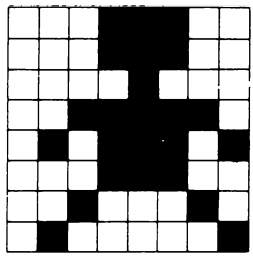
1. A letra D está entre aspas e é entrada no modo gráfico (linha 120).

2. A cada linha no caractere é dado um número. Por isso a linha de cima é "D" + 0, mas está escrita apenas como "D", a segunda linha é "D" + 1, a terceira "D" + 2, e assim por diante. D nessas linhas é entrado no modo normal (linhas 20 a 90).

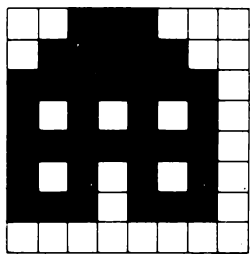
3. Nas linhas 20 a 90 há uma vírgula antes do comando BIN.

Atividades

1. Escreva programas para produzir os seguintes caracteres:
   (a) Um homem baixo e gordo



(b) Uma casa



2. Desenhe o seu próprio caractere de um carro. Faça um programa para o caractere e depois ponha o carro para passear pela tela.

Unidade 30:

Algumas Funções String Interessantes

LEN

A função LEN, acima da tecla K – modo estendido, é uma abreviação para **lenght** (em Inglês, **comprimento**), e calcula o comprimento de um string.

DIGITE:

```
10  REM  Usando LEN
20  LET  a$ = "gato"
30  PRINT  LEN  a$
40  LET  b$ = "hipopotamus"
50  PRINT  LEN  b$
60  LET  c$ = " "
70  PRINT  LEN  c$
80  STOP
RUN
```

O resultado é:
```
 4
12
 0
```

Assim, os comandos PRINT, nas linhas 30, 50 e 70 não funcionam normalmente, imprimindo o conteúdo das variáveis string, a$, b$ e c$, mas dá, ao invés disso, o comprimento, ou número de caracteres, no string. Uma variável string não contendo nada, como c$ na linha 60, retornará o valor 0.

Um programa pode ser escrito para calcular o comprimento de qualquer string de uma lista de dados.

DIGITE:

```
10  REM  Comprimento de strings
20  FOR  n = 1 TO 6
30  READ  a$
40  PRINT  a$; " tem "; LEN a$; " letras"
50  DATA  "LEN", "calcula", "o comprimento", "de", "strings"
60  NEXT  n
70  STOP

RUN
```

Atividade

1. Escreva um programa onde o usuário pode entrar quaisquer oito palavras, e onde a palavra e seu comprimento são impressos.

O próximo programa usa LEN para fixar o número de vezes que um laço FOR-NEXT será executado. Ele imprime a palavra que foi entrada e depois imprime a mesma palavra, uma letra a mais de cada vez, até ela estar completa. A unidade 50 usa a técnica de extrair partes de uma string, encontrada na Unidade 10.

DIGITE:

```
10  REM  Imprimindo uma palavra letra por letra
20  INPUT  "Entre a palavra "; a$
30  PRINT  a$
40  FOR  n = 1 TO LEN a$
50  PRINT  a$ (1 TO n)
60  PAUSE  50
70  NEXT  n
80  STOP

RUN.
```

Entre a palavra 'Spectrum'.

O resultado será:
```
Spectrum
S
Sp
Spe
Spec
Spect
Spectr
Spectru
Spectrum
```

A palavra 'Spectrum' tem oito caracteres, logo o laço FOR-NEXT 'n' será executado oito vezes. A linha 50 extrai parte de string do primeiro caractere ao n-ésimo caractere. A primeira vez que o laço é executado, 'n' é 1 e assim a$ (1 TO 1) é S, na segunda vez, 'n' é 2 e a$ (1 TO 2) é Sp, e assim por diante.

Execute o programa com várias palavras diferentes.

Podemos imprimir palavras (ou frases ou caracteres) detrás para diante, da mesma forma. Modifique o programa anterior:

```
35  LET   m = LEN a$
40  FOR   n = m TO 1 STEP -1
50  PRINT   a$ (n TO m)
```

Execute o programa diversas vezes, entrando palavras ou frases diferentes.

CHR$ e CODE

A função CHR$ (acima da tecla U — modo estendido), uma abreviação para **character** (em inglês, caractere) devolve o caractere correspondente a qualquer número dado. Existem 256 caracteres na memória do computador, cada um com um número entre 0 e 255. Há uma lista completa de caracteres no seu manual do TK90X.

DIGITE:

```
10  REM   Caracteres
20  PRINT   CHR$ 37
30  PRINT   CHR$ 101
40  PRINT   CHR$ 135
50  PRINT   CHR$ 198
60  PRINT   CHR$ 237
70  STOP

RUN
```

O resultado é: %
e

AND
GOSUB

Você pode verificar no manual se esses são os caracteres corretos para aqueles números. Note que as palavras GOSUB são tratadas como um único caractere.

DIGITE:

```
10  FOR   n = 32 TO 255
20  PRINT   CHR$ n;
30  PAUSE   40
40  NEXT   n
50  STOP

RUN
```

Esse programa produz todos os caracteres de 32 a 255. Há dois pontos a serem notados:

(1) Os caracteres de 0 a 31 não produzem nada que possa ser impresso na tela, são caracteres de controle. O caractere 32 é espaço, assim na tela você teve um salto na primeira linha.
(2) As letras maiúsculas de A a U aparecem 2 vezes, pois na segunda elas representam os caracteres definidos pelo usuário.

A função CODE (acima da tecla I, modo estendido) faz o oposto de CHR$. Ela devolverá o número do caractere dado.

DIGITE:

```
10  REM  Usando CODE
20  PRINT  CODE  "A"
30  PRINT  CODE  "£"
40  PRINT  CODE  "INKEY$"
50  PRINT  CODE  " CIRCLE"
60  PRINT  CODE  "hello"
70  STOP

RUN
```

O resultado será: 65
96
166
216
104

Novamente, você pode verificar no manual, se desejar. Note que para CODE "hello" o resultado é 104, que é o número da letra 'h', assim, se uma palavra normal é dada, o caractere considerado é apenas a letra inicial. CODE "INKEY$" e CODE "CIRCLE" produzem o número da palavra inteira, pois se trata de palavras-chaves.

VAL (não confundir com VAL$)

A função VAL (acima da tecla J—modo estendido), uma abreviação para **avaliar**, pode transformar uma string num número.

DIGITE:

```
10  REM  Usando VAL
20  LET  a$ = "1 + 2 + 3 + 4"
30  PRINT  a$
40  PRINT  VAL  a$
50  LET  b$ = "10 * 10"
60  PRINT  b$
70  PRINT  VAL  b$
80  STOP

RUN
```

As linhas 30 e 60 imprimem os conteúdos de a$ e b$ do jeito normal, mas a função VAL nas linhas 40 e 70 fazem com que o computador trate os conteúdos das variáveis string como números. Daí, VAL a$ dá o valor de 1 + 2 + 3 + 4 e VAL b$ dá o valor de 10 * 10.

Unidade 31:

Usando a Impressora

A impressora ZX está fixada ao longo do conector na borda detrás do computador. Não deve ser fixada ou separada enquanto o computador estiver ligado. As palavras-chaves LPRINT, LLIST e COPY são usadas com a impressora.

LPRINT (acima da tecla C — modo estendido)

```
DIGITE:  10  LPRINT "O resultado desse programa sera impresso em papel"
         20  LPRINT "O resultado nao aparecera na tela"
RUN          NÃO REMOVA
```

O resultado é como dito no programa. Logo, se você quiser, por alguma razão, que algo saia apenas no papel, use LPRINT ao invés de PRINT.

LLIST (acima da tecla V — modo estendido)

Adicione ao programa

```
30  LLIST
```

RUN APAGUE O PROGRAMA

Em resposta ao LLIST o computador produzirá uma listagem do programa no papel.

DIGITE:
```
10 PRINT "O resultado deste programa aparecera na tela"
20 PRINT "Uma listagem deste programa sera tambem impressa em papel"
30 LLIST
```

RUN NÃO REMOVA

Agora o resultado é impresso na tela, e a listagem do programa, no papel.

COPY (na tecla Z)

Mude as linhas 20 e 30

```
20 PRINT "e sera tambem impresso em papel"
30 COPY
```

RUN

Desta vez o resultado do programa está na tela e no papel. Quando usando COPY a impressora funciona, eventualmente desperdiçando papel; isso pode ser parado pressionando-se CAPS SHIFT e BREAK.

É possível comprar conectores especiais (chamados **interfaces**) que permitem que o TK90X seja conectado a uma gama variada de impressoras alternativas. Elas são, em geral, acompanhadas de um pequeno programa a ser carregado no computador, de forma a "traduzir" os sinais antes de enviá-los à impressora. Assim, resultados muito sofisticados podem ser obtidos se uma impressora de rolete ou uma impressora matricial, de alta qualidade, for usada. Você deve decidir sobre o tipo de impressora que precisa e depois verificar se uma interface pode ser obtida, antes de comprar qualquer coisa.

Soluções

Unidade 3

1.

```
10 PRINT "Sr. Jayme T. Filho"
20 PRINT "Av. Gen. Justo 171/606"
30 PRINT "Rio de Janeiro"
40 PRINT "CEP 20000"
```

2.

```
10 PRINT "Meu nome e Jayme Teixeira Filho"
20 PRINT "Minha idade e 24 anos"
30 PRINT "Minha data de nascimento e 4 de janeiro de 1962"
```

3.

```
10 PRINT "Somos 4 pessoas na minha familia"
20 PRINT "Eu tenho um irmao chamado Jose Heitor"
30 PRINT "Meus pais vivem no Rio de Janeiro"
```

Unidade 8

1.	2.	3.			
32	72	a)	ABC	f)	Abc
32	cães	c)	abc	i)	A2B2C
y	53	d)	Dois ABC	j)	a b c

Unidade 9

1.

```
10 LET a$ = "Joao"
20 LET s$ = "escola"
30 PRINT a$; " vai a "; s$
```

2.

```
10  LET  m$ = "Maria"
20  LET  a$ = "Ana"
30  LET  b$ = "12 anos"
40  LET  c$ = "13 anos"
50  PRINT  m$; " e "; b$; " velho"
60  PRINT  a$; " e "; c$; " velho"
```

3.

```
10  LET  a$ = "Roberto"
20  LET  b$ = "Joao"
30  LET  c$ = "Londres"
40  LET  d$ = "Liverpool"
50  PRINT  a$; " vive em "; c$
60  PRINT  b$; " vive em "; d$
```

4.

```
10  LET  n$ = "Rafael"
20  LET  d$ = "12 de setembro"
30  LET  w$ = "sabado"
40  LET  t$ = "8 da noite"
50  PRINT  "Caro "; n$
60  PRINT  "Por favor venha a festa
no "; w$; " o "; d$; " as "; t$
70  PRINT  "          abracos"
80  PRINT  "          Sandra"
```

5.

```
10  LET  a$ = "James"
20  LET  b$ = " Harriot"
30  PRINT  a$ + b$
```

6.

```
10  LET  a$ = "Sal"
20  LET  b$ = "va"
30  LET  c$ = "dor"
40  PRINT  a$ + b$ + c$
```

7.

```
10  LET  a$ = "O"
20  LET  b$ = "GAROTO"
30  LET  c$ = "GULOSO"
40  LET  d$ = "COMEU"
50  LET  e$ = "VARIOS"
60  LET  f$ = "BISCOITOS"
70  LET  g$ = "COM"
80  LET  h$ = "GELEIA"
90  LET  i$ = "E"
100 LET  j$ = "QUEIJO"
110 PRINT  a$; " "; c$; " "; d$; " "; g$
120 PRINT  a$; " "; b$; " "; c$; " "; d$;
" "; e$; " "; j$
130 PRINT  a$; " "; c$; " "; d$; " "; f$;
" "; j$
```

Unidade 10

1.

```
10  LET  b$ = "mareacao"
30  PRINT  b$; " "; c$
```

2.

```
60  PRINT  p$ (5 TO 6) + b$ (4)
70  PRINT  p$ (1 TO 2) + b$ (5)
80  PRINT  b$ (2) + p$ (3) + b$ (4)
90  PRINT  p$ (1) + b$ (2) + b$ (5)
100 PRINT  b$ (5) + b$ (4) +
p$ (1 TO 3)
```

3.

```
10  LET  a$ = "armacao"
20  LET  b$ = "rode"
30  PRINT  a$ (1 TO 2) + "b" + a$ (3
TO 4) + c$ (1 TO 1) + "b" + a$ (4 TO 8)
40  PRINT  b$ (1 TO 3) + a$ (4 TO 8)
```

Unidade 11

1.

a) 3 + 7 d) 5 * 2 + 8 g) 5 * 6 * 7
b) 3 * 7 e) 24 − 4 * 3 h) 81 − 27/2
c) 8/4 f) 30/3 + 2 i) 23/2 * 9

2.

a) 17 c) 100 e) 5
b) 200 d) 25 f) 3

3.

a) 26 b) 13 c) 8 d) 1

4.

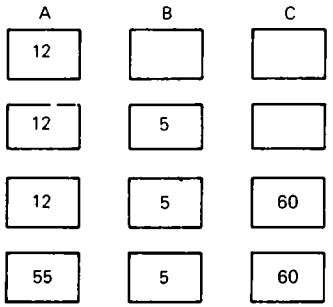
a) 14 c) 4 e) 15
b) 20 d) 24 f) 12

5.

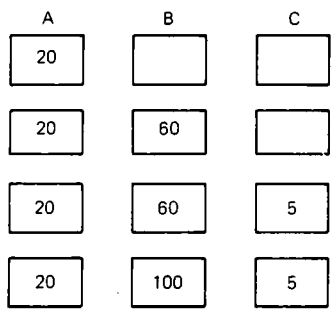
a) 17 c) 50 e) 7
b) 3 d) 5 f) 20

6.

a)



b)



7.

```
10  LET  a = 7
20  LET  b = 3
30  PRINT  a + b
40  PRINT  a − b
50  PRINT  a * b
60  PRINT  a/b
```

(quaisquer números aqui)

8.

```
10  LET  l = 35
20  LET  b = 16
30  PRINT  "A area do retangulo e ";
l * b; " cm quadrados"
40  PRINT  "O perimetro do retangulo
e "; l + l + b + b; " cm"
```

9.

```
10  LET  x$ = "Sr. Irineu"
20  LET  b = 93
30  LET  c = 9
40  PRINT  "Um saco de cimento custa ";
b; " Cz$"
50  PRINT  x$; " comprou "; c; " sacos"
60  PRINT  "O custo de "; c; " sacos e";
b * c; " Cz$"
```

10.

```
10  LET  a$ = "Susana Silva"
20  LET  b$ = "Joao Souza"
30  LET  c$ = "Maria Costa"
40  LET  a = 43
50  LET  b = 68
60  LET  c = 73
70  PRINT  a$; " tirou"; a; " pontos"
80  PRINT  b$; " tirou"; b; " pontos"
90  PRINT  c$; " tirou"; c; " pontos"
100  PRINT  "A media foi "; (a + b + c)/3
```

11.

```
10  LET  s = 65
20  LET  t = 6
30  PRINT  "A velocidade do carro e ";
s; " km/h"
40  PRINT  "O tempo gasto e "; t; " horas"
50  PRINT  "A distancia da viagem e ";
s * t; " km"
```

12.

```
10  LET  a = 21
20  LET  w = 10
30  LET  h = 160
40  LET  s = 5
50  PRINT  "Eu tenho "; a; " anos e
peso "; w; " kg. Eu tenho "; h; " cm de
altura e calco "; s; "sapatos"
```

13.

```
10  LET  a = 21
20  LET  j = 26
30  LET  k = 43
40  LET  m = 65
50  LET  f = 72
60  PRINT  "A soma das idades da familia
e "; a + j + k + m + f
70  PRINT  "A media de idade da familia
e "; (a + j + k + m + f)/5
```

idades das crianças

mae

pai

14.

```
10  LET  l = 1154
20  LET  j = 19
30  PRINT  "A quantidade de areia carre
gada a cada quatro semanas e "; l * j; " kg"
40  PRINT  "A quantidade carregada em
48 semanas e "; l * j * 48/4; " kg"
```

15.

```
10  LET  b = 0.44
20  LET  m = 2
```

```
30  LET  ba = 10.4
40  PRINT  "O preco de 3 paes e ";
3 * b; " Cz$"
50  PRINT  "O preco de 5 litros de leite
e "; 5 * m; " Cz$"
60  PRINT  "O preco de meio quilo de
bacon e "; 5 * m; " Cz$"
70  PRINT  "O custo total foi ";
3 * b + 5 * m + ba/2; " Cz$ "
```

Unidade 12

1.

```
10  PRINT  "————————————————————
————————————"
20  PRINT  "DIA"; TAB 10;
"TEMPERATURA"; TAB 23; "CHUVA"
30  PRINT  TAB 10; "GRAUS C"; TAB
23; "POLEGADAS"
40  PRINT  "————————————————————
————————————"
50  PRINT  "Segunda"; TAB 13; "10";
TAB 25; "0.2"
60  PRINT  "Terca"; TAB 13; "11"; TAB
25; "0.4"
70  PRINT  "Quarta"; TAB 13; "13"; TAB
25; "0.6"
80  PRINT  "Quinta"; TAB 13; "11"; TAB
25; "1.0"
90  PRINT  "Sexta"; TAB 13; "6"; TAB
25; "0"
```

Unidade 14

1.

```
10  REM  Numeros
20  INPUT  "Entre um numero "; n
30  PRINT  n
40  INPUT  "Entre um segundo numero ";
n2
50  PRINT  n2
60  INPUT  "Entre um terceiro numero ";
n3
70  PRINT  n3
80  INPUT  "Entre um quarto numero "; n4
90  PRINT  n4
100  LET  s=n+n2+n3+n4
110  PRINT  Soma dos numeros="; s
120  LET  a=s/4
130  PRINT  "Media dos numeros="; a
140  STOP
```

2.

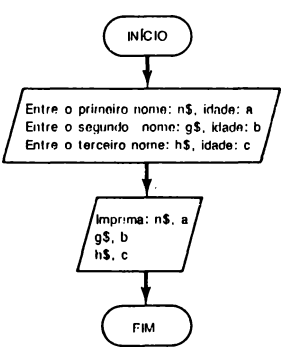
```
10  REM  Carta para entrevista
20  INPUT  "Entre o nome do entrevista
do "; n$
30  INPUT  "Entre a data da entrevista ";
d$
40  INPUT  "Entre a hora da entrevista ";
t$
50  INPUT  "Entre o local da entrevista ";
p$
60  INPUT  "Entre o nome do entrevista
dor "; m$
70  INPUT  "Entre o nome da pessoa que
enviou a carta "; s$
80  INPUT  "Entre a data da carta "; l$
90  PRINT  TAB 18; "Petrobras S.A."
100  PRINT  TAB 20; "Servico de material"
110  PRINT  TAB 22; "Rio de Janeiro"
120  PRINT  TAB 18; l$
130  PRINT  "Rio de Janeiro "; n$
140  PRINT  TAB 15; "Gratos pela resposta
a nosso anuncio. Por favor, compareca para
uma entrevista dia "; d$, " as "; t$;
150  PRINT  " A entrevista sera no "; p$;
" conduzida por "; m$
160  PRINT  TAB 15; "Atenciosamente"
170  PRINT  TAB 18; s$
180  STOP
```

3.

```
10  REM  Poupando o dinheiro
20  INPUT  "Entre a quantidade de dinhei
ro que voce deseja poupar "; s
30  INPUT  "Entre o quanto pode poupar
a cada semana "; w
40  LET  n=s/w
50  PRINT  "O numero de semanas neces
sarias para poupar Cz$ "; s; " e "; n
60  PRINT  "O numero de meses necessa
rios para poupar Cz$ "; s; " e "; n/4
70  STOP
```

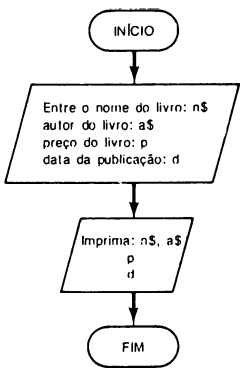
Unidade 15

1. Fluxograma



```
10 REM  Nomes e idades
20 INPUT "Nome da primeira garota";
n$
30 INPUT "Idade da primeira garota"; a
40 INPUT "Nome da segunda garota";
g$
50 INPUT "Idade da segunda garota"; b
60 INPUT "Nome da terceira garota"; h$
70 INPUT "Idade da terceira garota"; c
80 PRINT "NOME", "IDADE"
90 PRINT n$, a
100 PRINT g$, b
110 PRINT h$, c
120 STOP
```

2. Fluxograma

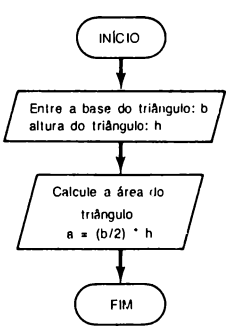



```
10 REM  Livro e Autor
20 INPUT  "Nome do livro? "; n$
30 INPUT  "Autor do livro? "; a$
40 INPUT  "Preco do livro? "; p
50 INPUT  "Data da publicacao? "; d
60 PRINT  n$; " por "; a$
70 PRINT  "Preco do livro Cz$ "; p
80 PRINT  "Data da publicacao "; d
90 STOP
```

3. a) Variáveis:

b = Base do triangulo
h = Altura do triangulo
a = Área do triangulo

Fluxograma

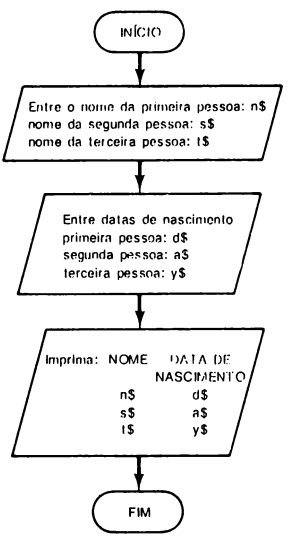



```
10 REM  Area de triangulos
20 INPUT  "Entre a base do triangulo ";
b
30 INPUT  "Entre a altura do triangulo ";
h
40 LET  a=b/2*h
50 PRINT  "A area do triangulo e "; a;
" cm2"
60 STOP
```

3. b) Variáveis:

n$ = nome da primeira pessoa
s$ = nome da segunda pessoa
t$ = nome da terceira pessoa
d$ = primeira data de nascimento
a$ = segunda data de nascimento
y$ = terceira data de nascimento

Fluxograma



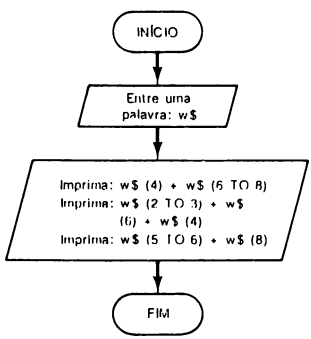

```
10 REM  Nomes e datas de nascimento
20 INPUT  "Nome da primeira pessoa ";
n$
30 INPUT  "Nome da segunda pessoa ";
s$
40 INPUT  "Nome da terceira pessoa ";
t$
50 INPUT  "Data de nascimento da 1ª
pessoa "; d$
60 INPUT  "Data de nascimento da 2ª
pessoa "; a$
70 INPUT  "Data de nascimento da 3ª
pessoa "; y$
80 PRINT  "NOME", "DATA DE NAS
CIMENTO"
```

```
 90  PRINT  n$, d$
100  PRINT  s$, a$
110  PRINT  t$, y$
120  STOP
```

3. c) Variável:

w$ = palavra com pelo menos 7 letras
(elefante)

Fluxograma

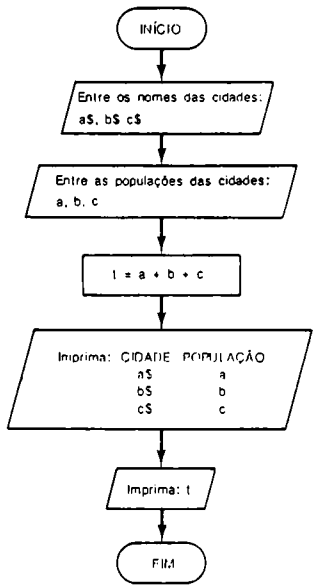



```
 10  REM  Palavras
 20  INPUT  "Entre uma palavra com pelo
menos sete letras "; w$
 30  PRINT  w$ (4) +w$ (6 TO 8)
 40  PRINT  w$ (2 TO 3) +w$ (6)
+w$ (4)
 50  PRINT  w$ (5 TO 6) +w$ (8)
 60  STOP
```

3. d) Variáveis:

a$ = nome da primeira cidade
b$ = nome da segunda cidade
c$ = nome da terceira cidade
a = populacao da primeira cidade
b = populacao da segunda cidade
c = populacao da terceira cidade
t = populacao total

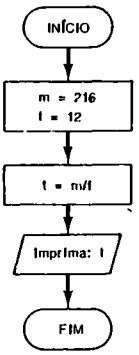
Fluxograma




```
 10  REM  Cidades e populacoes
 20  INPUT  "Nome da 1ª cidade " a$
 30  INPUT  "Nome da 2ª cidade " b$
 40  INPUT  "Nome da 3ª cidade " c$
 50  INPUT  "Populacao da 1ª cidade "; a
 60  INPUT  "Populacao da 2ª cidade "; b
 70  INPUT  "Populacao da 3ª cidade "; c
 80  LET  t=a+b+c
 90  PRINT  "CIDADE", "POPULA
CAO"
100  PRINT  a$, a
110  PRINT  b$, b
120  PRINT  c$, c
130  PRINT
140  PRINT  "O total de populacao "; t
150  STOP
```

3. e) Variáveis:

m = numero de chocolates
f = numero de amigos
t = total de chocolates que cada amigo recebera



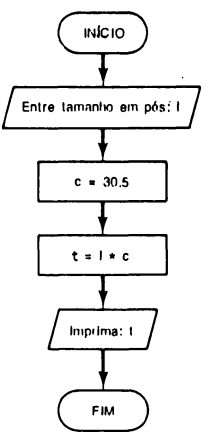

```
10  REM  Dividindo chocolates
20  LET  m=216
30  LET  f=12
40  LET  t=m/f
50  PRINT  "O numero de chocolates que
cada amigo recebera e "; t
60  STOP
```

3. f) Variáveis:

l = tamanho em pes
c = numero de centimetros em um pe
t = comprimento em centimetros

```
10  REM  Pes em centimetros
20  INPUT  "Entre tamanho em pes "; l
30  LET  c=30.5
40  LET  t=l*c
50  PRINT  l; " pes sao "; t; " cm"
60  STOP
```

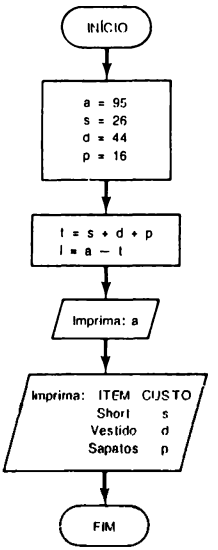
Fluxograma



3. g) Variáveis:

a = quantidade a gastar
s = preco do short
d = preco do vestido
p = preco dos tênis
t = total gasto
l = quanto sobrou

Fluxograma

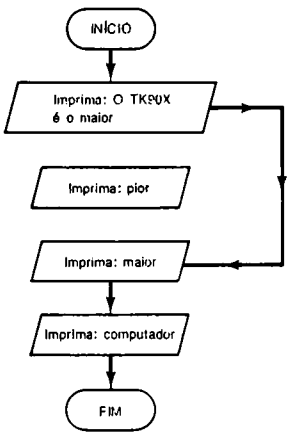



```
10  REM  Compras de uma garota
20  LET  a=95
30  LET  s=26
40  LET  d=44
50  LET  p=16
60  LET  t=s+d+p
70  LET  l=a-t
80  PRINT  "Quantidade a gastar
Cz$ "; a
90  PRINT
100 PRINT  "ITEM", "CUSTO"
110 PRINT  "short"; s
120 PRINT  "vestido"; d
130 PRINT  "sapatos"; p
140 PRINT
150 PRINT
160 PRINT  "Custo total = Cz$ "; t
170 PRINT
180 PRINT  "Sobraram Cz$ "; l
190 STOP
```

Unidade 16

1.

Fluxograma



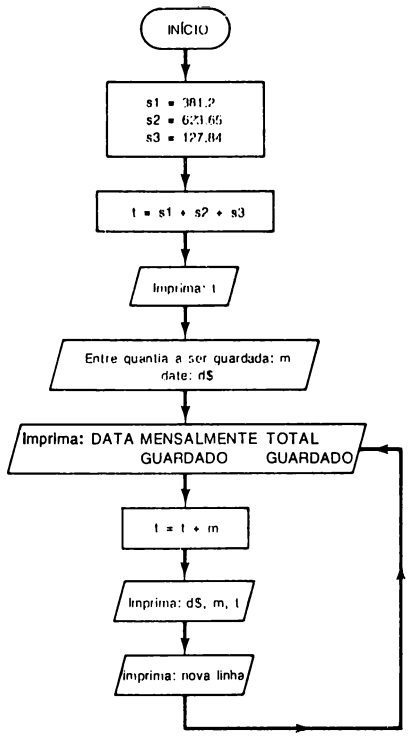
2.

```
10  REM  Somando numeros
20  INPUT  "Entre um numero "; n
30  INPUT  "Entre outro numero "; n2
40  LET  t=n+n2
50  PRINT  n;"+";n2;"=";t
60  GOTO 20
```

3.

```
10  REM  Contas do banco
20  LET  s1=381.20
30  LET  s2=623.65
40  LET  s3=127.84
50  LET  t=s1+s2+s3
60  PRINT  "Total economizado transferi
rido para uma conta e Cz$ "; t
70  INPUT  "Entre quantia a ser economi
zada este mes "; m
80  INPUT  "Entre a data "; d$
90  PRINT  "DATA"; TAB  12;
"MENSALMENTE"; TAB 23; "TOTAL"
100  PRINT  TAB  12;  "GUARDADO";
TAB 23; "GUARDADO"
110  LET  t=t+m
120  PRINT  d$; TAB 12; m; TAB 23; t
130  PRINT
140  GOTO 70
```

Fluxograma



Unidade 17

Questão.

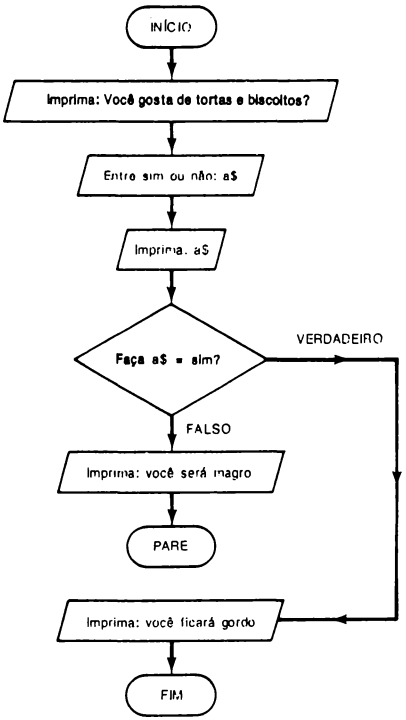
Por que foi possível usar o comando PRINT no meio das linha 40 e 50?
Por que a letra "P" não aparece?

Resposta.

Após a palavra THEN o cursor automaticamente passa de L para K, e assim o computador passa ao modo palavra-chave, o que significa que outra palavra-chave, tal como PRINT, pode ser entrada.

1.

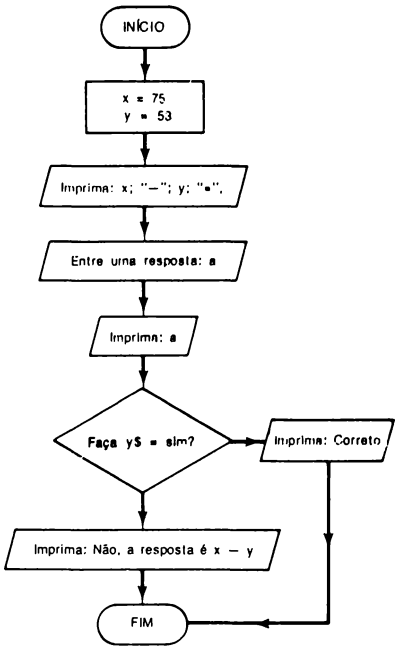
Fluxograma



2.

```
10  REM   Adivinhando a minha
idade
20  PRINT  "Qual a minha
idade?"
30  INPUT  "Entre um
numero "; a
40  PRINT  a
50  IF  a=21 THEN GOTO 80
60  PRINT  "Errado"
70  STOP
80  PRINT  "Correto"
90  STOP
```

3.



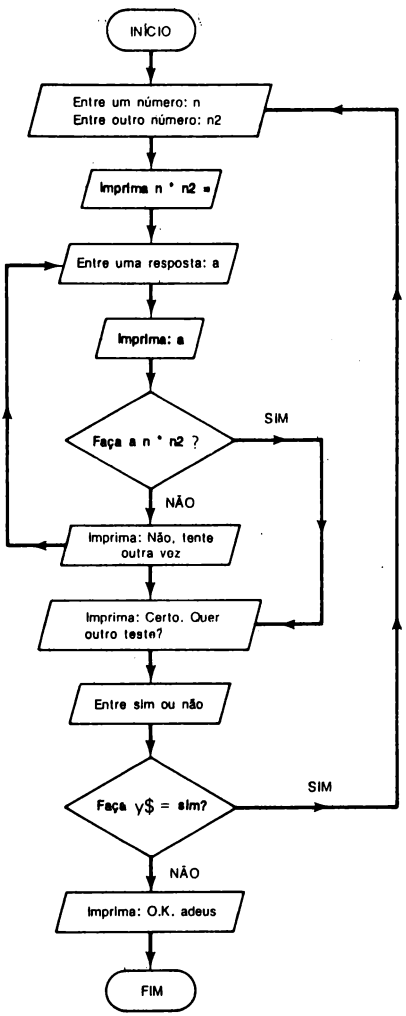

```
10  REM   Subtraindo numeros
20  LET  x=75
30  LET  y=63
40  PRINT  x;"—";y;"=";
50  INPUT  "Entre uma resposta "; a
60  PRINT  a
70  IF  a=x—y THEN PRINT "Correto":
GOTO 90
80  PRINT  "Nao, a resposta e "; x—y
90  STOP
```

4.

```
10  REM   Cor dos meus olhos
20  LET  b$="azul"
30  INPUT  "Adivinhe a cor dos meus
olhos "; e$
40  IF  b$=e$ THEN PRINT "Sim, voce
acertou": GOTO 60
50  PRINT  "Nao, voce errou"
60  STOP
```

5.

```
10 REM  Nomes de cidades
20 PRINT  "CIDADES NA GRA-BRE
TANHA"
30 INPUT  "Entre o nome de uma cidade
na Gra-Bretanha "; t$
40 IF  t$="Timbuktu" THEN GO TO 70
50 PRINT  t$
60 GOTO 30
70 STOP
```



6.

```
10 REM  Testando tabuadas de multiplica
cao
20 INPUT  "Entre um numero entre 2 e
12 "; n
30 INPUT  "Entre um segundo numero
entre 2 e 12 "; n2
40 PRINT  n;"*";n2;"=";
50 INPUT  "Entre uma resposta "; a
60 PRINT  a
70 IF  a=n*n2 THEN GOTO 100
80 PRINT  "Nao tente outra vez"
90 GOTO 50
100 PRINT  "Certo, quer um outro teste?"
110 INPUT  "Entre sim ou não "; y$
120 IF  y$="sim" THEN GOTO 20
130 PRINT  O.K. "adeus"
140 STOP
```

Unidade 18

1.

a)	nao	f)	sim	k)	sim	p)	sim
b)	sim	g)	nao	l)	nao	q)	nao
c)	sim	h)	sim	m)	sim	r)	sim
d)	sim	i)	nao	n)	sim	s)	nao
e)	sim	j)	sim	o)	nao	t)	sim

2.

a) $16 < 23$
b) $14 > 10$
c) $10 <= 10$ ou $10 >= 10$
d) $5 + 6 > 11 - 3$
e) $3 * 7 >= 13 + 8$
   ou
   $3 * 7 <= 13 + 8$
f) $x < y$
g) $x / 2 < y + 4$
h) $x * x > y / 2$
i) $(a+b) * 3 > a + b * 3$
j) $a + b - c > a + c - 10$

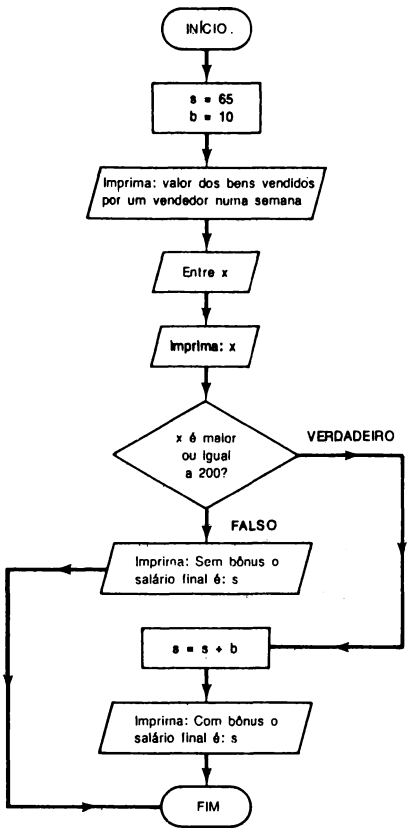
3.

```
10 REM  Qual e o numero maior
20 INPUT  "Entre um numero "; x
30 INPUT  "Entre um segundo numero ";
y
40 IF  x > y THEN GOTO 70
50 PRINT  y; " e maior que "; x
60 STOP
70 PRINT  x; " e maior que "; y
80 STOP
```

4.

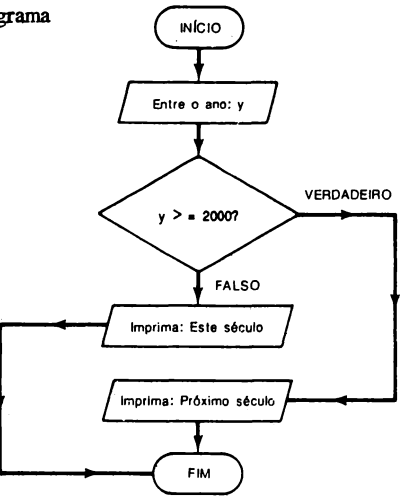
```
10 REM  se x e maior ou igual a 10
20 INPUT  "Entre um numero "; x
30 IF  x > =10 THEN GOTO 70
40 LET  y=x+10
50 PRINT  y
60 STOP
70 LET  w=x-10
80 PRINT  w
90 STOP
```

5.



6. Variável:
y = ano a ser entrado

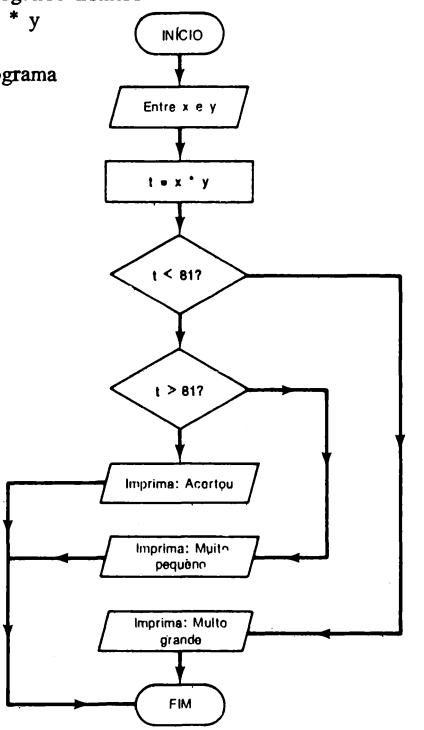
Fluxograma




```
10 REM  Qual e o seculo
20 INPUT  "Entre o numero do ano "; y
30 IF  y>=2000 THEN GOTO 60
40 PRINT  "Este seculo"
50 GOTO 70
60 PRINT  "Proximo seculo"
70 STOP
```

7. Variáveis:
x = um número
y = segundo número
t = x * y

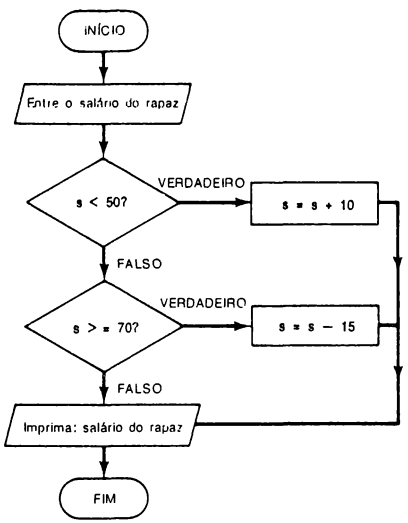
Fluxograma



```
10 REM  Maior ou menor que 81
20 INPUT  "Entre um numero "; x
30 INPUT  "Entre um segundo numero ";
y
40 LET  t=x*y
50 IF  t>81 THEN GOTO 110
60 IF  t<81 THEN GOTO 90
70 PRINT  "Acertou"
80 GOTO 120
90 PRINT  "Muito pequeno"
100 GOTO 120
110 PRINT  "Muito grande"
120 STOP
```

8. Variável:
s = salário do rapaz

Fluxograma



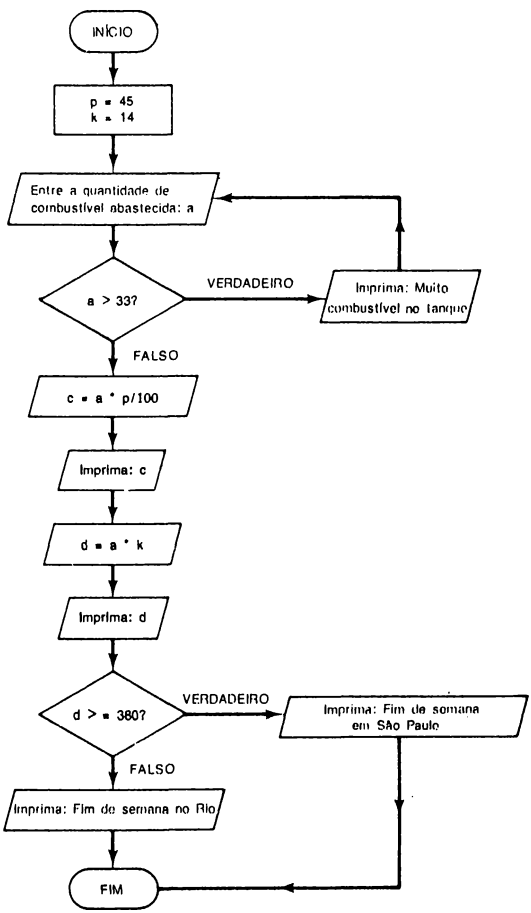

```
10 REM  Salario do rapaz
20 INPUT  "Entre o salario do rapaz ";
s
30 IF s<50 THEN LET s=s+10: GOTO
50
40 IF  s>=70 THEN LET s=s−15
50 PRINT  "Salario  semanal  do  rapaz
Cz$ "; s
60 STOP
```

9. Variáveis:

p = preco da gasolina
k = distancia percorrida com 1 litro de gasolina
d = distancia percorrida·
a = quantidade de combustivel
c = custo do combustivel

```
10 REM  Distancia percorrida
20 LET  p=45
30 LET  k=14
40 INPUT  "Entre a quantidade de
combustivel abastecida "; a
50 IF  a>33 THEN PRINT "Muito
combustivel no tanque"; GOTO 40
60 LET  c=a*p
70 PRINT  "O  custo  do  combustivel  e
Cz$ "; c/100
80 LET  d=a*k
90 PRINT  "A distancia que pode ser
percorrida com "; a; " litros e "; d; " km"
100 IF  d>=380 THEN PRINT "O homem
passa o fim de semana em Sao Paulo";
GOTO 120
110 PRINT  "O homem passa o fim de
semana no Rio"
120 STOP
```

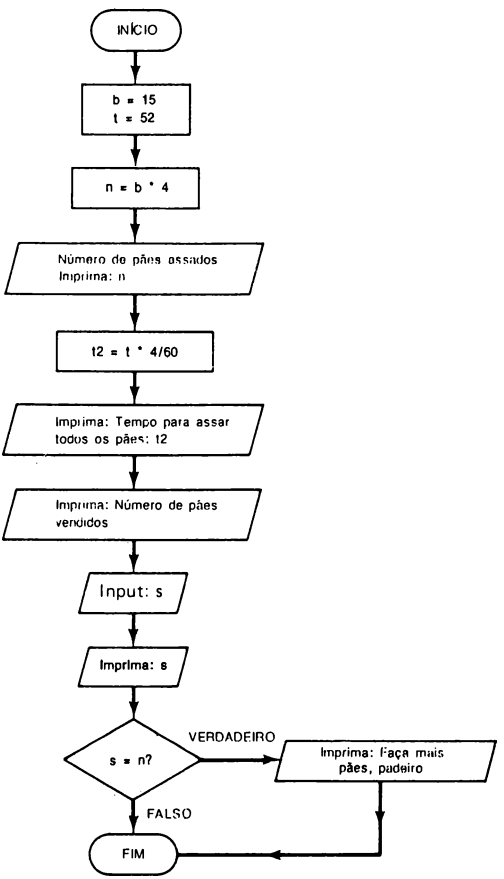
Fluxograma



10. Variáveis:

b = número de paes feitos pelo padeiro
t = tempo gasto para assar 15 pães
n = numero de paes feitos num dia
t2 = tempo gasto para assar todos os paes
s = numero de paes vendidos

Fluxograma

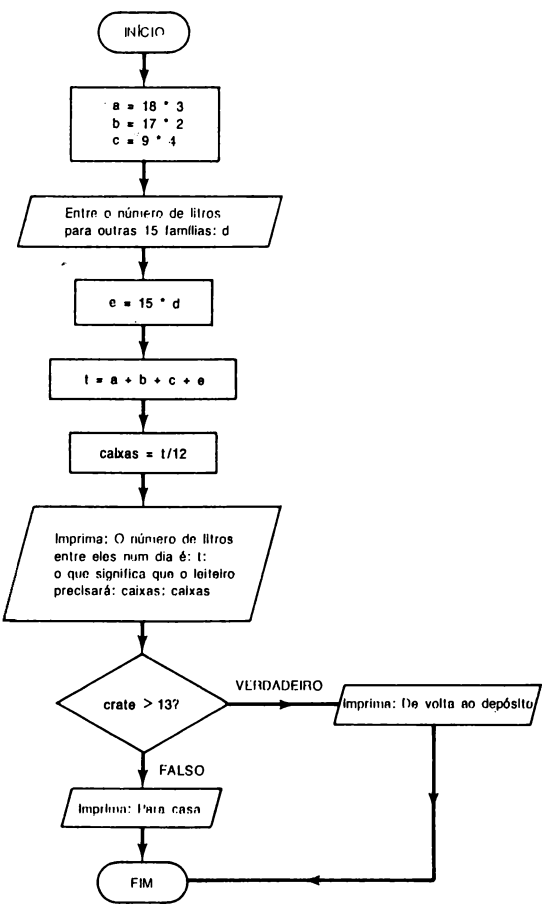



```
10  REM  Assando pao
20  LET  b=15
30  LET  t=52
40  LET  n=b*4
50  PRINT  "O numero de paes assados e ";
n
60  LET  t2=t*4/60
70  PRINT  "O tempo gasto para assar to
dos os paes e "; t2; " horas"
80  PRINT  "O numero de paes vendidos
e ";
90  INPUT  "Entre o numero "; s
100  PRINT  s
110  IF  s = n THEN PRINT "Asse mais
pao"
120  STOP
```

11. Variáveis:

a = numero de litros para 18 familias que levam 3 litros por dia
b = numero de litros para 17 familias que levam 2 litros por dia
c = numero de litros para 9 familias que levam 4 litros por dia
d = numero de litros para as outras 15 familias
e = numero de litros para as outras 15 familias levando 'd' litros por dia
caixas = numero de caixas necessarias

Fluxograma



```
10 REM Entrega do leiteiro
20 LET a=18*3
30 LET b=17*2
40 LET c=9*4
50 INPUT "Entre o numero de litros para
as outras 15 famílias "; d
60 LET e=15*d
70 LET t=a+b+c+e
80 LET caixas=t/12
90 PRINT "O numero de litros entregue
e "; t; " o que significa que o leiteiro
precisa "; caixas; " caixas"
100 IF caixa>13 THEN PRINT "de volta
ao deposito": GOTO 120
110 PRINT "Para casa"
120 STOP
```

12.

a) 30 b) 40 c) 140 d) 30
50 400 100

13.

```
10 REM Numero comparado com 100
20 INPUT "Entre um numero "; n
30 IF n>=100 AND n<=200 THEN
PRINT "Numero entre 100 e 200": GOTO
50
40 PRINT "Numero fora do intervalo
100 a 200"
50 STOP
```

14.

a) 80 b) 180 c) 160

15.

a) Adolescente
b) Adulto
c) Adolescente
d) Criança
e) Adolescente

16.

a) Sem caminhada hoje
b) Sem caminhada hoje
c) Sem caminhada hoje
d) Vamos sair para caminhar

Unidade 19

1.

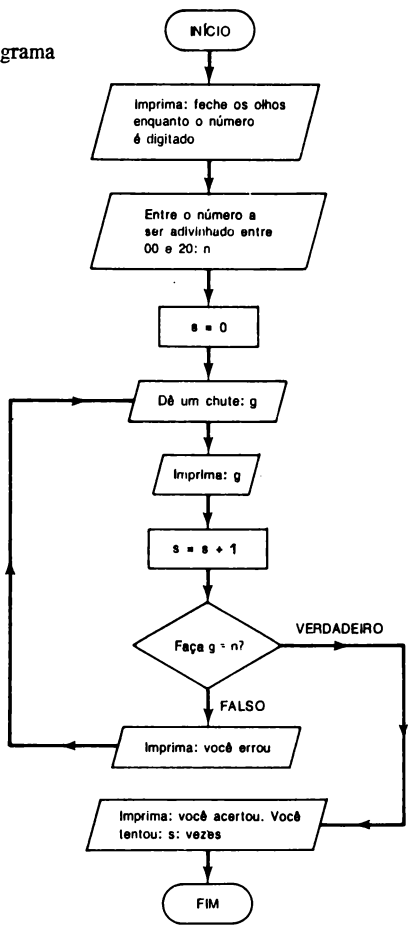
```
10 REM Flores
20 PRINT "FLORES"
30 LET c=0
40 INPUT "Entre o nome de uma flor ";
f$
50 PRINT f$
60 LET c=c+1
70 IF c=6 THEN GOTO 90
80 GOTO 40
90 STOP
```

2.

```
10 REM Flores
20 PRINT "FLORES"
30 LET c=0
35 INPUT "Entre o numero de flores a
serem listadas "; n
40 INPUT "Entre o nome de uma flor ";
f$
50 PRINT f$
60 LET c=c+1
70 IF c=n THEN GOTO 90
80 GOTO 40
90 STOP
```

3.

Fluxograma



```
10 REM Adivinhando um numero
20 PRINT "FECHE OS OLHOS ENQ
UANTO O NÚMERO É DIGITADO"
30 INPUT "Entre um numero entre 0 e
20 "; n
40 LET s=0
50 INPUT "Adivinhe um numero entre 0
e 20 "; g
60 LET s=s+1
70 PRINT g
80 IF g=n THEN GOTO 110
90 PRINT "Voce errou"
100 GOTO 50
110 PRINT "Voce acertou. Voce tentou ";
s; "vezes"
120 STOP
```

Unidade 20

1.

a)
```
1
2
3
4
5
6
7
8
9
10
11
12
```

b)
```
4
5
6
7
8
9
10
11
12
13
14
15
16
```

c)
```
Adeus
Adeus
Adeus
Adeus
```

d)
```
1
Isso e um laco
2
Isso e um laco
3
Isso e um laco
```

2.

a)
```
2
4
6
8
10
12
```

b)
```
15
19
23
```

c)
```
0
20
40
60
80
100
```

d)
```
50
25
0
- 25
```

e)
```
- 2
- 5
- 8
- 11
```

f)
```
- 10
Crescendo
- 5
Crescendo
0
Crescendo
5
Crescendo
10
Crescendo
15
Crescendo
20
Crescendo
25
Crescendo
```

3.

a)
```
10 FOR x = 1 TO 10
20 PRINT x
30 NEXT x
```

b)
```
10 FOR x = 10 TO 1 STEP - 1
20 PRINT x
30 NEXT x
```

c)
```
10 FOR x = 1 TO 6 STEP 2
20 PRINT x
30 NEXT x
```

d)
```
10 FOR x = n TO 10
20 PRINT x
30 NEXT x
```

e)
```
10 FOR x = 1 TO n
20 PRINT x
30 NEXT x
```

f)
```
10 FOR x = 1 TO 10
20 PRINT x
30 NEXT x
```

4.

a)
```
2
3
4
```

```
6
7
8
```

b)
```
10
12
14
16
```

```
12
14
16
18
```

c)
```
12
9
6
3
```

```
10
10
10
10
```

d)
```
100
125
150
175
200
```

```
90
115
140
165
190
```

```
110
135
160
185
210
```

5.

```
10 FOR x=13 TO 23
20 PRINT x
30 NEXT x
```

6.

```
10 FOR x=3 TO 37 STEP 2
20 PRINT x
30 NEXT x
```

7.

```
10 FOR x=5 TO 35 STEP 5
20 PRINT x
30 NEXT x
```

8.

```
10 FOR x=10 TO - 6 STEP - 2
20 PRINT x
30 NEXT x
```

9.

```
10 REM TABUADA DO 7
20 FOR x=1 TO 12
30 PRINT x;"*";7;"=";x*7
40 NEXT x
```

10.

```
10 PRINT "Eu estou na linha de cima"
20 FOR x=1 TO 5
30 PRINT
40 NEXT x
50 PRINT "Estou cinco linhas abaixo"
60 FOR x=1 TO 5
70 PRINT
80 NEXT x
90 PRINT "Estou cinco linhas mais abai
xo"
100 STOP
```

11.

```
10 FOR x=1 TO 8
20 PRINT "Spectrum"
30 NEXT x
```

12.

```
10 REM Anos bissextos
20 FOR x=1904 TO 1984 STEP 4
30 PRINT x
40 NEXT x
50 STOP
```

13.

```
10 FOR x=10 TO 20
20 PRINT x, x+3
30 NEXT x
```

14.

```
10 REM Nomes e idades
20 PRINT "NOME", "IDADE"
30 FOR x=1 TO 6
40 INPUT "Entre um nome "; n$
50 INPUT "Entre a idade da pessoa "; a
60 PRINT n$, a
70 NEXT x
80 STOP
```

15.

```
10 REM Nomes e precos de carros
20 PRINT "NOME DE CARRO",
"PRECO EM Cz$"
30 LET t=0
40 FOR x=1 TO 4
50 INPUT "Entre o nome do carro "; c$
60 INPUT "Entre o preco do carro "; p
70 PRINT c$, p
80 LET t=t+p
90 NEXT x
100 PRINT "O preco total dos carros e
Cz$ "; t
110 PRINT "O preco medio dos carros e
Cz$ "; t/4
120 STOP
```

16.

```
10 REM Crescimento demografico
20 PRINT "PAIS"; TAB 9; "TAXA DE
NASCIMENTO"; TAB 21; "TAXA DE
MORTALIDADE"
30 FOR x=1 TO 7
40 INPUT "Entre o nome do pais "; c$
50 INPUT "Entre a taxa de nascimento
"; b
60 INPUT "Entre a taxa de mortalidade
"; d
70 PRINT c$;TAB13;b;TAB 25;d
80 LET n=b-d
90 PRINT "O crescimento demografico
por 1000 pessoas e: "; n
100 PRINT
110 NEXT x
120 STOP
```

Unidade 21

1.

```
10 READ a, b, c
20 PRINT a, b, c
30 DATA 100, 200, 300
```

2.

```
10 READ x$, y$, z$
20 PRINT x$, y$, z$
30 DATA "gato", "cao", "peixe"
```

3.

```
10 READ c, a$, b$
20 PRINT c;a$;b$
30 DATA 3, "ratos", "cegos"
```

4.

```
10 READ x$, y, b$
20 PRINT y;b$;x$
30 DATA "garrafas", 10, "verdes"
```

5.

```
10 FOR n=1 TO 4
20 READ x
30 PRINT x
40 DATA 1, 2, 3, 4
50 NEXT n
```

6.

```
10 FOR n=1 TO 3
20 READ x$
30 PRINT x$
40 DATA "homem", "mulher", "crianca"
50 NEXT n
```

7.

```
10 FOR n=1 TO 3
20 READ b, a$
30 PRINT b, a$
40 DATA 3, "barcos", 2, "carros",
1, "trem"
50 NEXT n
```

8.

a$ = Antonio b$ = Jose c$ = Henrique

9.

a = 3 b$ = Noruega

10.

x = 10 y = 3 z = 17

11.

```
10 REM Dias da semana
20 FOR n=1 TO 7
30 READ d$
40 PRINT d$
50 DATA "Segunda", "Terca", "Quarta",
"Quinta", "Sexta", "Sabado", "Domingo"
60 NEXT n
70 STOP
```

12.

```
10 REM Lista de numeros
20 FOR x = 1 TO 10
30 READ n
40 PRINT n
50 DATA 35, 42, 57, 39, 48, 64, 44, 50,
61, 54
60 NEXT n
70 STOP
```

13.

```
10 REM Trocadilho
20 READ a$, b$, c$, d$
30 READ e$, f$, g$, h$, i$
40 DATA "O", "gato", "do", "rei", "de",
"Roma", "fugiu", "da", "Babilonia"
50 PRINT a$, " ", b$, " ", c$, " ",
d$, " ", e$, " ", f$, " ", g$, " ", i$
60 PRINT a$, " ", d$, " ", c$, " ",
b$, " ", h$, " ", i$, " ", g$, " ", e$,
" ", f$
70 PRINT a$; " "; f$; " "; c$; " "; d$;
" "; e$; " "; g$
80 STOP
```

14.

```
10 REM Comida e precos
20 READ l$, l, s$, s, j$, j
30 READ b$, b, e$, e, r$, r
40 PRINT "COMIDA", "PRECO"
50 PRINT l$, l
60 PRINT s$, s
70 PRINT j$, j
80 PRINT b$, b
90 PRINT e$, e
100 PRINT r$, r
110 DATA "pao", 36, "sopa", 19, "ge
leia", 58, "feijao", 17, "ovos", 42, "arroz", 88
120 LET t=1+s+j+b+e+r
130 PRINT "O custo total da comida e
Cz$ "; t/100
140 STOP
```

Unidade 22

1.

```
10 REM Asteriscos
20 FOR n=1 TO 10
30 FOR x=1 TO 10
40 PRINT "*";
50 NEXT x
60 PRINT
70 NEXT n
80 STOP
```

2.

```
10 REM  Poupancas
20 PRINT "Nome", "Quantia poupada"
30 FOR  n=1 TO 5
40 INPUT "Entre o nome da pessoa "; n$
50 INPUT "Entre a quantia paga "; s
60 PRINT n$, s
70 FOR  r=6 TO 10 STEP 2
80 LET  i=s*r/100
90 PRINT "A taxa de juros de "; r; "%
e Cz$ "; i
100 PRINT "O total poupado e Cz$ "; s+1
110 NEXT r
120 PRINT
130 NEXT n
140 STOP
```

3.

```
10 REM  Retangulos de asteriscos
20 FOR  n=1 TO 5
30 INPUT "Entre comprimento de retan
gulo "; l
40 INPUT "Entre a largura "; w
50 FOR  x=1 TO 1
60 FOR  y=1 TO w
70 PRINT "*";
80 NEXT y
90 PRINT
100 NEXT x
110 PRINT
120 PRINT
130 NEXT n
140 STOP
```

Unidade 23

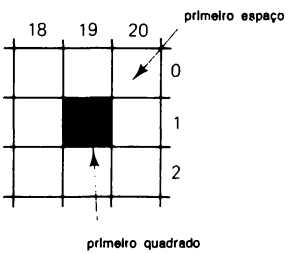
2. a

```
10 REM  Movendo um carro
20 BORDER 6
30 PAPER  1
40 LET  a=10
50 FOR  b=0 TO 24
60 INK  7
70 PRINT AT a-2, b+1; "  "
80 PRINT AT a-1, b; "  "
90 PRINT AT a, b+1; "O"  } O maiúsculo,
100 PRINT AT a, b+5; "O" } não zero
110 PAUSE  5
120 NEXT  b
130 STOP
```

2. b

```
10 REM
20 BORDER  6
40 LET  a=10
45 FOR  i=1 TO 6
47 INK  i
50 FOR  b=0 TO 24
70 PRINT  AT a-2, b+1; "  "
80 PRINT  AT a-1, b; "  "
90 PRINT  AT a, b+1; "O"
100 PRINT  AT a, b+5; "O"
110 PAUSE  5
120 NEXT  b
125 CLS  ← Isso e para apagar o carro
127 NEXT  do lado direito
130 STOP  da tela
```

3.



4.

```
10 REM  Quadrado indo do canto inferior
direito para o superior esquerdo
20 FOR  n=20 TO 1 STEP - 1
30 PRINT  AT n, n; "  "
40 PRINT  AT n-1, n-1; "  "
50 PAUSE  10
60 NEXT  n
70 STOP
```

5.

```
10 REM  Movendo o quadrado
20 LET  a=11
30 LET  b=16
40 PRINT  AT a-1, b; "  "
50 PRINT  AT a+1, b; "  "
60 PRINT  AT a, b-1; "  "
70 PRINT  AT a, b+1; "  "
80 PRINT  AT a, b; "  "
90 IF  INKEY$="5" THEN LET b=b-1
100 IF  b<1 THEN LET b=1
110 IF  INKEY$="6" THEN LET a=a+1
120 IF  a>20 THEN LET a= 20
130 IF  INKEY$="7" THEN LET a=a-1
140 IF  a<1 THEN LET a=1
150 IF  INKEY$="8" THEN LET b=b+1
160 IF  b>30 THEN LET b=30
170 GOTO 40
```

Unidade 24

1. a & b

```
 10 REM  Montanhas
 20 DIM  a$ (6, 11)
 30 DIM  x (6)
 40 DIM  c$ (6, 12)
 50 FOR  n=1 TO 6
 60 READ  a$ (n)
 70 DATA  "Everest", "Kilimanjaro", "Ta-
ble"; "Blanc", "Matterhorn", "Ben Nevis"
 80 NEXT  n
 90 FOR  m=1 TO 6
100 READ  x (m)
110 DATA  8848, 5895, 1087, 4807, 4478,
1347
120 NEXT  m
130 PRINT  "MONTANHA", "ALTURA
EM METROS"
140 PRINT
150 FOR  y=1 TO 6
160 PRINT  a$ (y), x (y)
170 NEXT  y
180 FOR  c=1 TO 6
190 READ  c$ (c)
200 DATA  "Nepal-Tibet", "Tanzania", "Afr
ica do Sul", "Franca", "Suica", "Escocia"
210 NEXT  c
220 PRINT  "MONTANHA", "PAIS"
230 FOR  w=1 TO 6
240 PRINT  a$ (w), c$ (w)
250 NEXT  w
260 STOP
```

2.

```
 10 REM  Invertendo numeros
 20 DIM  x (8)
 30 FOR  n=1 TO 8
 40 READ  x (n)
 50 DATA  10, 20, 30, 40, 50, 60, 70, 80
 60 NEXT  x
 70 FOR  y=8 TO 1 STEP - 1
 80 PRINT  x (y)
 90 NEXT  y
100 STOP
```

3.

```
 10 REM  Saltando numeros
 20 DIM  x (12)
 30 FOR  n=1 TO 12
 40 READ  x (n)
 50 DATA  5, 10, 15, 20, 25, 30, 35, 40,
45, 50, 55, 60
 60 NEXT  n
 70 FOR  y=1 TO 12 STEP 2
 80 PRINT  x (y)
 90 NEXT  y
100 STOP
```

4. a

```
 10 REM  Dias da semana
 20 DIM  d$ (7, 9)
 30 FOR  n=1 TO 7
 40 READ  d$ (n)
 50 DATA  "Segunda", "Terca", "Quarta",
"Quinta", "Sexta", "Sabado", "Domingo"
 60 NEXT  n
 70 INPUT  "Entre um numero entre 1 e
7 "; n
 80 PRINT  "Dia "; n; " e "; d$ (n)
 90 STOP
```

4. b — modificado

```
 10 REM  Dias da semana
 20 DIM  d$ (7, 9)
 30 FOR  n=1 TO 7
 40 READ  d$ (n)
 50 DATA  "Segunda", "Terca", "Quarta",
"Quinta", "Sexta", "Sabado", "Domingo"
 60 NEXT  n
 70 INPUT  "Entre um numero entre 1 e
7 "; n
 80 IF  n < 1 THEN PRINT "Numero
muito pequeno": GOTO 70
 90 IF  n > 7 THEN PRINT "Numero
muito grande": GOTO 70
100 PRINT  "Dia"; n; "é"; d$ (n)
110 STOP
```

4. c

Adicione ao programa acima:

```
 65 FOR  y=1 TO 4
105 NEXT  Y
```

5.

```
 10 REM  Nomes dos meses
 20 DIM  m$ (12, 9)
 30 DIM  w$ (1, 9)
 40 FOR  x=1 TO 12
 50 READ  m$ (x)
 60 DATA  "Janeiro", "Fevereiro", "Mar
co", "Abril", "Maio", "Junho", "Julho",
"Agosto", "Setembro", "Outubro",
"Novembro", "Dezembro"
 70 NEXT  x
 80 INPUT  "Entre uma palavra "; w$ (1)
 90 FOR  y=1 TO 12
100 IF  m$ (y)=w$ (1)  THEN  PRINT
m$ (y); "e mes"; y: GOTO 130
```

```
110 NEXT y
120 PRINT w$ (1); " não é um mês"
130 STOP
```

6.

```
10 REM Como a eletricidade e produzida
20 DIM e$ (4,7)
30 DIM p (4)
40 FOR x=1 TO 4
50 READ e$ (x), p (x)
60 DATA "petroleo", 79, "nuclear", 8,
"hidroeletrica", 11, "carvao", 2
70 NEXT x
80 PRINT "COMO A ELETRICIDADE
É PRODUZIDA"
90 PRINT TAB 13; "NA INGLATERR
A"
100 PRINT "FONTE DE"; TAB 16; "%
TOTAL"
110 PRINT TAB 3; "ELETRICIDADE";
TAB 22; "PRODUZIDA"
120 PRINT "————————————————
————————"
130 FOR y=1 TO 4
140 PRINT e$ (y); TAB 21; p (y)
150 PRINT
160 NEXT y
170 STOP
```

7.

```
a$ (2, 1) = Paulo
a$ (1, 3) = Luiza
a$ (2, 4) = Marcos
```

8.

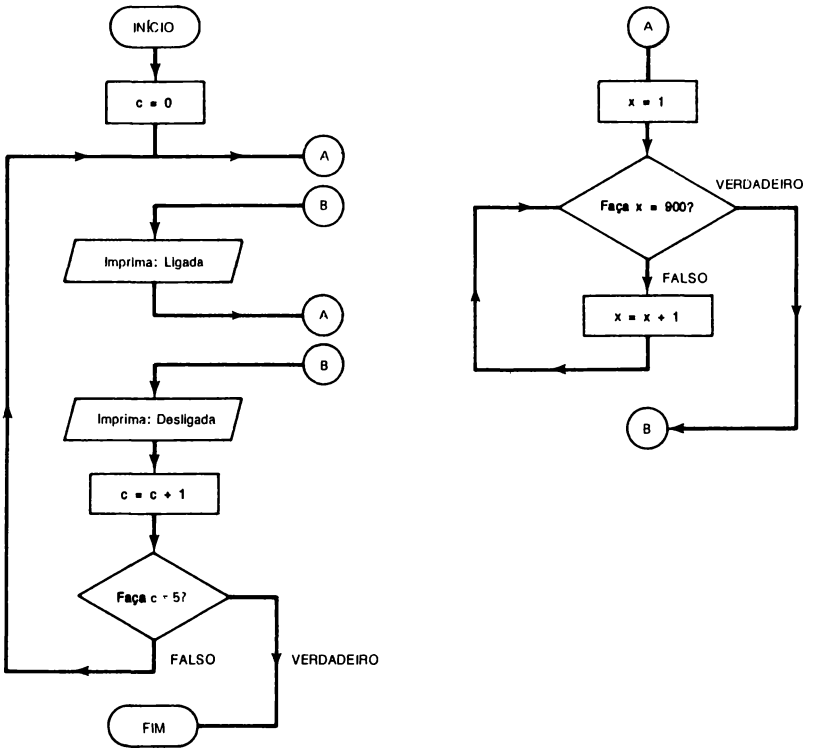
```
10 REM Esportistas
20 DIM s$ (5, 2, 12)
30 PRINT "NOME", "ESPORTE"
40 FOR x=1 TO 5
50 FOR y=1 TO 2
60 READ s$ (x, y)
70 DATA "J. Connors", "Tenis", "S.
"Cram", "Atletismo", "S. Davis", "Sinuca",
"L. Pignott", "Hipismo", "F. Bruno", "Boxe"
80 PRINT s$ (x, y)
90 NEXT y
100 PRINT
110 NEXT x
120 STOP
```

9.

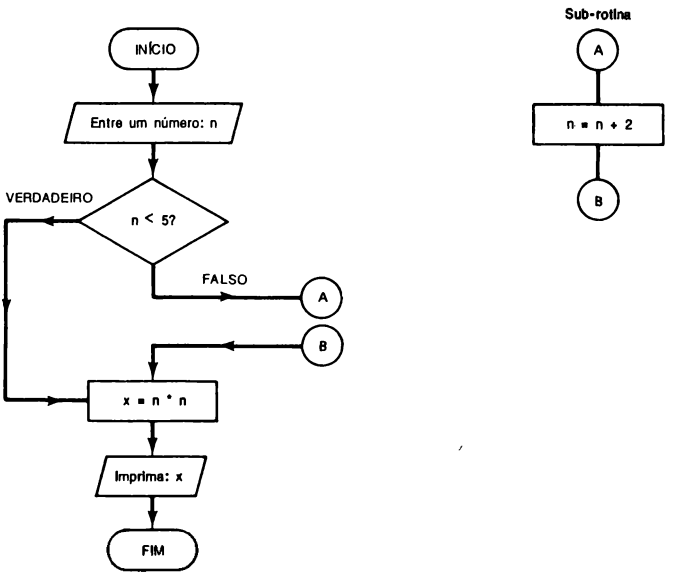
```
10 REM Riqueza mundial
20 DIM t (3, 3)
30 PRINT "DIVISAO ANUAL DA
RIQUEZA EM %"
40 PRINT TAB 7;"DESENVOLVIDOS"
; TAB 1; "EM DESENVOLVIMENTO"
50 PRINT TAB 7; "PAISES"; TAB 19;
"PAÍSES"
60 PRINT "————————————————
————————"
70 FOR x=1 TO 3
80 FOR y=1 TO 3
90 READ t (y, x)
100 DATA 1950, 68, 32, 1960, 79, 21,
1970, 82, 18
110 PRINT t (y, x); "  ";
120 NEXT y
130 PRINT
140 NEXT x
150 STOP
```

Unidade 25

1. Fluxograma



2. Fluxograma



3.

a) 49 b) 4

4.

```
10 REM  Questoes sobre sua casa
20 PRINT  "Quantos quartos tem? ";
30 INPUT  "Entre um numero "; n
40 PRINT  n
50 GOSUB 200
60 PRINT  "Qual o numero da sua
casa? ";
70 INPUT  "Entre um numero "; x
80 PRINT  x
90 GOSUB 200
100 PRINT  "Tem jardim? ";
110 INPUT  "Entre sim ou nao "; a$
120 PRINT  a$
130 GOSUB 200
140 PRINT  "Tem garagem? ";
150 INPUT  "Entre sim ou nao "; b$
160 PRINT  b$
170 GOSUB 200
180 STOP
200 REM  Sub-rotina para asteriscos
210 FOR  n=0 TO 31
220 PRINT  "*";
230 NEXT  n
240 RETURN
```

5.

```
10 REM  Ex25.5
20 INPUT  "Entre um numero entre 1 e
100"; n
30 IF  n < 50 THEN GOSUB 100
40 FOR  x=0 TO n STEP 5
50 PRINT  x
60 NEXT  x
70 STOP
100 REM  Sub-rotina
110 LET  n=n+25
120 RETURN
```

6.

```
10 REM  Adivinhando um numero
20 PRINT  "Feche os olhos, jogador"
30 INPUT  "Entre um numero a ser
adivinhado entre 1 e 100"; a
40 INPUT  "Jogador, entre um numero
entre 1 e 100"; n
50 PRINT  n
60 IF  n=a THEN PRINT "Correto":
GOTO 20
70 IF  n < a THEN GOSUB 100
80 IF  n < a THEN GOSUB 200
90 GOTO  40
100 REM  Sub-rotina
110 PRINT  "Muito alto, tente outra vez"
120 RETURN
200 REM  Sub-rotina
210 PRINT  "Muito baixo, tente outra vez"
220 RETURN
```

7.

```
10 REM  ABRINDO GARRAFAS
20 INPUT  "Entre o tipo da garrafa: rolha
ou tampa "; b$
30 IF  b$ = "rolha" THEN GOSUB 200
40 IF  b$ = "tampa" THEN GOSUB 300
50 IF  b$ < > "rolha" AND b$ < >
"tampa" THEN PRINT "Tipo errado": GO
TO 20
60 PRINT
70 PRINT  "Garrafa aberta. Posso servir
seu copo?"
80 PRINT
90 PRINT  "Delicia!!"
100 STOP
200 REM  Sub-rotina para abrir a garrafa
com rolha
210 PRINT  "1. Pegue o saca-rolhas e ajus
te na rolha da garrafa"
220 PRINT  "2. Com movimentos circula
res perfure a rolha com o saca-rolhas"
230 PRINT  "3. Puxe o saca-rolhas para
fora"
240 RETURN
300 REM  Sub-rotina para abrir a garrafa
com tampa
310 PRINT  "1. Pegue o abridor"
320 PRINT  "2. Encaixe na tampa"
330 PRINT  "3. Segurando a garrafa, faca
um movimento de alavanca com o abridor"
340 RETURN
```

8.

```
10 REM  Pintando paredes
20 LET  r=0.15
30 INPUT  "Entre o numero de paredes a
serem pintadas "; n
40 LET  t=0
50 FOR  w=1 TO n
60 INPUT  "Entre largura da parede "; f
70 GOSUB 300
80 LET  l=m
90 INPUT  "Entre altura da parede "; f
100 GOSUB 300
110 LET  h=m
```

```
120  LET  a=1*h
130  PRINT  "Area da parede "; a
140  LET  t=t+a
150  PRINT  "Area total ="; t
160  PRINT
170  NEXT  w
180  LET  p=t*r
190  PRINT  "Quantidade de tinta necessa-
ria e "; p; " litros"
200  STOP
300  REM  Sub-rotina para passar pes para
metros
310  LET  m=f/3.208
320  RETURN
```

9.

Distâncias entre cidades

Linha 20: Cria um array de strings unidimensional com 7 elementos, cada um com no máximo 10 caracteres, chamado t$. Os dados são fornecidos pelas linhas 50 a 80. t$ armazena os nomes das cidades.

Linha 30: Cria um array numérico bidimensional, 'd', com 49 elementos — 7 colunas e 7 linhas. Os laços aninhados (linhas 90 a 140) colocam as distâncias entre as cidades neste array.

Linha 40: Cria um array de string unidimensional com um elemento de no máximo 10 caracteres. A informação para esse array é entrada pelo usuário nas linhas 150 a 190.

Linha 150: A cidade para onde se vai é entrada em c$ (1) e o computador vai para sub-rotina.

Linha 300-350: A sub-rotina compara a cidade entrada em c$ (1) com todas as cidades em t$ ( ).
Se c$ (1) for igual a alguma das cidades em t$ ( ) então o computador retorna ao programa. Caso contrário, atribui 0 a x e retorna.

Linha 170: Na volta da sub-rotina, se x = 0 então a cidade não foi encontrada, e uma mensagem de "Sem informações" é enviada e o usuário é remetido de volta à linha 150, onde uma outra cidade pode ser entrada.

Linha 180: Se a cidade existe em t$, então x terá um valor entre 1 e 7, dependendo de qual for o elemento de t$ encontrado. Armazenamos x em y, para uso posterior, pois x vai ser reutilizado, e por isso será alterado.

Linha 190: A cidade de onde se vem é entrada em c$ (1); o computador vai para a sub-rotina onde o processo de verificação da cidade é feito de novo (veja as linhas 300-350).

Linha 210: É quase o mesmo da linha 170; se x = 0 então não há informação sobre a cidade, e o usuário volta à liha 190, para entrar outra cidade.

Linha 220: A distância entre as cidades é impressa. Isso é feito buscando-se nomes no array t$ ( ). Uma cidade está armazenada em t$ (y) — veja a linha 180 — e a outra em t$ (x). Os números contidos em y e x são então usados para buscar a distância no array 'd'. Por exemplo, se y = 5 e x = 3 t$ (5) é Liverpool e t$ (3) é Plymouth o elemento d (5, 3) é buscado, dando o valor 278, como pode ser visto na tabela a seguir.

	1 L´DN	2 T´TN	3 P´MTH	4 N´HAM	5 L´PL	6 LEEDS	7 HULL
1 LONDON	0	144	211	122	197	190	168
2 TAUNTON	144	0	74	180	203	237	247
3 PLYMOUTH	211	74	0	254	278	312	321
4 NOTTINGHAM	122	180	254	0	97	67	73
5 LIVERPOOL	197	203	278	97	0	73	128
6 LEEDS	190	237	312	67	73	0	55
7 HULL	168	247	321	73	128	55	0

Unidade 26

1. E Acima = 16 B Abaixo = − 13

2. Inclua:

```
15 FOR n = 1 TO 3
25 NEXT n
```

3. Inclua:

```
22 PAUSE 3 * 50
```

4. a

```
10 REM Escala do C maior toda em semi colcheias
20 BEEP .25, 0: BEEP .25, 2: BEEP .25, 4: BEEP .25, 5: BEEP .25, 7: BEEP .25, 9: BEEP .25, 11: BEEP .25, 12
30 STOP
```

b

```
10 REM Escala do C maior com semini mas e colcheias
20 BEEP .5, 0: BEEP .5, 2: BEEP 1, 4: BEEP .5, 5: BEEP .5, 7: BEEP 1, 9: BEEP .5, 11: BEEP .5, 12
30 STOP
```

c

```
10 REM Escala do C maior com varias notas diferentes
20 BEEP 2, 0: BEEP .25, 2: BEEP .25, 4: BEEP .25, 5: BEEP .25, 7: BEEP .5, 9: BEEP .5, 11: BEEP 4, 12
30 STOP
```

5.

```
10 REM London's Burning
20 FOR x=1 TO 2
30 BEEP .5, 2: BEEP .5, 2: BEEP 1, 7: BEEP 1, 7
40 NEXT x
50 FOR y=1 TO 2
60 BEEP .5, 9: BEEP .5, 9: BEEP 1, 11: BEEP 1, 11
70 NEXT y
80 FOR w=1 TO 2
90 BEEP 1, 14: BEEP 2, 14
100 NEXT w
110 FOR z=1 TO 2
120 BEEP .5, 14: BEEP .5, 12: BEEP 1, 11: BEEP 1, 11
130 NEXT z
140 STOP
```

6. a

```
10 REM Escala do G maior
20 BEEP .5, −5: BEEP .5, −3: BEEP .5, − 1: BEEP .5, 0: BEEP .5, 2: BEEP .5, 4: BEEP .5, 6: BEEP .5, 7
30 STOP
```

```
b
G   A   B   C   D   E   F#  G
7   9   11  12  14  16  18  19
C
10  REM  Escala do G maior
20  BEEP .5, -5: BEEP .5, -3: BEEP
.5, -1: BEEP .5, 0: BEEP .5, 2: BEEP
.5, 4: BEEP .5, 6: BEEP .5, 7
30  BEEP .5, 9: BEEP .5, 11: BEEP .5,
12: BEEP .5, 14: BEEP .5, 16: BEEP .5,
18: BEEP .5, 19
40  STOP
```

7.

```
10  REM  Good King Wenceslas
20  FOR  n=1 TO 2
30  BEEP 1, 7: BEEP 1, 7: BEEP 1, 7:
BEEP 1, 9: BEEP 1, 7: BEEP 1, 7: BEEP
2, 2: BEEP 1, 4: BEEP 1, 2: BEEP 1, 4:
BEEP 1, 6: BEEP 2, 7: BEEP 2, 7
40  NEXT  n
50  GOSUB 200
60  BEEP 1, 11: BEEP 1, 9: BEEP 2, 7:
BEEP 1, 4: BEEP 1, 2: BEEP 1, 4: BEEP
1, 6: BEEP 2, 7: BEEP 2, 7
70  BEEP 1, 2: BEEP 1, 2: BEEP 1, 4:
BEEP 1, 6: BEEP 1, 7: BEEP 2, 9
80  GOSUB 200
90  BEEP 2, 7: BEEP 2, 12: BEEP 4, 7
100 STOP
200 REM  Sub-rotina
210 BEEP 1, 14: BEEP 1, 12: BEEP 1, 11:
BEEP 1, 9
220 RETURN
```

8.

```
10  REM  The Holly and the Ivy
20  FOR  n=1 TO 4
30  FOR  x=1 TO 29
40  READ  a, b
50  BEEP  a, b
60  DATA  1, 7, .5, 7, .5, 7, 1, 7, 1, 16,
1, 14, 1, 11, 1, 7, .5, 7, .5, 7, 1, 7, 1,
16, 2, 14, .5, 14, .5, 12, .5, 11, .5, 9, 1,
7, 1, 11, .5, 4, .5, 4, 1, 2, 5, 7, .5, 9, .5,
11, .5, 12, 1, 11, 1, 9, 2, 7
70  NEXT  x
80  PAUSE  2*50
90  RESTORE
100 NEXT  n
110 STOP
```

Unidade 27

1. a 4 b 103 c 99
   d −6 e −88 f −1
   g 0

2. **Linha 30:** LET X = INT (21*RND) não funciona como LET X = INT (20*RND), pois INT arredonda para baixo; 21 passará automaticamente a gerar números só até 20, que é o que queremos.

3. Inclua:

```
65  INPUT  "Entre o numero que você
deseja que seja contado"; n
```

Alter:

```
70  IF  d = n THEN LET c = ç + 1
120 PRINT  "O numero de vezes que ";
n; "apareceu foi"; c
```

4.

```
10  REM  Lista de 20 numeros
20  FOR  n=1 TO 20
30  LET  x=INT (101*RND)
40  PRINT  x
50  NEXT  n
60  STOP
```

5.

```
10  REM  Lista de 15 numeros
20  FOR  n=1 TO 15
30  LET  x=INT (20*RND)+1
40  PRINT  x
50  NEXT  n
60  STOP
```

6.

```
10  REM  Mesma sequencia de numeros
aleatórios
20  RANDOMIZE 600
30  FOR  n=1 TO 10
40  LET  x=INT (11*RND)
50  PRINT  x
60  NEXT  n
70 STOP
```

7.

```
10  REM  Quantas vezes numeros de 1 a
5 ocorrem dentre 100 numeros aleatorios
20  DIM  a (5)
30  FOR  y=1 TO 5
40  READ  a (y)
```

```
 50  DATA  0, 0, 0, 0, 0
 60  NEXT  Y
 70  FOR  n=1 TO 10
 80  FOR  m=1 TO 10
 90  LET  s=INT (5*RND)+1
100  PRINT  s; " ";
110  FOR  x=1 TO 5
120  IF  s = x THEN LET a (x) = a (x) + 1
130  NEXT  x
140  NEXT  m
150  PRINT
160  NEXT  n
170  PRINT
180  PRINT  "NUMERO", "APARECEU"
190  FOR  w=1 TO 5
200  PRINT  w, a (w)
210  NEXT  w
220  STOP
```

8.

```
 10  REM  Testando tabuadas de multipli
 cacao
 20  LET  x=INT (11*RND)+2
 30  LET  y=INT (11*RND)+2
 40  PRINT  x; "*"; y; "="
 50  INPUT  "Entre uma resposta "; a
 60  PRINT  a
 70  IF  a=x*y THEN PRINT "Certo":
 GOTO 90
 80  PRINT  "Errado, a resposta e "; x*y
 90  STOP
```

9. a

```
 10  REM  Jogando 2 dados
 20  LET  c=0
 30  FOR  n=1 TO 50
 40  LET  d1=INT (6*RND)+1
 50  LET  d2=INT (6*RND)+1
 60  LET  s=d1+d2
 70  IF  s=7 THEN LET c=c+1
 80  NEXT  n
 90  PRINT  "O numero de vezes que o 7
 ocorreu foi "; c
100  STOP
```

b

```
 10  REM  Jogando 2 dados
 20  LET  t=0
 30  FOR  x=1 TO 5
 40  LET  c=0
 50  FOR  n=1 TO 50
 60  LET  d1=INT (6*RND)+1
 70  LET  d2=INT (6*RND)+1
 80  LET  s=d1+d2
 90  IF  s=7 THEN LET c=c+1
100  NEXT  n
110  PRINT  "O numero de vezes que o 7
ocorreu foi "; c
120  LET  t=t+c
130  PRINT  "O total de vezes que o 7
ocorreu foi "; t
140  NEXT  x
150  PRINT
160  PRINT  "A media de ocorrencias do 7
e "; t/5
170  STOP
```

10.

```
 10  REM  Contando caras e coroas
 20  LET  h=0
 30  LET  t=0
 40  FOR  n=1 TO 80
 50  LET  x=INT (2*RND)
 60  IF  x=0 THEN LET h=h+1: GOTO
 80
 70  LET  t=t+1
 80  NEXT  n
 90  PRINT  "O numero de caras e "; h
100  PRINT
110  PRINT "O numero de coroas e "; t
120  STOP
```

Unidade 28

1. A 7,7 B 16,8
   C 0,15 D 31,11
   E 0,0 F 21,0
   G 11,9

2. a

```
 10  REM  Linha A
 20  PLOT  16, 130
 30  DRAW  80, 0
 40  REM  Linha B
 50  PLOT  24, 72
 60  DRAW  28, 24
 70  REM  Linha C
 80  PLOT  183, 40
 90  DRAW  - 127, 0
100  REM  Linha D
110  PLOT  116, 167
120  DRAW  0, - 103
130  REM  Linha E
140  PLOT  136, 119
150  DRAW  111, - 103
160  REM  Linha F
170  PLOT  231, 88
180  DRAW  - 79, 71
```

b

```
190 PRINT AT 4, 7; "A"
200 PRINT AT 10, 4; "B"
210 PRINT AT 15, 11; "C"
220 PRINT AT 8, 13; "D"
230 PRINT AT 11, 23; "E"
240 PRINT AT 6, 26; "F"
250 STOP
```

3.

```
10 REM Shape A
20 PLOT 32, 32
30 DRAW 0, 39
40 DRAW 39, 0
50 DRAW 0, - 39
60 DRAW - 39, 0
70 PRINT AT 15, 6; "A"
100 REM Shape B
110 PLOT 48, 88
120 DRAW 23, 71
130 DRAW 24, - 63
140 DRAW - 47, - 8
150 PRINT AT 7, 9; "B"
200 REM Shape C
210 PLOT 120, 152
220 DRAW 15, 15
230 DRAW 80, - 72
240 DRAW - 15, - 15
250 DRAW - 80, 72
260 PRINT AT 6, 21; "C"
300 REM Shape D
310 PLOT 96, 24
320 DRAW 0, 47
330 DRAW 31, 32
340 DRAW 48, - 32
350 DRAW 0, - 47
360 DRAW - 79, 0
370 PRINT AT 14, 16; "D"
400 REM Shape E
410 PLOT 192, 39
420 DRAW 0, 17
430 DRAW 15, 15
440 DRAW 17, 0
450 DRAW 15, - 15
460 DRAW 0, - 17
470 DRAW - 15, - 15
480 DRAW - 17, 0
490 DRAW - 15, 15
500 PRINT AT 15, 27; "E"
510 STOP
```

4.

```
10 REM Desenhando triangulos
20 INPUT "Entre coordenada x "; x
30 IF x<0 OR x>255 THEN PRINT
"Nao aceitavel": GOTO 20
40 INPUT "Entre coordenada y "; y
50 IF y<0 OR y>175 THEN PRINT
"Inaceitavel": GOTO 40
60 INPUT "Entre com a largura "; w
70 IF x+w>225 THEN PRINT "Muito
largo": GOTO 60
80 INPUT "Entre com a altura "; h
90 IF y+h>175 THEN PRINT "Muito
alto": GOTO 80
100 CLS
110 PLOT x, y
120 DRAW w/2, h
130 DRAW w/2, - h
140 DRAW - w, 0
150 STOP
```

5.

Adicione ao programa anterior:

```
92 INPUT "O que voce deseja desenhar?
Entre t para triangulo e p para pentagono "
; a$
94 IF a$ = "t" THEN GOTO 100
96 IF a$ = "p" THEN GOTO 160
160 CLS
170 PLOT x, y
180 DRAW 0, h/2
190 DRAW w/2, h/2
200 DRAW w/2, - h/2
210 DRAW 0, - h/2
220 DRAW - W, 0
230 STOP
```

6.

```
10 REM quadro
20 REM arvores
30 FOR n = 1 TO 20
40 READ x, y
50 DATA 20, 96, 24, 52, 40, 40, 54, 40,
68, 100, 80, 124, 92, 112, 104, 132, 114,
112, 124, 132, 132, 112, 100, 48, 112, 64,
140, 32, 160, 32, 108, 104, 196, 96, 216,
128, 236, 124, 244, 52
60 PLOT x, y
70 DRAW 0, 8
80 DRAW - 7, 0
90 DRAW 7, 32
100 DRAW 7, - 32
110 DRAW - 7, 0
120 NEXT n
130 REM casas
140 FOR h = 1 TO 4
150 READ x, y
160 DATA 32, 96, 64, 40, 144, 112, 20,
8, 72
170 PLOT x, y
```

```
180 DRAW 0,24
190 DRAW 12, 12
200 DRAW 12, - 12
210 DRAW 0, - 24
220 DRAW - 16, 0
230 DRAW 0, 16
240 DRAW 8, 0
250 DRAW 0, - 16
260 DRAW - 16, 0
270 NEXT h
280 REM ruas
290 PLOT 0, 40
300 DRAW 103, 0
310 DRAW 37, 56
320 DRAW - 84, 0
330 DRAW 0, 8
340 DRAW 91, 0
350 DRAW 5, 8
360 DRAW 8, 0
370 DRAW - 20, - 32
380 DRAW 68, 0
390 DRAW 0, - 8
400 DRAW - 74, 0
410 DRAW - 24, - 40
420 DRAW - 109, 0
430 STOP
```

8.

Mantenha as linhas 10 – 110
Inclua:

```
120 FOR n = 1 to h - 1
130 PLOT x, y + n
140 DRAW w - 1, 0
150 NEXT n
160 STOP
```

9.

```
 10 REM NAVIO VIKING
 20 REM Casco do navio
 30 PLOT 72, 72
 40 DRAW 104, 24, 2.5
 50 DRAW 8, - 4
 60 DRAW - 8, - 4
 70 REM Escudo
 80 LET y = 68
 90 LET r = 8
100 FOR s = 1 TO 5
110 READ x
120 DATA 153, 136, 120, 103, 86
130 CIRCLE x, y, r
140 NEXT s
150 REM Mastro
160 PLOT 120, 76
170 DRAW 0, 88
180 REM Bandeira
190 DRAW - 20, - 5
200 DRAW 20, - 5
210 REM Velas
220 DRAW 0, - 4
230 DRAW 54, 8
240 DRAW - 8, - 68
250 DRAW - 100, - 12
260 DRAW 8, 64
270 DRAW 46, 8
280 REM remos
290 LET y = 32
300 FOR 0 = 1 TO 5
310 READ x
320 DATA 64, 82, 100, 118, 136
330 PLOT x, y
340 DRAW 19, 28
350 NEXT o
360 PRINT AT 20, 9; "NAVIO
VIKING"
370 STOP
```

Unidade 29

1.

a

```
 10 REM Baixinho gordo
 20 POKE USR "M", BIN 00011100
 30 POKE USR "M", + 1, BIN 00011100
 40 POKE USR "M", + 2, BIN 00001000
 50 POKE USR "M", + 3, BIN 00111110
 60 POKE USR "M", + 4, BIN 01011101
 70 POKE USR "M", + 5, BIN 00011100
 80 POKE USR "M", + 6, BIN 00100010
 90 POKE USR "M", + 7, BIN 01000001
100 PRINT "M" (Entrando em modo
                gráfico)
110 STOP
```

b

```
 10 REM casa
 20 POKE USR "H", BIN 00111000
 30 POKE USR "H" + 1, BIN 01111100
 40 POKE USR "H" + 2, BIN 11111110
 50 POKE USR "H" + 3, BIN 10101010
 60 POKE USR "H" + 4, BIN 11111110
 70 POKE USR "H" + 5, BIN 10101010
 80 POKE USR "H" + 6, BIN 11101110
 90 POKE USR "H" + 7, BIN 00000000
100 PRINT "H" (Entrando em modo
                gráfico)
110 STOP
```

2.

```
 10 REM Movendo carro
 20 POKE USR 'C", BIN 00000000
```

```
 30  POKE  USR "C", + 1, BIN 00000000
 40  POKE  USR "C", + 2, BIN 00111100
 50  POKE  USR "C", + 3, BIN 01010100
 60  POKE  USR "C", + 4, BIN 11111111
 70  POKE  USR "C", + 5, BIN 11111111
 80  POKE  USR "C", + 6, BIN 01100110
 90  POKE  USR "C", + 7, BIN 00000000
100  LET  a = 11
110  FOR  b = 0 TO 30
120  PRINT  AT a, b; " "
130  PRINT  AT a, b + 1; "C" (Entrando
                              em modo
                              gráfico)
140  NEXT  b
150  STOP
```

Unidade 30

1.

```
 10  REM Comprimento de palavras
 20  PRINT  "PALAVRA", "TAMANH
     O"
 30  FOR  n = 1 TO 8
 40  INPUT "Entre uma palavra "; a$
 50  PRINT a$, LEN a$
 60  NEXT  n
 70  STOP
```

Impresso na
ERCA Editora e Gráfica Ltda.
Rua Silva Vale, 870 - Cavalcante
Rio de Janeiro - RJ

JUDITH MILLER

DOMINANDO O TK90X E TK95

DOMINANDO O TK 90X® E TK 95®

JUDITH MILLER

Este livro foi escrito numa linguagem acessível onde cada conceito de programação é explicado de forma detalhada e acompanhado de exemplos de aplicação.

O objetivo principal é o domínio completo do BASIC no TK 90X e TK 95.

Além de se constituir num verdadeiro guia de programação para os usuários destes micros, o livro pode ser perfeitamente utilizado no ensino de 1º e 2º graus, pois a sua apresentação didática é primorosa.

MAIS UM LANÇAMENTO
DA
LIVROS TÉCNICOS E CIENTÍFICOS EDITORA S.A.

ISBN: 85-216-0511-0