# *AUTORIZAÇÃO*

**O download deste material é autorizado pelo autor, o Prof. Wilson José Tucci, desde que seja mantido o crédito e não seja usado para fins lucrativos.**

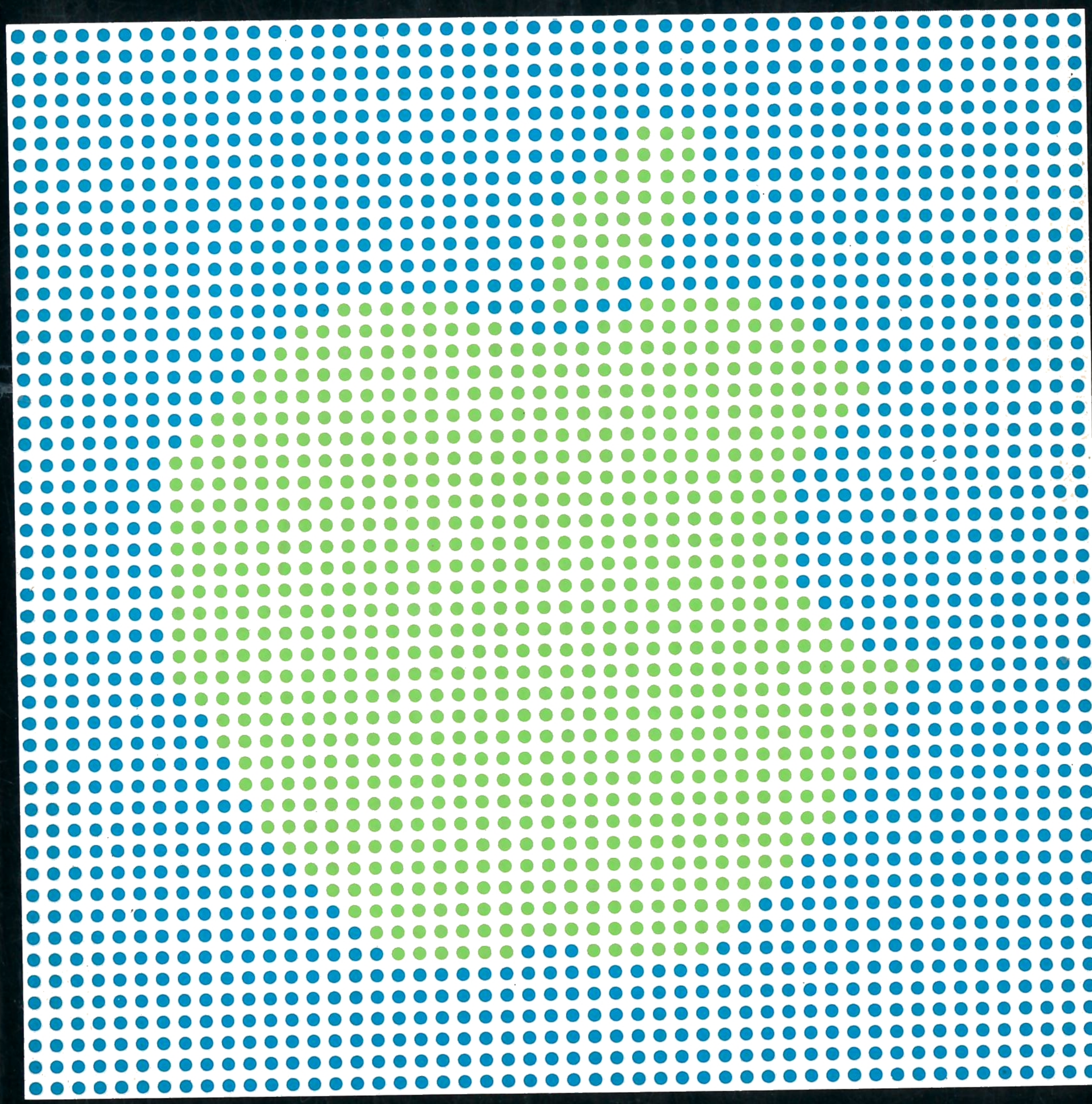
**Agradecemos ao Prof. Tucci que foi um "Desbravador" desde o começo da informática no Brasil, se dedicou de forma simples e direta a estimular a nossa curiosidade para esse mercado que tanto cresceu no País.**

**Muito Obrigado!**

**Equipe Datassette.org**

Wilson J. Tucci

# por dentro do APPLE



nobel

A sair:
POR DENTRO DO **DOS**
POR DENTRO DO **LOGO**
POR DENTRO DO **VISICALC**

POR DENTRO DO APPLE leva o leitor, passo a passo, através da linguagem do APPLE, desde um nível introdutório até a apresentação de técnicas avançadas para otimizar o processamento de programas no computador, através de exemplos e aplicações práticas.

Servindo como texto fundamental e como modelo didático-pedagógico aos cursos de BASIC e de fundamentos de processamento de dados, o livro dirige-se a estudantes, profissionais e mesmo a pessoas que não tenham conhecimento prévio de computação.


ISBN 85-213-0193-6

Tucci

por dentro do APPLE

nobel

Wilson J. Tucci

# por dentro do APPLE

nobel

# POR DENTRO DO APPLE

CIP-Brasil. Catalogação-na-Publicação
Câmara Brasileira do Livro, SP

T824p Tucci, Wilson José, 1946-
Por dentro do Apple / Wilson José Tucci. — São Paulo : Nobel, 1984.

Bibliografia.
ISBN 85-213-0193-6

1. Apple (Computador) 2. BASIC (Linguagem de programação para computadores) 3. Linguagem de programação (Computadores) 4. Programação (Computadores eletrônicos) I. Título.

17. CDD-651.8
18. -001.64
18. -001.642
18. -001.6424

83-1719

Índices para catálogo sistemático:

1. Apple : Computadores eletrônicos : Processamento de dados 651.8 (17.) 001.64 (18.)
2. BASIC : Linguagem de programação : Computadores : Processamento de dados 651.8 (17.) 001.6424 (18.)
3. Computadores: Linguagem de programação : Processamento de dados 651.8 (17.) 001.6424 (18.)
4. Computadores : Programação : Processamento de dados 651.8 (17.) 001.642 (18.)
5. Linguagem de programação : Computadores : Processamento de dados 651.8 (17.) 001.6424 (18.)
6. Programação de computadores : Processamento de dados 651.8 (17.) 001.642 (18.)

CAPA: Regina Knoll

FOTO DA CAPA: O autor e o equipamento μ6502, gentilmente cedido pela Appletrônica Computadores e Sistemas Ltda. (Foto de Miguel Benevides)

Wilson José Tucci

# POR DENTRO DO APPLE

## UMA ABORDAGEM DO "APPLESOFT"

**Colaboradores:**

DANIEL ROBERTO FALCONER
JOSÉ EDUARDO MOREIRA

**2ª edição**

**1985**

LIVRARIA NOBEL S.A.
EDITORA — DISTRIBUIDORA
LOJA 1: RUA DA CONSOLAÇÃO, 49 - CEP01301
LOJA 2: RUA MARIA ANTÔNIA, 108 - CEP01222
LOJA 3: RUA PEDROSO ALVARENGA, 704 - CEP04531
LOJA 4: RUA BARÃO DO TRIUNFO, 371 - CEP04602
EDITORA: RUA DA BALSA, 559 - CEP02910
FONES: (PABX) 857-9444 e 257-2144 - SÃO PAULO - S.P.

Copyright by **Livraria Nobel S.A.**

**É PROIBIDA A REPRODUÇÃO**

Nenhuma parte desta obra poderá ser reproduzida, **sem a permissão por escrito dos editores,** através de qualquer meio: XEROX, FOTOCÓPIA, FOTOGRÁFICO, FOTOMECÂNICO. Tampouco poderá ser copiada ou transcrita, nem mesmo transmitida através de meios eletrônicos ou gravações. Os infratores serão punidos através da Lei 5.988, de 14 de Dezembro de 1973, artigos 122-130.

Impresso no Brasil/Printed in Brazil

*Este livro é dedicado a todos que possuem sensibilidade e imaginação...*

*... se o príncipe da Borralheira conhecesse informática, não precisaria mover céus e terras pela sua Cinderela...*

*... eles teriam mais tempo para si mesmos.*

# Prefácio

Com o advento do microprocessador, uma nova era começou a surgir em nosso mundo, e a cada dia que passa sentimos cada vez mais de perto a sua influência em nosso meio de vida.

Hoje, a imaginação do homem faz com que um microprocessador seja utilizado, desde um simples brinquedo, painel de automóvel, forno de microondas etc., até em complexos sistemas de controle de processo, controles de comunicação, microcomputadores etc.

A grande preocupação dos fabricantes do "Ouro Energético" ou microprocessadores é com a sua aplicabilidade industrial, esquecendo-se de outro ponto fundamental que é a disseminação dessa incrível cultura para os nossos jovens que hoje ocupam os bancos das escolas.

Se esse ainda não é o ponto forte enfocado pelas indústrias, ele o é por pessoas que dedicam o seu tempo particular para traduzir numa linguagem simples, clara e objetiva os conceitos básicos e avançados para controlar as forças dos microprocessadores.

Como indicação desse tipo de dedicação, temos o exemplo prático do Prof. Wilson José Tucci, que tem utilizado toda a sua experiência de pedagogo e seus profundos conhecimentos sobre a mina de "Ouro Energético" para traduzir de uma maneira simples, objetiva e didática os conceitos da linguagem "Basic".

Vocês terão a oportunidade de verificar, em cada capítulo desse livro, que o Prof. Wilson José Tucci utilizou exemplos práticos para elucidar conceitos bastante teóricos, o que propicia um entendimento fácil da linguagem "Basic".

Na minha opinião, esse livro reúne totais condições para suprir a necessidade que o mercado de informática possui na área de treinamento da linguagem "Basic".

Jarbas Coura Mendes
Gerente de Sistemas
da Kodak do Brasil

# Agradecimentos

Este livro não aconteceria se algumas pessoas e instituições não tivessem participado e colaborado.

Sou sinceramente agradecido a:

Daniel Roberto Falconer e José Eduardo Moreira, colegas e companheiros, pelas sugestões, testes, comentários e críticas aos programas e exercícios.

Meus alunos, eternos questionadores, fazedores e professores, pois, sem eles, este livro não teria sentido e validade.

Pueri Domus Escola Experimental Ltda. pela confiança e credibilidade no nosso trabalho e também pela crença, visão e pioneirismo.

Ricardo Amaral Gurgel, amigo e colega, pelos longos "papos" e troca de idéias que germinaram esse trabalho.

Appletrônica Computadores e Sistemas por prestar todo o apoio técnico e informações específicas, além de fornecer o equipamento indispensável para o desenvolvimento dos programas.

Polymax Sistemas e Periféricos pelo fornecimento do equipamento no qual foram executados em grande parte os programas.

José Lopes, da livraria LITEC, batalhador e divulgador incansável em prol do livro técnico nacional.

Livraria Nobel por manter uma equipe editorial com um "pique" de atendimento e apoio incríveis.

O autor.

# Sumário

## NOTA SOBRE AS ABERTURAS DOS CAPÍTULOS

As figuras que definem as aberturas dos capítulos foram inspiradas nos símbolos do *I Ching* ou *Livro das Mutações,* cuja origem remonta à China pré-histórica. A autoria do texto do *I Ching,* segundo a tradição, é atribuída a quatro santos sábios: o imperador Fu Hsi (aproximadamente 3.000 a.C.), o rei Wen e seu filho, o Duque de Chou (por volta do século XII a.C.), e Confúcio (século VI a.C.).

Difundido por todo o mundo, sendo um dos livros mais antigos da humanidade, o *I Ching* serviu de apoio aos princípios fundamentais do confucionismo e do taoísmo.

O *Livro das Mutações* é usado para consultas sobre determinadas situações e problemas humanos e também fenômenos naturais.

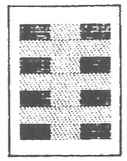
Sua simbologia consta de 64 hexagramas, construídos a partir da combinação de 6 barras interrompidas (Yin) ou contínuas (Yang), acompanhados de textos explicativos para cada símbolo. A seleção do hexagrama é feita com o auxílio de varetas de bambu ou moedas.

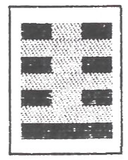
Nesta adaptação, barras escuras percorrem, segundo uma codificação binária, posições que geram os números compreendidos entre 0 e 15.

Ainda inspirado no *I Ching,* "bolamos" um programa para você consultar o computador e receber uma "dica" para o seu dia. Digite o programa e, após executá-lo, introduza o seu nome e o dia do mês. Consulte, em seguida, o símbolo resultante e a sua interpretação.

... e tenha um ótimo dia!!!

```
100  GR : HOME
110  INPUT "DIGITE O SEU NOME:  ";N$
120  INPUT "DIGITE A DATA DE HOJE:  ";D$
130 N$ = N$ + D$:S = 0
140  FOR I = 1 TO  LEN (N$):S = S +  ASC ( MID$ (N$,I)): NEXT I
150 S = S / 16:S = (S -  INT (S)) * 16
160 B = S / 8
170  FOR I = 0 TO 3
180 A(I) = B > .99:B = (B - (B > .99)) * 2
190  NEXT I
200  FOR I = 0 TO 3:V = I * 2
210  COLOR= 15: PLOT 18,V + 10: PLOT 20,V + 10
220  IF  NOT (A(I)) THEN  COLOR= 1
230  PLOT 19,V + 10
240  IF I < 3 THEN  COLOR= 1: FOR H = 18 TO 20: PLOT H,V + 11: NEXT H
250  NEXT I
260  HOME : PRINT  TAB( 5);"ESTE E' O SEU SIMBOLO DE HOJE": PRINT : GOTO 110
```

Ø  sucesso pela obediência

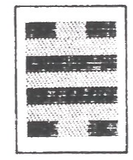
1  cuidado – problemas

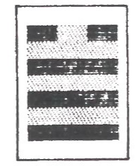
2  o trabalho leva à experiência

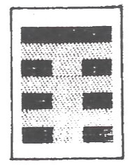
3  sucesso temporário

4  aja com firmeza

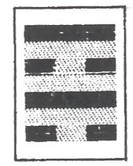
5  comporte-se bem

6  avance com razão

7  enriquecimento

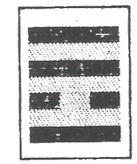
8  ação certa traz felicidad

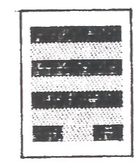
9  sucesso e felicidade

10  chegou o momento de agir

11  reconheça os erros

12  sua segurança inspira respeito

13  aja com bom senso

14  persistência recompensa

15  hora de avançar

# Introdução

Há quatorze anos atrás, o mundo inteiro via, maravilhado, o primeiro homem pondo os pés na Lua. Era o filho de Adão que, cortando o seu cordão umbilical, deixava a Terra-mãe e conquistava novos mundos.

Este feito notável só se tornou realidade graças ao avanço colossal que o homem, com sua imensa capacidade, soube dar à tecnologia da computação e da informática.

Um outro enorme palco das realizações humanas estava aberto.

Agora, e principalmente num futuro bem próximo, não poderemos prescindir da valiosa colaboração que a Informática nos dará e que já está nos proporcionando.

Não há dúvida que a segunda metade do século XX é a verdadeira era da Informática. Este livro, desenvolvido sobre o computador pessoal de maior sucesso no mundo todo, a famosa maçã, tem por finalidade, dentre outras, ser um suporte para referências para computadores da família Apple* e aumentar, na medida do possível, seu conhecimento nesse campo maravilhoso e fantástico do saber humano, usando uma abordagem extremamente didática do Applesoft** – o BASIC do Apple.

Fluxogramas, programas, conversas interativas entre texto-computador-leitor são usados em todos os capítulos do livro para fixar e desenvolver a teoria apresentada de uma forma clara, concisa e prática. Para tanto, o leitor é solicitado a participar, respondendo perguntas, fazendo e analisando programas, resolvendo exercícios. Por isso, recomendamos que o leitor acompanhe os capítulos praticando em um computador, de preferência da linha Apple, como, por exemplo, MAXXI$^{+}$, Unitron$^{++}$, Micro Engenho$'$, Dactron$''$, $\mu$6502$^{-}$ etc., pois assim seu trabalho apresentará melhor rendimento.

Toda manifestação de sua parte, caro leitor, será por nós recebida como estímulo e incentivo ao nosso trabalho e terá sempre o nosso melhor reconhecimento.

Wilson José Tucci
Coordenador dos Departamentos de Projetos Especiais e de Computação da Escola Experimental Pueri Domus.

(*, **) Apple e Applesoft são marcas registradas da Apple Computer, Inc.
(+) MAXXI e Polysoft são marcas registradas da Polymax Sistemas e Periféricos S/A.
(++) Unitron é marca registrada da Unitron Eletrônica Ltda.
(′) Micro Engenho é marca registrada da Spectrum Equipamentos Eletrônicos.
(″) Dactron é marca registrada da Micronix.
(-) O $\mu$6502 é marca registrada da Appletrônica – Computadores e Sistemas Ltda.

# Breve histórico

*"Sempre me pareceu estranho que todos aqueles que estudam seriamente esta ciência acabam tomados de uma espécie de paixão pela mesma. Em verdade, o que proporciona o máximo prazer não é o conhecimento e sim a aprendizagem; não é a posse mas sim a aquisição; não é a presença mas o ato de atingir a meta."*

*Carl Friedrich Gauss*
*(1777-1855)*

# I. INTRODUÇÃO

A facilidade com que a tecnologia de hoje nos coloca frente a um teclado de computador permite que, praticamente, o nosso limite em informática seja a nossa imaginação.

Desde a mais antiga idade do pensar, o homem busca meios materiais para vencer o cansaço e a morosidade do cálculo matemático.

Dos dedos da mão (até hoje em uso!) às primeiras tábuas de cálculo, pequenas fendas em placas de argila (cerca de 3.000 anos A.C.), utilizadas pelos povos mesopotâmicos, passando pelo "swanpan" ou "soroban" (cerca de 900 D.C.), pequenas armações com fios fixos e contas deslizantes, usadas pelos povos da China, Japão e Ásia Central; pelos "Ossos" de **Napier** (1617), pequenas tiras individuais com tabelas numéricas que associadas processavam multiplicações e divisões; pelo "Calculador" de **Pascal** (1645), que processava adições e subtrações usando engrenagens acopladas; pela "Roda" de **Leibniz** (1673), pequena calculadora mecânica que já realizava as quatro operações, o homem descobre a manipulação e processamento numéricos.

Em 1801, **Joseph Marie Jacquard** desenvolve um dispositivo de fiação com uma cadeia de cartões de metal perfurados que passava diante das agulhas de tecelagem. Somente as agulhas que coincidiam com os furos desses cartões operavam, definindo, assim, a padronagem do tecido. Esta foi, historicamente, a primeira máquina programável, a primeira a processar automaticamente informações não numéricas.

**Charles Babbage**, professor de matemática da Universidade de Cambridge, em 1830 projetou uma calculadora mecânica e automática. Não foi possível construí-la, pois necessitava de peças demasiadamente precisas para a tecnologia da época, gerando problemas técnicos e específicos demasiadamente grandes, além de ter tido pouca ajuda financeira.

Em 1885, por **William Burroughs**, e em 1911, por **Jay Monroe**, foram introduzidas as primeiras calculadoras dirigidas para o comércio e estabelecimentos bancários produzidas em massa. À medida que a tecnologia avançava, iam surgindo máquinas mecânicas e eletromecânicas que, além de aumentar a velocidade operacional, já realizavam as quatro operações com grande precisão.

Somente após a Primeira Guerra Mundial, passou-se a se considerar como distintas as partes da *unidade aritmética* e da *unidade de controle*. A unidade aritmética manipula os dados de entrada, os números; a unidade de controle comunica à unidade aritmética como e quando deve manejá-los.

Conforme as máquinas foram ficando mais automatizadas, a unidade de controle foi se tornando, por seu lado, cada vez maior e mais importante, exigindo mais atenção e funções mais diversificadas do operador.



# II. ARMAZENAMENTO, CONTROLE E PROCESSAMENTO

Babbage desenvolveu não só um dispositivo que não pôde ser construído na época, como uma filosofia que só foi concretizada cerca de 120 anos depois.

Havia proposto para sua máquina a entrada de dados e a acumulação desses dados, para em seguida serem processados. Dividiu seu Calculador Diferencial em três partes, que denominou: armazenador, mecanismo operador e controle. Veja a figura Ø.1:

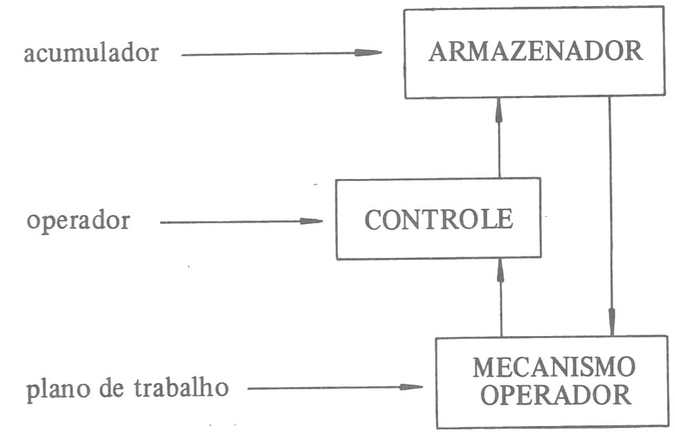



fig. Ø.1

Os dados no armazenador se distribuíam de forma ordenada, facilmente transferível para o mecanismo operador, se necessário. A unidade aritmética estava contida no mecanismo operador. Ao terminar um cálculo, a unidade de controle levaria os dados ao armazenador.

Babbage demonstrou que não adiantava armazenar dados isoladamente ou dispor rapidamente do dado; a máquina deveria saber também quando somar, subtrair, multiplicar ou dividir, e para tal projetou dois armazenadores separados: um para armazenar instruções e outro para armazenar dados. Planejou ter todas as instruções necessárias para uma série de operações, preparadas anteriormente e colocadas no armazenador de instrução na ordem de uso. Quando o mecanismo operador finalizava uma operação, devia solicitar a seguinte ao armazenador de instrução.

Em um computador, as instruções e os dados são armazenados na mesma unidade. Veja a figura Ø.2:

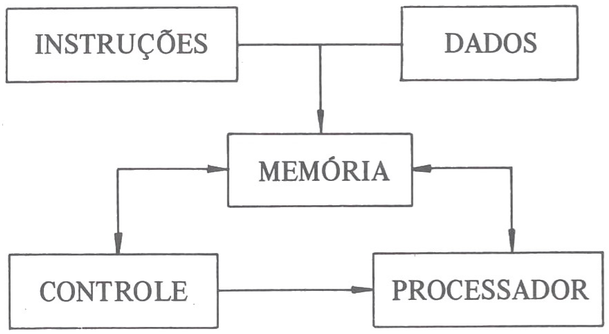



fig. Ø.2

Com o avanço da tecnologia os modernos computadores digitais apresentam a estrutura mostrada pela figura Ø.3:

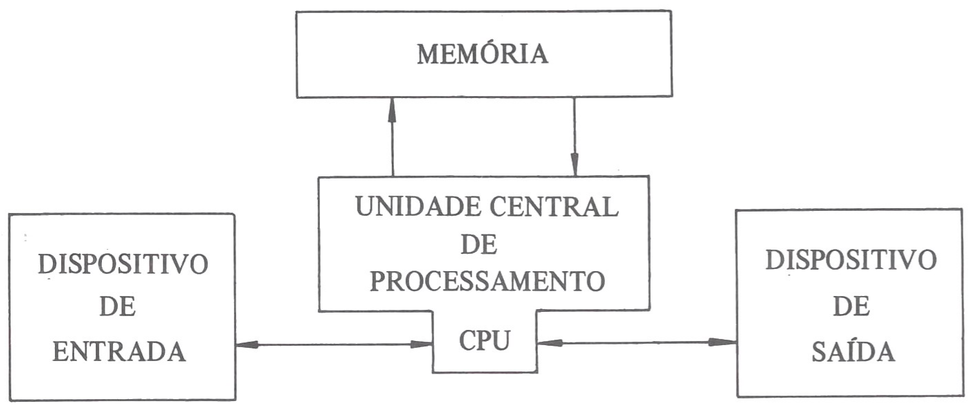



fig. Ø.3

As unidades de entrada e saída possibilitam a comunicação do meio externo com o computador, e vice-versa, do computador com o meio exterior. O dispositivo de entrada alimenta a memória com os dados e as instruções, os programas; as unidades de saída registram, após serem os dados processados, os resultados.

Como o próprio nome diz, as memórias armazenam todos os dados e o programa que processará os dados residentes e/ou fornecidos pelo usuário.

A unidade central de processamento reúne, num só bloco, as unidades de processamento e de controle. Aqui estão todos os circuitos que definem as operações lógicas e aritméticas e que coordenam as ações de todas as funções das outras unidades, como um sistema integrado e de forma totalmente automática.



# III. A IDADE DA INFORMÁTICA

As necessidades de métodos de processamento de dados mais rápidos e de menor custo de parte da ciência e do comércio exigiram máquinas eletromecânicas que acabaram por gerar os modernos computadores digitais.

O computador, por sua vez, tem contribuído com uma diversidade de tributos para melhorar nossa sociedade, fazendo até com que a segunda metade do século XX fosse denominada A Idade da Informação, ou da Informática.

Para se somar dois números de 5 dígitos cada um, demorava-se cerca de 10s, usando desde a técnica tradicional até aos dispositivos desenvolvidos por volta de 1930. A partir dessa data, a velocidade de processamento aumentou tanto que os nossos computadores já executam, usando os dois números de 5 dígitos, 10 000 000 de somas por segundo.

Podemos dividir a Idade da Informática em quatro eras:

a) Era Inicial (1940-1955)
b) Era do Crescimento (1955-1964)
c) Era do Refinamento (1964-1983)
d) Era da Maturação (1983-2000)

## III.1 – A Era Inicial (1940-1955)

Muitos progressos técnicos importantes, que contribuíram para o crescimento e as capacidades de sistemas de processamento de dados, aconteceram durante a Era Inicial.

A Era Inicial começou com o uso da tecnologia existente: relês, fita e cartões perfurados. Os tubos de vácuo, milhares de vezes mais rápidos, vieram para substituir os relês. Foram desenvolvidos sistemas de memória principal com o uso de núcleos magnéticos, e fitas e tambores magnéticos foram introduzidos como memória secundária.

O conceito de programa armazenado na memória foi implementado; o *software*, porém, era muito primitivo. A comunicação com a máquina era muito trabalhosa, já que todos os números, letras e símbolos, além dos próprios programas, tinham que ser introduzidos como grupos de ums e zeros, num código binário. Sistemas operacionais praticamente eram inexistentes. Por isso, o operador-programador tinha que estar envolvido em cada passo do processo computacional, dificultando muito a comunicação com usuários não-técnicos do computador.

Até o fim da Era Inicial, a utilização de computadores começou a se expandir rapidamente, com o ingresso de grandes corporações na manufatura destes. Entre

essas corporações estão Burroughs, Control Data, General Electric, Honeywell, IBM, National Cash Register, RCA e Univac.

## III.2 – A Era do Crescimento (1955-1964)

A Era do Crescimento foi marcada por importantes desenvolvimentos na área de *hardware* e *software,* ficando ainda a dificuldade de criar computadores que viessem de encontro à grande variedade de necessidades dos diferentes tipos de usuários, bem como as deficiências na área de comunicações.

Entre os desenvolvimentos mais importantes em *hardware,* temos o uso do transistor, o melhoramento das memórias de núcleo magnético e dos sistemas de fita magnética, a introdução do disco magnético e aparelhos periféricos como dispositivos de memória, leitoras de cartões e impressoras.

Houve também um progresso significativo em *software,* especialmente no que se refere às **linguagens simbólicas**, praticamente eliminando a necessidade de se lidar com os códigos binários. Estas linguagens simbólicas inicialmente eram de baixo nível, isto é, o programador podia substituir combinações de ums e zeros por palavras mnemônicas de fácil memorização; até o final da Era do Crescimento, porém, surgiram várias linguagens de alto nível, utilizando comandos bastante semelhantes ao inglês de uso corrente. Foram introduzidos também os sistemas operacionais, liberando os programadores da necessidade de executar tarefas repetitivas, como a movimentação de dados entre o computador e seus aparelhos periféricos. Faltava, ainda, o desenvolvimento da chamada "arquitetura do computador", que é a combinação perfeita de *hardware* e *software* para formar um sistema de computação funcional, do ponto de vista do usuário. Um sistema pode ser tecnicamente sólido mas pode ter uma arquitetura pouco eficaz para corresponder às necessidades atuais ou futuras de seus usuários, de forma eficiente. A comunicação entre o usuário e o programador, bem como a própria educação desse usuário, era muito limitada, dificultando o desenvolvimento de sistemas.

## III.3 – A Era do Refinamento (1964-1983)

Esta nova era começou com a dominância de grandes computadores centrais, terminando com uma "revolução dentro da revolução dos computadores". Esta revolução interna, baseada na habilidade da indústria na miniaturização de computadores, foi a distribuição do poder da computação a todos os lugares onde era necessário. Os principais refinamentos na área do *hardware* foram:



**Unidade central de processamento**: os transistores, que haviam tomado o lugar dos tubos de vácuo, foram, por sua vez, substituídos por uma tecnologia de circuitos integrados (CIs), pela qual milhares de transistores e de outros componentes eletrônicos são formados sobre uma pequena pastilha de silício, os *chips,* sendo que o número de circuitos que podiam ser formados num desses *chips* aumentou continuamente durante a Era do Refinamento.

**Canais**: pequenos computadores internos, que recebiam este nome, foram incorporados na UCP para melhorar principalmente a velocidade das operações de entrada/saída.

**Memória**: foram desenvolvidos maiores e melhores sistemas de armazenamento por núcleos magnéticos, sendo suplementados ou substituídos por circuitos integrados; no que se refere a armazenamento secundário, foram desenvolvidos sistemas melhores e mais rápidos de fitas e discos magnéticos.

Com o baixo custo relativo dos sistemas de memória, os sistemas operacionais ficaram mais sofisticados, tornando o trabalho do programador muito mais fácil.

Combinações de *hardware* e *software* tornaram possível a execução de dois ou mais programas ao mesmo tempo pelo computador. Com os desenvolvimentos em sistemas de *hardware* e *software* orientados para as comunicações, surge o processamento à distância, em que um computador central controla vários terminais ligados a ele.

Dos terminais inteligentes evoluíram os **minicomputadores**, capazes de funcionar tanto como um terminal local, ligado a um processador central, como serem usados como um computador independente. Com a miniaturização, surgem as UCPs completas num *chip*, que recebem o nome de **microprocessadores**. Combinando um microprocessador com circuitos miniaturizados de memória e mais alguns componentes, temos o **microcomputador**.

Surge um novo profissional, o **analista de sistemas**, como resultado de esforços feitos para formalizar os procedimentos no desenvolvimento de projetos aplicativos. Esse profissional foi preparado por educação e experiência para analisar problemas e, quando necessário, desenvolver soluções, e veio preencher o grande espaço que existia, até então, entre o programador e o usuário.

## III.4 – A Era da Maturação (1983-2000)

Vamos descrever apenas alguns dos pontos altos previsíveis para a Era da Maturação:

A relação *custo/performance* continuará a diminuir, com a miniaturização cada vez maior das UCPs e unidades de memória, fazendo com que o microprocessador passe a ser utilizado em sistemas grandes e pequenos.

Serão (e já estão sendo) construídos os chamados "supercomputadores", e colocados no mercado computadores completos (UCP, memória e entrada/saída), contidos num único *chip*.

O computador começa a entrar em todas as áreas da vida cotidiana: automóveis, eletrodomésticos, jogos, medicina, compras, aplicações bancárias e em escritórios.

Novos aparelhos de entrada/saída serão desenvolvidos continuamente para reduzir os custos de sistemas miniaturizados.

Os sistemas operacionais e linguagens de programação para os microcomputadores serão melhorados, chegando ao nível em que se encontram aqueles implementados nos computadores maiores.

As redes de comunicação entre pequenos computadores crescerão, com a produção de novos sistemas de *software* dirigidos a essa área.

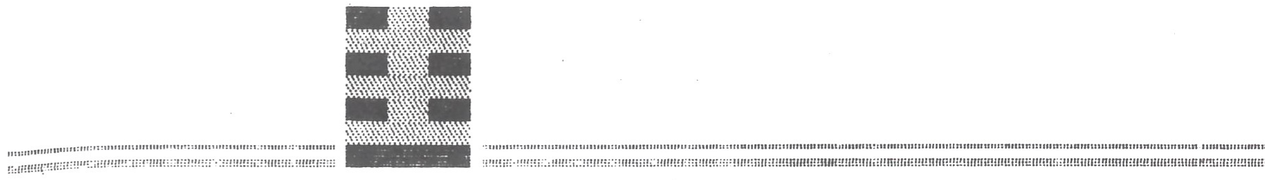
Seguramente, o maior desafio da Era da Maturação será na parte da melhora das técnicas de criação de *software,* desde o início de um projeto até a utilização do sistema, na sua forma final.

## IV. RAXAKUKA

0.1 Seu relógio adianta 4 minutos em 24 horas. Se ele marca 7 horas e 30 minutos e meio às 7:30 h da manhã, quanto estará adiantado às 12 horas do mesmo dia?

0.2 Um rapaz gastou metade do seu dinheiro no lanche e metade no cinema e ficou com Cr$ 400,00. Quanto gastou no cinema?

0.3 Numa corrida de cavalos, o vencedor cruzou o disco de chegada às 3 horas e um minuto da tarde, com 4 corpos de vantagem sobre o terceiro colocado. Este chegou com 2 corpos de desvantagem sobre o segundo. O segundo cruzou o disco com 4 1/2 corpos de vantagem sobre o quarto colocado, que fez a corrida em 61 segundos e 3/10. Na última quarta parte da corrida, cada cavalo avança um corpo em 1/5 de segundo. A que horas teve início a corrida?



# Conhecendo o computador

*"Eu me sentia satisfeito sobretudo nas matemáticas, por causa da certeza e evidência de suas razões; mas não compreendia seu verdadeiro uso e, pensando que não serviam senão às artes mecânicas, eu me espantava de que, sendo seus fundamentos tão firmes e sólidos, nunca tivessem conduzido a algo mais elevado."*

*René Descartes*
*(1596-1650)*

# I. INTRODUÇÃO

Até há pouco tempo, os computadores estavam restritos às áreas industrial, financeira, universitária e militar, ocupando e exigindo instalações espaçosas e dispendiosas. Com o avanço, cada vez mais acelerado, da indústria eletrônica, o consumo das "comodities" está mais acessível e rápido.

Talvez a característica mais marcante de um computador ou cérebro eletrônico seja não ter inteligência inata. Ele não compreende nenhuma linguagem humana – sua linguagem interna é denominada *linguagem de máquina*; contudo o computador pode operacionalizar instruções dadas em uma linguagem que por ele possa ser compreendida, traduzindo-as para a sua linguagem de máquina. A linguagem que usaremos no nosso computador é o **BASIC**.

**BASIC**, palavra formada pelas iniciais de **B***eginner's* **A***ll-purpose* **S***ymbolic* **I***nstruction* **C***ode,* foi criada em 1964 por **Kemeny** e **Kurtz**, no Dartmouth College. Essa linguagem, por ser baseada em termos comuns de sintaxe de uso corrente, é muito fácil de se usar e aprender, sendo implementada na grande maioria dos microcomputadores atuais.

# II. ANTES DE COMEÇAR

Se seu computador ainda não estiver montado, siga as instruções do manual do usuário para ligar à unidade central um monitor de vídeo ou TV, e à unidade de disco e/ou um gravador cassete. Você poderá ter mais periféricos ligados ao computador, como um *modem* ou impressora, mas seu uso não será abordado neste livro.

O texto foi elaborado com base num **Apple II Plus**, com **48K** e **uma unidade de disco**.

Se você estiver usando *cassete* em lugar do *disk drive*, consulte o seu manual nas seções que se referem à sua utilização. Tendo menos de 48K, você poderá seguir o texto normalmente, com exceção do que se refere a gráficos de alta resolução, abordados no capítulo 13.

# III. FAMILIARIZAÇÃO COM O COMPUTADOR

A configuração do computador que vamos utilizar em nossos primeiros contatos consiste em um *monitor de vídeo* e um *processador,* ao qual está acoplado um *teclado.*



Também está acoplada a esse sistema uma unidade de disco (*disk drive*). A unidade de disco está ligada por um cabo chato e largo à unidade de processamento. A função do *drive* é guardar, em discos magnéticos, informações da unidade de processamento e ler antigas informações previamente armazenadas nesses discos, enviando-as para a unidade de processamento.

Essas informações podem ser arquivos, programas feitos pelo usuário ou *software* comercial (programa pronto). Para podermos operacionalizá-las, os programas e dados devem estar na memória eletrônica do processador; mas essa memória é volátil, isto é, ela se apaga ao desligarmos a alimentação do computador. Por isso, geralmente *gravamos* todo o trabalho feito, antes de desligarmos o computador. O disco é, por assim dizer, uma *memória permanente,* enquanto a memória do processador é uma *memória de trabalho.*

Nosso computador é do tipo que, ao ser ligado, inicia a leitura do disco, no *drive.* Se o disco lá estiver, ele pára logo; caso contrário, o computador ficará procurando indefinidamente, até que seja pressionada a tecla *RESET.*

O teclado é utilizado para digitarmos instruções, informações, dados e comandos ao computador. As informações digitadas, processadas ou ainda eventuais mensagens do computador são visualizadas no monitor de vídeo.

Analise o teclado (figura 1.1) do computador; ele é bem semelhante ao teclado de uma máquina comum de escrever. Devemos perceber que não existem letras minúsculas (*caixa baixa-lower case*).

Algumas teclas, como Ñ, Ï, { 8 etc., apresentam dois símbolos, isto é, valem por duas. Se quisermos digitar 8, simplesmente pressionamos a tecla { 8; porém, para abrirmos um parêntese, devemos *pressionar e manter pressionada* a tecla *SHIFT* e simultaneamente pressionamos a tecla { 8.

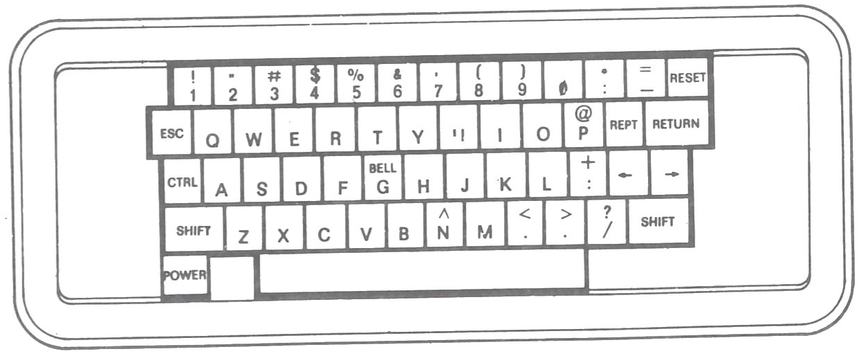



fig. 1.1

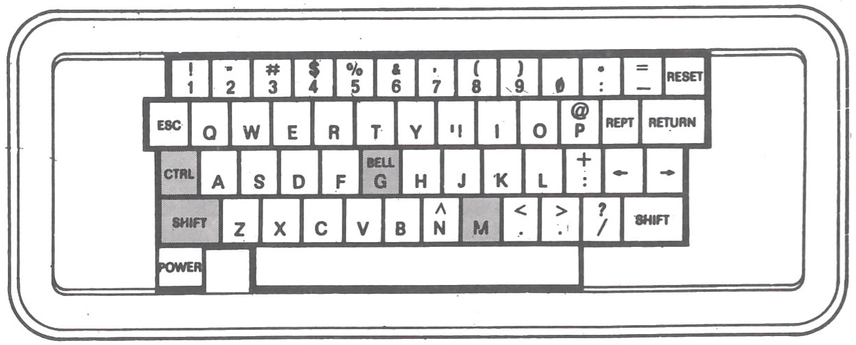



fig. 1.2

Existem, no entanto, duas exceções: as teclas M e G (figura 1.2).

Quando deslocada (*shifted*), a tecla M apresenta o colchete voltado para a esquerda ]. A tecla G apresenta a palavra *bell* escrita em sua parte superior e, quando pressionada junto com a tecla *CTRL* (da mesma forma que *SHIFT* com M), produz um *bip* no alto-falante do computador.

Outra importante diferença, entre o teclado do nosso computador e o de uma máquina de escrever, é a existência do símbolo numérico "1", que será usado para representar a unidade e não, como comumente é feito, usando-se a caixa baixa da letra "L" (figura 1.3).

Deve ser notada, também, a existência do símbolo "Ø" para representar o numeral zero e não, como é usado em máquinas comuns, a letra maiúscula O. A barra inclinada, cortando os símbolos, foi colocada justamente para diferenciar os "zeros" dos "os" (figura 1.3).

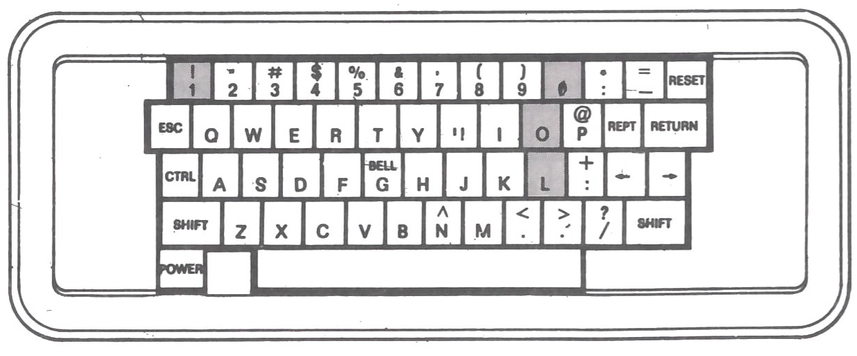



fig. 1.3



## IV. LIGANDO O COMPUTADOR

Para ligarmos o computador devemos sempre seguir a seqüência abaixo.

1. Verificar se há disco no *disk drive* e fechá-lo, havendo ou não. Inicialmente usaremos o computador sem disco.

2. Ligar o monitor de vídeo ou aparelho de TV conforme as instruções do próprio manual.

3. Ligar o computador, acionando o interruptor localizado na sua parte posterior esquerda (figura 1.4). A lâmpada piloto do teclado deverá acender (figura 1.5) com um *bip* do computador e o acionamento do *drive*.

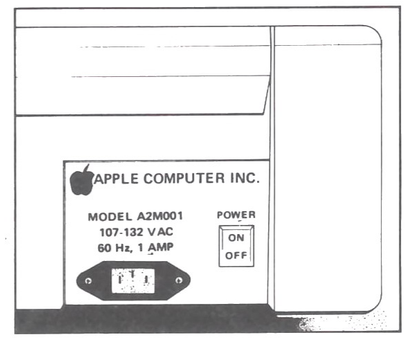



fig. 1.4

4. *SE NÃO HOUVER DISCO NO* DRIVE, pressione a tecla *RESET**; caso contrário o disco parará automaticamente, após alguns segundos.

5. Em ambos os casos, aparecerá um colchete ] (figura 1.6) acompanhado de um quadrado, que ficará "pisca-piscando", denominado *CURSOR*.

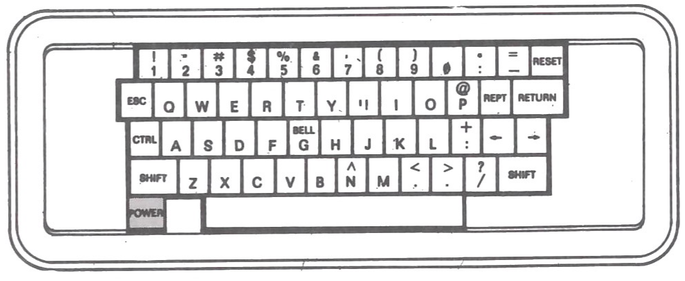



fig. 1.5

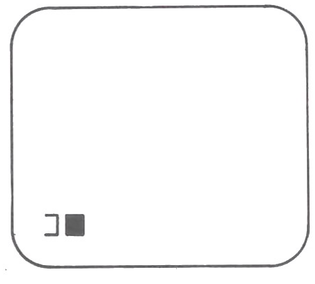


fig. 1.6

(*) Em alguns modelos, a tecla *RESET* não funciona sozinha, devendo ser acionada junto com a tecla *CTRL*.

## V. INICIANDO O BASIC

Para que possamos ordenar ou impor instruções ou comandos a serem executados pelo computador, é necessário que esses comandos ou instruções sejam dados em uma linguagem por ele inteligível.

*Linguagem* é o conjunto de termos e sinais preestabelecidos que permite a interação e comunicação do usuário com a máquina, sempre seguindo regras bem definidas que variam de uma linguagem para outra.

### V.1 – O Comando PRINT

A visualização do comando **PRINT** é muito simples. Se o computador não estiver ligado, veja como fazê-lo na unidade IV.

Digite:

```
OI PESSOAL
```

e, em seguida, pressione a tecla *RETURN*. O que apareceu na tela?

**R-1**

OBS.: Os espaços em branco, indicados por **R-n**, são reservados para a sua resposta.

A tecla *RETURN* serve para indicar ao computador que execute a mensagem impressa na tela.

A mensagem

```
?SYNTAX ERROR
```

não significa que você danificou o computador; simplesmente ele não reconheceu a mensagem digitada, isto é, em seu vocabulário *BASIC* não existe tal instrução.

OBS.: Sempre que aparecer o colchete ] e o cursor ■ o computador está apto a receber novas instruções através do teclado.



Digite, agora:

```
PRINT "OI PESSOAL"
```

não esquecendo de pressionar a tecla *RETURN*. O que aparece na tela?

**R-2**

Digite, a seguir:

```
PRINT OI PESSOAL
```

Pressione *RETURN*; qual o resultado?

**R-3**

O comando **PRINT** diz ao computador para visualizar no vídeo todos os caracteres entre aspas. A quantidade máxima de caracteres entre as aspas na instrução **PRINT** é de cerca de 25Ø.

Tente essas instruções, não esquecendo de pressionar a tecla *RETURN*, ao final.

```
PRINT "182"
```

**R-4**

```
PRINT 182
```

**R-5**

Aparentemente, não houve distinção entre as duas situações acima; no entanto, a primeira processa uma informação alfanumérica e a segunda processa uma informação numérica.

Faça, não esquecendo de pressionar *RETURN* ao final de cada instrução:

```
PRINT "2 + 5"
```

**R-6**

```
PRINT 2 + 5
```

**R-7**

Qual a diferença observada?

**R-8**



## V.2 – Corrigindo Enganos, Antes do RETURN

Você já deve ter cometido enganos ao digitar determinados textos, comandos ou instruções, e o computador, através das mensagens, detectou esses enganos.

O *APPLE* apresenta alguns recursos para correção de erros, utilizando as teclas de *backspace* e *retype* (figura 1.7).

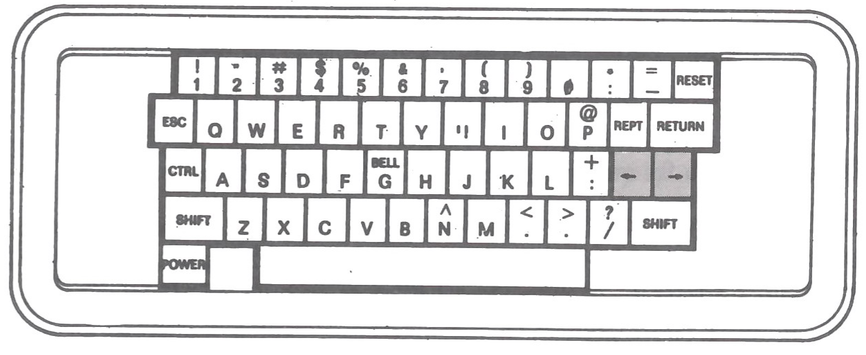



fig. 1.7

A tecla [←] é denominada *backspace key* e é utilizada para reposicionar o cursor no caractere errado, como fazemos com a tecla de correção numa máquina de escrever comum. Digite o comando abaixo:

```
PRINT APPLE"
```

**R-9**

O computador respondeu com o número zero, já que faltavam aspas no início da palavra.

Digite agora, *sem pressionar RETURN:*

```
PRINT "APLLE"
```

Nada acontece, pois o computador ainda não recebeu o *RETURN.* O cursor, que está parado logo após as aspas finais, deverá ser posicionado *sobre* a primeira letra *L,*

que será trocada por um *P* para corrigir o erro. Faça isso, pressionando a tecla de correção [←] quatro vezes. Aperte agora a letra *P*, substituindo automaticamente o *L* incorreto, e *RETURN*. Qual a saída?

**R-1Ø**

O que aconteceu foi que, cada vez que voltamos o cursor com a tecla [←] o computador *esquece* o caractere que ficou para trás. Em outras palavras, o computador só *vê* as letras que estão à esquerda do cursor. Uma solução seria redigitar, logo após a correção da letra *L*, os caracteres LE" e pressionar *RETURN*. Faça-o; qual o resultado?

**R-11**

Há, no entanto, um método mais simples para fazer essa correção, usando a tecla [→].

Digite, novamente (sem o *RETURN*):

```
PRINT "APLLE".
```

Como fizemos antes, pressione [←] quatro vezes, seguido pelo P. Aperte agora a tecla [→] três vezes, levando o cursor até o final do comando, depois das aspas finais, e aperte *RETURN*.

**R-12**



As duas teclas de correção são muito úteis, permitindo-nos corrigir pequenos enganos sem ter que redigitar a linha inteira.

## V.3 – O Computador Como Calculadora

Por meio do comando **PRINT**, podemos usar o *APPLE* como uma calculadora; ele responde diretamente aos cálculos propostos.

Tente esses exemplos:

```
PRINT 4 + 6
```

**R-13**

```
PRINT 5ØØ + 85Ø + 38
```

**R-14**

```
PRINT 6ØØ - 374
```

**R-15**

```
PRINT 100 * 23
```

**R-16**

```
PRINT 1 / 2
```

**R-17**

```
PRINT 4 * 4 * 4
```

**R-18**

```
PRINT 4 ^ 3
```

**R-19**

```
PRINT 3 * 4 * 100 - 800
```

**R-20**



```
PRINT 3 * 4 * (100 - 800)
```

**R-21**

```
PRINT 9 ^ .5
```

**R-22**

```
PRINT 9 ^ 1 / 2
```

**R-23**

```
PRINT 9 ^ (1 / 2)
```

**R-24**

Você deve ter percebido que os símbolos usados nas expressões aritméticas são diferentes dos usuais:

Tabela 1

| OPERAÇÃO | SÍMBOLO NO APPLESOFT |
|---|---|
| soma | + |
| subtração | - |
| multiplicação | * |
| divisão | / |
| potenciação | ^ |

Também pelos exemplos acima vemos que o computador executa as operações seguindo uma certa ordem predefinida.

A hierarquia que existe entre os operadores aritméticos resume-se na tabela abaixo. Os operadores estão em ordem de execução, na vertical, da maior prioridade (parênteses) até a menor (soma e subtração). Operadores colocados na mesma linha têm a mesma prioridade, sendo executados na ordem em que se encontram no **PRINT**, da esquerda para a direita.

Tabela 2

| | | |
|---|---|---|
| 1º | ( ) | |
| 2º | + - | quando usados como sinal |
| 3º | ^ | |
| 4º | * / | |
| 5º | + - | como soma e subtração |

Como exercício, complete a tabela 3 com o resultado previsto (calculado à mão) e a resposta do computador:



Tabela 3

| EXPRESSÃO | VALOR PREVISTO | RESPOSTA DO APPLE |
|---|---|---|
| 6 / 4 - 8 * 1 | | |
| 5 - 4 / 2 | | |
| 6 * - 2 + 6 / 3 + 8 | | |
| 4 + - 2 * 2 ^ 3 | | |
| 2 ^ (2 ^ 2) + 1 | | |
| 2 * 2 * 2 + 1 | | |
| 4 + 5 ^ 1 / 2 | | |
| 2 * 2 + 2 * 1 | | |
| 8 / (2 / 2 / 1) | | |
| 6 * 2 / 2 + (3 * 2) ^ 2 * 3 | | |
| 3 + 8 / 24 + 32 | | |
| 3 + 8 / (24 + 32) | | |
| 1ØØ / (2ØØ / (1 * (9 - 5))) | | |
| 32 / (1 + (7 / 3) + (5 - 4 / 3)) | | |

Escreva o comando que calcula cada uma das expressões abaixo:

$\frac{25 + 48}{3.18} =$

**R-25**

```
PRINT (25 + 48) / 3.18
```

$\frac{45 - 18}{7.2 + 1.9} + 162 =$

**R-26**

$$\frac{56 - 95}{2.3 \times 4} =$$

**R-27**

$$\frac{87 + 8.7}{5.9 : 3} + \frac{65 \times 4.2}{35 - 1\emptyset 4} =$$

**R-28**

$$\left(\frac{97 - 1\emptyset 9}{53 - 42}\right)^3 =$$

**R-29**

$$\left(\frac{27 - 89}{54 - 3\emptyset}\right)^2 - \sqrt{\frac{73 - 25}{1 . 73}} = \text{(lembrando que } \sqrt{x} = x^{\frac{1}{2}}\text{)}$$

**R-3Ø**

OBS.: Na dúvida, utilize parênteses.



## V.4 – Formato Padrão dos Números

Digite:

```
PRINT 45.34Ø
```

O seu computador deverá responder:

**R-31**

```
 45.34
```

não imprimindo os zeros, à direita do número.

Agora digite:

```
PRINT .Ø5ØØØ
```

**R-32**

```
PRINT 1ØØØ.ØØ
```

**R-33**

```
PRINT Ø58.Ø
```

**R-34**

O computador também não imprime os zeros que antecedem um número.
Para números muito pequenos, ocorre uma mudança automática de formato.
Faça:

```
PRINT .005
```

**R-35**

```
PRINT .00091
```

**R-36**

```
PRINT .08 / 10.2
```

**R-37**

```
PRINT 0.0329
```

**R-38**

```
PRINT 0.00329
```

**R-39**



```
PRINT 0.000329
```

**R-40**

```
PRINT 0.000000000329
```

**R-41**

Este novo formato recebe o nome de *notação* científica e também é usado na representação de números muito grandes. Faça:

```
PRINT 259 ^ 8.3
```

**R-42**

```
PRINT 3290000000
```

**R-43**

```
PRINT 32900000000
```

**R-44**

```
PRINT 3290000000000000000
```

**R-45**

```
PRINT 8.156 * 10 ^ 23
```

**R-46**

```
PRINT -2 * 10 ^ 15
```

**R-47**

A notação

**m.mm E + nn**

representa o número

$m.mm \cdot 10^{nn}$

O número de Avogadro ($6.02 \cdot 10^{23}$) é representado, na notação do Apple, como *6.02E + 23*; portanto, os dois comandos abaixo fornecem o mesmo resultado:

```
PRINT 6.02 * 10 ^ 23
```

**R-48**



```
PRINT 6.02E+23
```

**R-49**

Vamos, a seguir, analisar o "arredondamento":

```
PRINT 657438.6489
```

**R-50**

```
PRINT 12345678.631
```

**R-51**

```
PRINT 10045.678925
```

**R-52**

Nas três instruções anteriores, observamos um arredondamento, utilizando o critério de aproximação, isto é, se o primeiro dígito desprezado for maior ou igual a cinco, acresce-se uma unidade ao dígito anterior, somente se o número em questão apresentar até nove dígitos significativos.

Experimente:

```
PRINT 1234567890
```

O computador deverá responder:

```
1.23456789E+Ø9
```

que é o mesmo número apresentado sob forma de notação científica. A mudança de formato ocorreu pelo fato do número não poder ser representado com nove casas decimais.

Tente agora:

```
PRINT 1.Ø123456
```

**R-53**

```
PRINT 1.Ø1234567
```

**R-54**

```
PRINT 1.Ø12345678
```

**R-55**

```
PRINT 1.Ø123456789
```

**R-56**



```
PRINT 999.8765434567
```

**R-57**

## VI. EXERCÍCIOS PROPOSTOS

1.1 Expanda a frase da capa para a informática atual. Como você a interpreta?

1.2 Qual (ou quais) a(s) diferença(s) entre informações numéricas e alfanuméricas?

1.3 Para visualizar uma informação alfanumérica, como devemos proceder?

1.4 Mostre como você instruiria o computador para executar a operação 4 + 3 - 1.

1.5 Nos exemplos das páginas 23 e 25 você verificou que, ao usarmos parênteses, o resultado é modificado. Como você explica esse fato?

1.6 Escreva a instrução correta para o cálculo de:

a) $\frac{5}{4} + 2$

b) $\frac{3 * 4}{8 - 3}$

c) $\frac{5 + 7}{3 * 8} + 2^3$

d) $\frac{2^2}{3 - 8} + \sqrt{\frac{87 - 3}{9 * 5}}$

## VII. RAXAKUKA

1.1 Qual o sinal matemático que devemos colocar entre 2 e 3 para formar um número maior do que 2 e menor do que 3?

1.2 O casal Silva tem vários filhos. Cada filha tem o mesmo número de irmãos e irmãs, e cada filho tem duas vezes mais irmãs que irmãos. Quantos filhos e filhas o casal tem?

1.3 Um tijolo *pesa* um quilo mais meio tijolo. Quanto *pesa* um tijolo e meio?

# Armazenando informações

*"Não tiro conclusões de uma só experiência, nem são inúteis as repetições que faço; também não desvirtuo nenhum dado experimental para que se enquadre em noções pré-concebidas; ao contrário, empenho-me em ser versado nas diversas técnicas; de todos e quaisquer desses ensaios observo suas normas e critérios e seus meios de verificação, para através deles poder examinar minhas noções prévias."*

*Robert Hooke*
*(1635-1703)*

# I. INTRODUÇÃO

Analogamente ao termo matemático, mas com um significado um pouco diferente, *variável* define uma posição de memória, no computador, onde podemos armazenar informações. Quando recebem informações quantitativas (números) são designadas por *variáveis numéricas* e, se receberem cadeias de caracteres (*string*), *variáveis alfanuméricas.*

O *nome* com que representamos a variável fica sendo o nome dessa memória, não tendo nenhuma dependência com o valor armazenado nesse lugar. Esses nomes não podem ser quaisquer; devem obedecer a determinadas regras, como veremos adiante.

# II. VARIÁVEIS NUMÉRICAS

Em BASIC, as variáveis numéricas são semelhantes às variáveis matemáticas; elas guardam um determinado valor.

Digite:

```
LET M=12Ø
```

Observamos que o valor 12Ø não é visualizado no vídeo; ele foi armazenado na memória designada por M. Digite, agora:

```
PRINT M
```

**R-1**

e o computador colocará no vídeo o valor **12Ø**, memorizado anteriormente por meio da instrução **LET**, isto é, a instrução através da qual atribuímos às variáveis os valores desejados.

No BASIC utilizado pelo nosso computador, a presença de instrução **LET** é opcional, isto é, escrever **LET M = 12Ø** ou **M = 12Ø** é interpretado do mesmo modo pelo computador.

Faça, a seguir,



```
LET M=321
```

e ordene ao computador para que ele imprima o conteúdo da variável **M.**

O computador visualizou o número **321.**

O que aconteceu com o número **12Ø**?

**R-2**

Esse número foi perdido, pois a memória especificada só armazena um dado de cada vez. Quando você coloca um novo valor na variável **M** o valor anterior é perdido.

Digite, a seguir, os comandos:

```
PRINT "M"
```

**R-3**

```
PRINT M
```

**R-4**

Qual a diferença entre as duas instruções acima?

**R-5**

Na primeira instrução, o computador imprimiu o conteúdo da cadeia *string* entre as aspas e, na segunda, imprimiu o conteúdo da variável numérica M.

Digite, agora:

```
HOME
```

O que aconteceu?

**R-6**

O comando **HOME** faz com que o computador limpe somente a tela de texto no vídeo, posicionando o cursor no canto superior esquerdo. **HOME** não altera o conteúdo das variáveis ou das memórias. Como você pode verificar este fato?

**R-7**

## II.1 – Armazenando e Processando Expressões

O computador pode armazenar diretamente o resultado de expressões matemáticas. Faça:

```
LET A=7+3
LET B=5*4

PRINT A
```

**R-8**



```
PRINT B
```

**R-9**

e pode, ainda, processar diretamente o valor atribuído anteriormente à variável:

```
PRINT A+5
```

**R-10**

```
PRINT A+B
```

**R-11**

```
PRINT A+B/2
```

**R-12**

```
PRINT (A+B)/2
```

**R-13**

```
LET C=A*B
PRINT C
```

**R-14**

```
LET D=(B*(5+C))/A
PRINT D
```

**R-15**

```
LET A=A+5
PRINT A
```

**R-16**

Até aqui foram definidas as variáveis **M, A, B, C** e **D**.
Digite:

```
CLEAR
```

Qual foi a resposta do computador?

**R-17**



Faça:

```
PRINT M
PRINT A
PRINT B
PRINT C
PRINT D
```

Quais as respostas a esses comandos?

**R-18**

O comando **CLEAR** *zera* todas as variáveis. Seus conteúdos são perdidos e Øs são nelas introduzidos, sem alterar a tela de texto no vídeo.

## II.2 – Particularidades do Applesoft

Além de conter o BASIC padrão, apresenta particularidades:

a) O comando **LET** é opcional. As intruções

```
LET B=3
```

e

```
B=3
```

apresentam o mesmo efeito.

b) O nosso computador apresenta 936 variáveis endereçáveis e é permitido o uso de mais de uma letra (até 238 caracteres); o primeiro caractere *deve ser sempre uma letra.* Exemplos:

| variáveis permitidas | variáveis não permitidas |
|---|---|
| **A** | **Ø** |
| **A B** | **ØØ** |
| **A 1** | **1 A** |
| **X X** | **D *** |
| **ALICE** | **H /** |
| **WALTER** | **! A** |
| **TESOURA** | **+ -** |

Apesar de ser permitido até 238 caracteres, o computador, após analisar o nome, só reconhece os dois primeiros caracteres. Digite:

```
JOSE = 10.5
SOMA = 45+9
PRINT JOSE
```

**R-19**

```
PRINT SOMA
```

**R-20**

Porém devemos tomar cuidado na escolha do nome ou designação da variável. Digite:

```
JOEL = 2.4
ABRICOT = 5.8
JOAO = 4
ABACAXI = 9
PRINT JOAO
```

**R-21**

```
PRINT ABACAXI
```

**R-22**



```
PRINT JOEL
```

**R-23**

```
PRINT ABRICOT
```

**R-24**

```
PRINT JOAQUIM
```

**R-25**

```
PRINT ABOBRINHA
```

**R-26**

Apesar de não termos definido as variáveis **JOAQUIM** e **ABOBRINHA**, o computador imprimiu números atribuídos, não sabemos por quem, a essas variáveis. E o que dizer de **JOEL** e **ABRICOT**?

**R-27**

Na realidade, as supostas seis variáveis não são seis, mas sim somente *duas variáveis*. Se você estudar as designações, perceberá que as letras **JO** e **AB** são comuns; logo, as variáveis definidas são **AB** e **JO**. Tente:

```
PRINT JO
```

**R-28**

```
PRINT AB
```

**R-29**

Devemos tomar, ainda, um outro cuidado na escolha dos nomes das variáveis; tente:

```
ABOBORA = 10
```

O que o computador respondeu?

**R-30**

Tente agora:

```
VALE = 5000
```

**R-31**

```
NOTA = 4.0
```

**R-32**



```
DIALETO = 27
```

**R-33**

As respostas seguem no próximo item.

c) O BASIC do computador reservou para si algumas palavras que têm um significado específico a ele e não podem ser utilizadas. São denominadas *palavras reservadas.*

A relação primeira que se segue contém todas as palavras reservadas do BASIC usado pelo nosso computador. A segunda relação apresenta mais algumas palavras reservadas que se tornam, em certos casos, válidas somente quando utilizamos o DOS – Sistema Operacional de Disco.

Applesoft:

**&**
**ABS**
**AND**
**ASC**
**AT**
**ATN**

**CALL**
**CHR$**
**CLEAR**
**COLOR =**
**CONT**
**COS**

**DATA**
**DEF**
**DEL**
**DIM**
**DRAW**

**END**
**EXP**

**GET**
**GOSUB**
**GOTO**
**GR**

**HCOLOR =**
**HGR**
**HGR2**
**HIMEM:**
**HLIN**
**HOME**
**HPLOT**
**HTAB**

**IF**
**IN#**
**INPUT**
**INT**
**INVERSE**

**LEFT$**
**LEN**
**LET**

**NEW**
**NEXT**
**NORMAL**
**NOT**
**NOTRACE**

**ON**
**ONERR**
**OR**

**PDL**
**PEEK**
**PLOT**
**POKE**
**POP**
**POS**
**PRINT**
**PR#**

**READ**
**RECALL**
**REM**
**RESTORE**

**SAVE**
**SCALE =**
**SCRN (**
**SGN**
**SHLOAD**
**SIN**
**SPC (**
**SPEED =**
**SQR**
**STEP**
**STOP**
**STORE**
**STR$**

**TAB (**
**TAN**
**TEXT**
**THEN**
**TO**
**TRACE**

**USR**

| | | | |
|---|---|---|---|
| | LIST | RESUME | VAL |
| FLASH | LOAD | RETURN | VLIN |
| FN | LOG | RIGHT$ | VTAB |
| FOR | LOMEM: | RND | |
| FRE | | ROT = | WAIT |
| | MID$ | RUN | |
| | | | XPLOT |
| | | | XDRAW |

DOS:

| | | | |
|---|---|---|---|
| INIT | LOAD | DELETE | VERIFY |
| CATALOG | RUN | LOCK | MON |
| SAVE | RENAME | UNLOCK | NOMON |
| FP | INT | PR# | MAXFILES |
| OPEN | READ | APPEND | IN# |
| CLOSE | WRITE | POSITION | CHAIN |
| BLOAD | BRUN | EXEC | BSAVE |

Digite, a seguir:

```
LETRA = 140
PRINT LETRA
```

**R-34**

```
PRINT RA
```

**R-35**

Como você explica as diferentes respostas acima?

**R-36**



d) Todas as variáveis numéricas vistas até agora armazenavam números reais. São também chamadas de *variáveis reais*. Temos ainda um outro tipo de variável numérica, a *variável inteira*. Essa variável é apta somente para armazenar números inteiros entre -32767 e +32767; para diferenciá-la da variável real, além das regras de nomenclatura já colocadas, devemos acrescentar o símbolo "%". Faça:

```
A% = 1
B% = 2.3
C% = 0.05
D% = -2.8
PRINT A%
```

**R-37**

```
PRINT B%
```

**R-38**

```
PRINT C%
PRINT D%
```

**R-39**

Valores fora dos limites estipulados causam mensagem de erro. Tente:

```
F1% = 32767
F2% = 32768
```

**R-40**

## III. VARIÁVEIS ALFANUMÉRICAS

Do mesmo modo que números são armazenados nas variáveis numéricas, cadeias de caracteres, ou *strings*, são armazenadas em variáveis alfanuméricas ou variáveis *strings*.

As regras das variáveis numéricas são aplicadas, de um modo geral, às variáveis *strings*, exceto se estas possuem um símbolo de cifrão (*dollar sign*) $ em seu final.

Digite:

```
LET A$ = "SAO PAULO"
LET B$ = "BRASIL"
PRINT A$
```

**R-41**

```
PRINT B$
```

**R-42**

Perceba que os caracteres a serem armazenados devem vir sempre entre aspas.

Digite:

```
LET X$ = "45"
LET N1$ = "ENLACE"
PRINT X$
```

**R-43**



```
PRINT N1$
```

**R-44**

Tente agora:

```
LET Y$ = 54
```

**R-45**

```
LET N1 = "ENLACE"
```

**R-46**

```
LET A$ = B
```

**R-47**

```
LET M = F1$
```

**R-48**

Compare a sua explicação às respostas obtidas anteriormente com as fornecidas nos apêndices 5 e 8.

É importante notar que o computador também aceita, simultaneamente, a variável numérica **A**.

Faça:

```
LET A = 1234
```

e, em seguida,

```
PRINT A$
```

**R-49**

```
PRINT A
```

**R-5ø**

## III.1 – Particularidades do Applesoft

a) As cadeias *strings* a serem armazenadas podem ter desde o tamanho zero (*string* nulo) até 255 caracteres.

b) O comando **CLEAR** zera todas as variáveis; nas variáveis *string* coloca o *string* nulo em todas elas.

c) O comando **LET** é também opcional.

d) A instrução **A$** = " " coloca um *string* nulo em **A$**.

e) Como nas variáveis numéricas, apenas os dois primeiros caracteres são reconhecidos.

## III.2 – Concatenando Strings

Como as variáveis alfanuméricas não são quantidades, não podemos fazer operações algébricas com elas. No entanto, podemos fazer a operação de *concatenação.*

Digite:



```
NOME$ = "APPLE"
PRENOME$ = "COMPUTADOR"
E$ = "ø"
```

(o símbolo "ø" significa espaço em branco)

Agora faça:

```
PRINT PRENOME$ + E$ + NOME$
```

**R-51**

A operação de concatenação nada mais é do que a junção das cadeias de caracteres das variáveis *strings* em questão, e o seu operador é o símbolo **+**.

## IV. EXERCÍCIOS PROPOSTOS

2.1 Quando digitamos um comando direto e pressionamos *RETURN,* o que o computador entende?

2.2 Suponha que você digitou erradamente um comando ou uma mensagem. Como você procede para corrigir esse engano, antes de pressionar *RETURN*?

2.3 Se digitarmos:

```
LICAO DE CASA
```

o que o computador imprimirá?

2.4 Se digitarmos:

```
PRINT "LICAO DE CASA"
```

o que o computador imprimirá?

2.5 Digitando:

```
PRINT LICAO DE CASA
```

qual a mensagem do computador?

2.6 Fazendo:

```
A$ = "APPLE"
B$ = "***Ø"
C$ = "Ø***"
```

qual a resposta do computador ao comando

```
PRINT B$ + A$ + C$ ?
```

2.7 Fazendo z = 2, dê as respostas para:

```
PRINT Z-Z-Z
PRINT Z+Z/Z-Z
PRINT (Z+Z)/(Z-Z)
```

2.8 Digitando:

```
A=3
B=5
C=1Ø
```

dê as saídas das instruções:

```
PRINT (A/B)+C
PRINT (C-B)*A
PRINT B*C+1/B
PRINT B+C*A-C+A*B/C^A
```

2.9 O comando **LET A + B = C** é um comando válido em Applesoft BASIC?

2.10 Indique quais das variáveis abaixo não poderão assumir seus valores e explique por quê.

```
NOME$ = "JOSE"
NUMERO = 37
VALOR = 7
ENTRADA$ = CARLOS
JOAO = PEDRO
OURO = 5
PRATA = OURO
CARLOS = 45
JOAO = "WRITE"
ANTONIO = "38"
NUMBER$ = "135"
ANOTACAO$ = DADO
INTEIRO = 1
DOS$ = "PEDRO"
CATALOG$ = 138
```



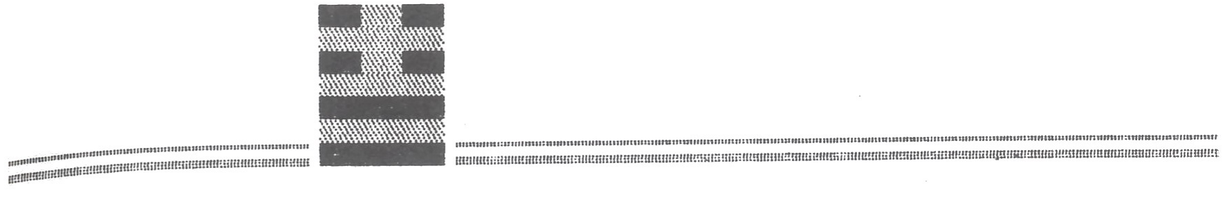
# V. RAXAKUKA

2.1 Se, de um número de 3 algarismos, subtrairmos 7, ele se tornará divisível por 7; se subtrairmos 8, ele se tornará divisível por 8 e, se subtrairmos 9, ele se tornará divisível por 9. Qual é o número?

2.2 Cinco meninos estavam assistindo televisão. Eles estavam sentados em 2 cadeiras e 3 poltronas. Você pode descobrir onde estavam sentados A, B, C, D e E, se você souber que:

A e B sentavam-se num mesmo tipo de assento;
B e D sentavam-se em tipos diferentes;
D e F sentavam-se em tipos diferentes.

2.3 Dois homens vão fazer uma viagem de 18 ØØØ km, de automóvel. Entretanto, os pneus só agüentam 12 ØØØ km. Quantos pneus, no mínimo, precisam levar de reserva?



# Iniciando a programação

*"Os doutos fatigam-se nas especulações que giram em torno de si mesmos; escrevo somente para vós, homens sem idéias pre-concebidas, que sabeis verdadeiramente filosofar, que procurais a ciência não apenas nos livros, mas nas próprias coisas."*

*William Gilbert*
*(1544-1603)*

# I. INTRODUÇÃO

Nos capítulos anteriores, tivemos os primeiros contatos com o modo de trabalhar do computador e vimos que o computador obedeceu certas instruções e comandos como se fosse uma calculadora.

Mas, se tivéssemos que fazer repetidas vezes uma mesma seqüência de cálculos, ou se tivéssemos que seguir sempre a mesma rotina de trabalho, onde reside a mágica do computador?

A programação, a que fica sujeito um computador, não é uma espécie de mágica que resolverá todos os seus problemas ou dificuldades; é simplesmente uma *seqüência lógica de instruções* a serem seguidas, instrução por instrução e sem pular nenhuma, *para se alcançar um resultado ou um objetivo.*

Para se efetuar um programa, devemos, antes, estruturar o algoritmo e o diagrama de blocos do problema em questão e, evidentemente, você deverá saber resolver o problema. Não devemos esquecer que a nossa capacidade de estruturar algoritmos e programá-los é resultado de treinamento contínuo e raciocínio aplicado. *Quanto mais fizermos, melhor saberemos fazê-los.*

# II. ALGORITMO, FLUXOGRAMA, PROGRAMA: NOVOS TERMOS

Neste parágrafo, você terá uma participação mais ativa no andamento dos trabalhos que aqui se desenvolverão. Falaremos sobre algoritmos, fluxogramas, diagramas de bloco, estruturação de programas, codificação, para, em seguida, executarmos ou processarmos nossos problemas.

Você saberia explicar por que o procedimento da divisão de 85 por 4 é o mostrado pela figura 3.1?

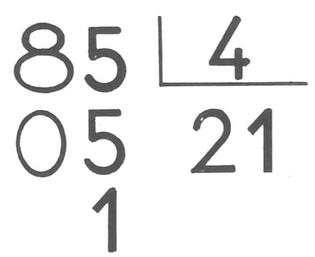



fig. 3.1



Também já reparou que o procedimento na divisão de dois números é sempre o mesmo?

Este fato ocorre porque estamos usando o *algoritmo* da divisão, um procedimento tão antigo e comum que quase não é necessário pensar para executarmos a divisão.

A motivação que leva o indivíduo a elaborar um programa é sempre a procura de solução ou soluções eficientes para um determinado tipo de problema.

Você estará procurando soluções numéricas para alguns tipos de problemas que serão apresentados oportunamente. Ao encontrar esses tipos de problemas, o seu mecanismo de raciocínio começará a entrar em ação para definir o problema como um todo, seguindo os passos:

a) *Identificação do Problema:* consiste em definir exatamente qual a situação a ser estudada e nesse ponto podemos abrir os subitens:

a.1 – Leitura do problema: neste passo você procurará a(s) pergunta(s) do problema dando-lhe(s) a maior atenção possível.

a.2 – Determinações dos assuntos relacionados: aqui, procurará identificar todos os conceitos envolvidos, métodos, algoritmos e as equações necessárias.

a.3 – Identificação dos dados: busca dos elementos essenciais à solução do problema.

a.4 – Fixação do objetivo: é necessário definir muito bem o fim do problema, isto é, uma vez atingido um resultado, verificar se é necessário continuar ou se já se chegou ao resultado final.

A partir desse ponto, começa a nascer a solução propriamente dita. Dos passos acima deve surgir uma rotina de trabalho, ou seja, um tratamento metódico na organização da solução do problema.

b) *Elaborar o Fluxograma:* definir, simbolicamente, quais as entradas, quais as saídas, detalhando as operações lógicas e matemáticas.

c) *Codificação*: traduzir o fluxograma produzido no passo b para a linguagem que o computador opera, no nosso caso o **Applesoft BASIC**.

d) *Depurar:* testar o programa procurando os erros. Quando encontrado um erro, se este for de sintaxe, isto é, tiver forma errada, devemos voltar ao passo c para traduzi-lo corretamente; se, por outro lado, o erro for de lógica, devemos voltar ao passo b, e se for de método, voltar ao passo a.2.

## II.1 – Algoritmo

Como definição, *algoritmo é um conjunto de regras gerais que especificam uma seqüência de operações utilizadas na resolução de um determinado problema.*

O conceito básico de um conjunto de instruções que especificam uma seqüência de operações é aplicado em muitos e diferentes campos. A figura 3.2, que ilustra uma receita de culinária, é um tipo de algoritmo.

**SOPA LAVOISIER**

Ingredientes:

3 batatas
2 cenouras
2 mandioquinhas
1 tomate
150g de músculo
1 ovo
1 colher (sobr.) sal
1 colher (sopa) queijo ralado
1 colher (chá) manteiga

Preparo:

Em uma panela, coloque 3 copos de água fria; junte as batatas, as cenouras, as mandioquinhas, o músculo, o sal e o tomate.

Ponha a panela ao fogo, até ferver. A seguir, tampe a panela e deixe em fogo lento por 25 minutos. Após esse tempo, jogue o conteúdo no liquidificador, batendo até atingir a consistência desejada.

Quebre o ovo no prato em que se vai servir e, após um prévio aquecimento, derrame a sopa por cima do ovo. Coloque a manteiga e cubra com queijo ralado.

fig. 3.2

De modo semelhante, as instruções para se determinar a massa específica de um corpo metálico irregular formam um algoritmo. Veja a figura 3.3.

**MASSA ESPECÍFICA**

Equipamento:
balança, cilindro graduado

Instruções:

1. Medir, por meio da balança, a massa do corpo. Seja m sua massa, em gramas.
2. Colocar água (suficiente para cobrir o corpo) no cilindro graduado. Seja $V_1$ o volume no cilindro, em $cm^3$.
3. Colocar o objeto metálico no cilindro. Seja $V_2$ o novo volume no cilindro, em $cm^3$.
4. Calcular a massa específica do corpo metálico fazendo $\mu = \frac{m}{V_2 - V_1}$, em $g/cm^3$.

fig. 3.3



Um outro exemplo, um pouco estranho, é mostrado pela figura 3.4.

**ROLINHO**

Montar 24 pp. ag. 5. 1ª CARR: 12m: 2m, 2t, 2m, 2t, 2m, 2t; 2ª CARR: 2m, 2t, 2m, 2t, 2m, 2t, 12t; 3ª CARR: 12t, 2m, 2t, 2m, 2t, 2m, 2t; 4ª CARR: ig. 1ª CARR; 5ª CARR: ig. 2ª CARR; etc.

Depois da 14ª CARR, inv., i.é.: 2t, 2m, 2t, 2m, 2t, 2m, 12t; e ass. p. diante.

fig. 3.4

A maioria dos leitores não encontrará sentido algum para esse conjunto de instruções; porém, para quem estiver acostumado a essa linguagem e a essas abreviações, comuns e corriqueiras na arte de tricotar, facilmente decifrará este algoritmo.

Como podemos estruturar um algoritmo e saber se está correto ou não?

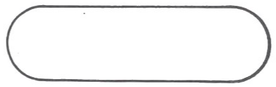
A primeira resposta razoável é tentar elaborar o algoritmo a partir dos conceitos envolvidos no problema, e tomar um conjunto de dados relativos ao mesmo tipo de problema, cuja resposta é conhecida, e testá-los no algoritmo elaborado, sempre usando o computador mais barato do mundo: papel, borracha, lápis e... cérebro!

## II.2 – Fluxograma

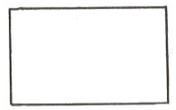
*Fluxograma* ou *diagrama de blocos* é o desenho de um algoritmo.

É muito útil o uso do fluxograma para descrever os vários estágios da realização de um conjunto de operações antes de se tentar escrever as instruções detalhadas no programa. Para cada algoritmo poderemos fazer uma representação gráfica capaz de mostrar todos os passos a serem seguidos.

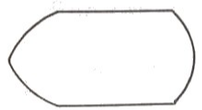
Existem símbolos adequados, convencionais, que facilitam a comunicação universal daquelas pessoas que trabalham com programação e elaboram algoritmos. A figura 3.5 mostra alguns símbolos mais comumente utilizados e seus significados.



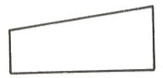
*Terminal:* ponto de início, término ou interrupção de um diagrama.



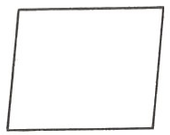
*Processamento:* conjunto de instruções cuja função é executar um procedimento.



*Saída:* exibição dos dados processados em terminal de vídeo.

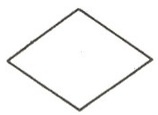


*Entrada:* entrada de dados, via teclado.

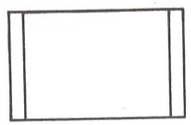


*Entrada/Saída E/S* ou *I/O:* função de entrada ou saída de um periférico qualquer.

*Saída:* exibição dos dados processados em impressora (formulário).



*Comparação/Decisão:* indica a possibilidade de desvios para diversos pontos do programa (fluxo).



*Processamento Predefinido: Sub-rotinas* grupo de instruções não incluídas no diagrama de blocos.


indica continuação do diagrama na mesma página e N é um número de referência.

indica continuação em outra página, onde N é um número de referência.

cada uma destas setas indica o "caminho" natural de um diagrama de blocos.

fig. 3.5

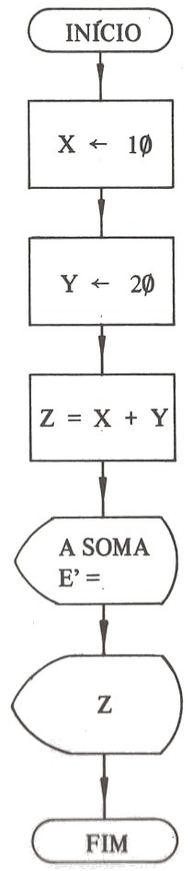


## II.3 – Programa

Após elaborarmos o algoritmo do problema, construir o fluxograma correspondente, verificá-lo através de testes, analisar as possíveis variações e fazer os comentários necessários, estamos prontos para programá-lo, isto é, codificá-lo na linguagem inteligível ao nosso computador. Por exemplo:

Desenhe o fluxograma e codifique, em linguagem *BASIC*, o algoritmo que resolve a expressão Z = X + Y, onde X = 1Ø e Y = 2Ø.

Fluxograma




Codificação

```
1Ø  LET X = 1Ø
2Ø  LET Y = 2Ø
3Ø  LET Z = X + Y
4Ø  PRINT "A SOMA E'"
5Ø  PRINT Z
```

P3.1

Embora este exemplo seja muito simples, ele mostra algumas instruções, já nossas conhecidas, sendo operadas em *modo indireto*, isto é, não são executadas via teclado (*comando direto* ou *execução imediata*), mas sim executadas uma a uma, segundo a ordem numerada, numeração à esquerda, e após ao comando **RUN**.

Digite a seqüência de linhas do programa P3.1, não se esquecendo do *RETURN*, no fim de cada linha, e, em seguida, faça:

```
RUN
```

**R-1**

Ao digitarmos **RUN**, todas as linhas de instrução armazenadas na memória de programação do computador foram executadas, uma a uma, e na seqüência estipulada; porém o computador não "esqueceu" o programa introduzido.

Digite, novamente:

```
RUN
```

**R-2**

E ainda tem mais: você pode limpar a tela do monitor de vídeo que o computador não esquece o programa.

Digite:

```
HOME
```

O que aconteceu? O que faz o comando **HOME**?

**R-3**

Será que o programa P3.1 foi apagado?



**R-4**

Digite, agora:

```
RUN
```

**R-5**

O comando **HOME** só apaga o conteúdo da tela, não alterando o programa.
Quando você digitou:

```
10 LET X=10
```

e em seguida pressionou a tecla *RETURN*, aparentemente nada aconteceu, isto é, nada de novo apareceu na tela do monitor de vídeo. No entanto, o computador armazenou essa linha de programa. **Toda vez que uma frase for iniciada por um número ou por um algarismo, essa frase será entendida pelo computador como sendo uma instrução, isto é, uma frase de programação contendo os comandos necessários para a sua execução.** Para visualizarmos o conteúdo da memória de programação do computador, ou para visualizarmos o programa, acionamos o comando **LIST**.

## III. APRESENTANDO: LIST, NEW E REM

Limpe a tela, digitando **RUN** e, em seguida, faça:

```
LIST
```

**R-6**

O que faz o comando **LIST**?

**R-7**

Agora, muita atenção: digite, na seqüência, os comandos abaixo:

```
HOME
NEW
LIST
```

Verificamos que nada apareceu na tela. Ao digitarmos o comando **NEW** e pressionarmos *RETURN*, ocorreu o "apagamento" da memória de programação do computador, isto é, o programa é *perdido permanentemente.*

Digite:

```
RUN
```

**R-8**

e nada aparecerá na tela do monitor de vídeo. É claro, pois você "apagou" a memória de programação do computador.

Cada um dos números 1Ø, 2Ø, 3Ø, 4Ø e 5Ø, escritos na frente de cada linha, ou instrução, é denominado *número de linha,* e por que não usarmos 1, 2, 3, 4 e 5?

Carregue o computador com o programa P3.2 abaixo:

```
1 PRINT "A"
2 PRINT "A"
4 PRINT "A"
Ø PRINT "P"
3 PRINT "N"
```

P3.2

OBS.: Como convenção, o símbolo ♭ (b cortado) significa espaço em branco (*space-bar*), lembra-se?

Digite, a seguir, **RUN**; o computador mostrará:



```
P
A
A
N
A
```

Por intermédio do comando **LIST**, faça a listagem do programa.

```
Ø  PRINT "P"
1  PRINT "A"
2  PRINT "A"
3  PRINT "N"
4  PRINT "A"
```

Tudo bem. Vimos que podemos numerar as linhas em ordem qualquer que o computador as executará e as listará sempre em ordem crescente; uma outra informação importante: a primeira linha permitida é Ø e a linha de maior número, permitida pela estrutura do nosso computador, é a de número 63999.

Entretanto, a mensagem pretendida era:

```
P
A
R
A
N
A
```

Não podemos simplesmente adicionar ao programa a linha:

```
1.5 PRINT "R"
```

pois a variação permitida na numeração das linhas é unitária. Então, para inserir **PRINT "R"**, devemos redigitar o programa a partir da linha 2, isto é:

```
2  PRINT "R"
3  PRINT "A"
4  PRINT "N"
5  PRINT "A"
```

cuja listagem fornecerá:

```
Ø  PRINT "P"
1  PRINT "A"
2  PRINT "R"
3  PRINT "A"
4  PRINT "N"
5  PRINT "A"
```

P3.3

Faça **RUN** e observe o monitor de vídeo:

**R-9**

Pronto. A mensagem desejada está na tela. Para evitarmos a dificuldade de inserir ou adicionar novas instruções, vamos, via de regra, numerar as instruções de 1Ø em 1Ø.

Digite **NEW** e carregue o computador com o programa P3.3:

```
1Ø  PRINT "P"
2Ø  PRINT "A"
3Ø  PRINT "R"
4Ø  PRINT "A"
5Ø  PRINT "N"
6Ø  PRINT "A"
```

P3.3

Digite, agora, **RUN**.
A resposta do computador é:

**R-1Ø**

Sem digitar **NEW**, carregue o computador com o programa P3.4:

```
1Ø  PRINT "PARANA"
2Ø  PRINT
3Ø  PRINT "P";
4Ø  PRINT "A"
5Ø  PRINT "R";
6Ø  PRINT "A"
7Ø  PRINT "N";
8Ø  PRINT "A"
```

P3.4



A seguir, digite **RUN**:

**R-11**

Como o programa anterior (P3.3) foi apagado sem acionarmos o **NEW**?

**R-12**

O que faz a instrução 2Ø?

**R-13**

Em que alterou a resposta a presença do " ; " (*ponto-e-vírgula*)?

**R-14**

Vamos retomar o primeiro programa, P3.1, com algumas modificações.
Digite **NEW** e a seguir:

```
5  HOME
1Ø  REM  SOMA SIMPLES
2Ø  REM  O VALOR 1Ø E' ATRIBUIDO A X
3Ø X = 1Ø
4Ø  REM  O VALOR 2Ø E' ATRIBUIDO A Y
5Ø Y = 2Ø
6Ø Z = X + Y
7Ø  PRINT "A SOMA E' ";Z
```

P3.5

Faça a listagem e a seguir rode o programa P3.5.

**R-15**

O que faz a instrução 5?

**R-16**

Quais as diferenças entre as instruções 4Ø e 5Ø do P3.1 e a instrução 7Ø do P3.5?

**R-17**

O comando **REM** indica que a linha contém um *remark* (observação ou comentário). **REM** é usado para documentar um programa; esse comando não conduz a nenhum tipo de trabalho ou desvio; o computador simplesmente ignora a linha remarcada.

O *remark* descreve o propósito da linha ou das instruções, e, portanto, é muito útil para definir variáveis, sub-rotinas, arquivos e, também, quando o programa for utilizado por outras pessoas, as informações e detalhes ficam mais acessíveis.

Digite agora:

```
1Ø
2Ø
4Ø
```

e, em seguida, rode o programa.

Qual a resposta que você obteve? É a mesma que a obtida anteriormente?

**R-18**



Faça a listagem do programa que já está inserido no computador, digitando **LIST**:

**R-19**

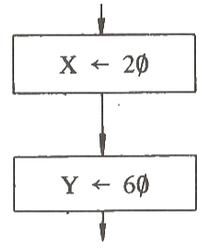
O programa acima é idêntico ao programa P3.5? É equivalente?

**R-2Ø**

O que causa no programa digitarmos o número da linha e a seguir pressionarmos *RETURN*?

**R-21**

Para mudarmos os valores das variáveis X e Y, como proceder?
É só mudar o fluxograma para, por exemplo,




E as linhas 3Ø e 5Ø para:

```
3Ø  X=2Ø
5Ø  Y=6Ø
```

Rode novamente o programa P3.5 e comprove a sua alteração.

## III.1 – Mais Funções do LIST

Reintroduza novamente no computador o programa P3.5, e digite:

```
LIST 6Ø
```

**R-22**

Perceba que o computador 'listou' apénas a linha de número 6Ø.

```
LIST 2Ø-5Ø
```

**R-23**

O comando **LIST** *nnn - lll* lista todas as linhas desde a de número *nnn* (inclusive) até a de número *lll* (inclusive). Faça:

```
LIST Ø-1ØØ
```

**R-24**

```
LIST 3Ø-
```

**R-25**



```
LIST -6Ø
```

**R-26**

**LIST Ø – 1ØØ:** lista as linhas entre Ø e 1ØØ (Ø e 1ØØ inclusive)
**LIST – 6Ø:** lista as linhas todas, até a linha 6Ø (inclusive)
**LIST 3Ø –:** lista todas as linhas, desde o número 3Ø (inclusive), até a última linha do programa.

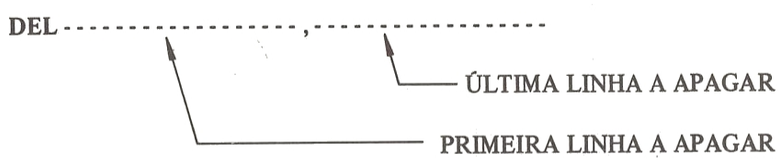
Para parar uma listagem enquanto o computador estiver imprimindo as linhas, aperte a tecla *CTRL* e, simultaneamente, a tecla *C*. *CTRL-C* interrompe, também, a execução de um programa (às vezes é necessário *CTRL-C* e *RETURN*).

Outra maneira de parar uma listagem é apertar *CTRL-S*; nesse caso, apertando qualquer tecla a listagem continua.

Já foi visto, durante a explicação do primeiro programa, que, se quisermos mudar uma das linhas de um programa, teremos de redigitar a nova linha com o mesmo número da antiga.

Para se apagar uma linha qualquer, basta digitar o número da linha e pressionar *RETURN*.

Esse recurso é pouco prático para apagar várias linhas de um programa. Nesse caso, existe o comando **DEL**, que tem a forma:




Digite:

```
DEL 5,2Ø
RUN
LIST
```

Você percebeu que, embora o programa forneça os mesmos resultados, estes são apresentados de maneira um pouco diferente. No entanto, ao observar a listagem do programa, agora já modificado, nota-se que, com efeito, todas as linhas entre 5 e 2Ø foram apagadas da memória, mas o programa ainda existe ...e é executável.

# IV. UMA INTRODUÇÃO AO DISCO

Antes de começar a trabalhar com a unidade de disco, veremos como preparar um disco virgem para poder ser usado no computador. Este processo, chamado de inicialização, é discutido no apêndice 3. Com o computador desligado, pegue o disco que veio com o computador (provavelmente com o título de *SYSTEM MASTER*) e abra a porta do seu *drive*. Insira o disco, no *drive*, com a etiqueta virada para cima e voltada para você, e em seguida feche o *drive*. Ligue o computador *sem pressionar RESET*, conforme as instruções no capítulo 1. Se a unidade de disco não parar após alguns segundos, tipicamente 5 a 2Ø segundos, você provavelmente está com o disco errado, isto é, não inicializado.

Para preparar (inicializar) um disco novo, tire o *SYSTEM MASTER* do *drive*, substituindo-o pelo disco a ser inicializado. Tendo o *DOS* no computador, digite:

```
NEW

1Ø  HOME
2Ø  PRINT "ESTE E' O MEU PRIMEIRO DISCO"
3Ø  PRINT "INICIALIZADO EM 17/1Ø/83"
4Ø  END

INIT HELLO
```

O disco entrará em funcionamento por quase um minuto. Quando parar, o disco estará pronto para ser usado.

## IV.1 – Gravando um Programa

Vamos gravar no disco recém-inicializado o programa P3.5. Digite-o novamente, eliminando eventuais erros de datilografia. Para gravá-lo com o nome de *SOMA*, digite

```
SAVE SOMA
```

O programa foi transferido ao disco, mas ainda está na memória do computador (tente um **LIST**).

Faça agora:

```
CATALOG
```



Qual a saída?

**R-27**

Os nomes de todos os programas gravados no disco aparecem na tela, incluindo o programa *HELLO*, que é rodado automaticamente sempre que o computador é ligado com esse disco.

## IV.2 – Recuperando Programas do Disco

Digite **NEW** para apagar o programa da memória. Para rodar o programa *SOMA* do disco, podemos digitar:

```
RUN SOMA
```

O *drive* é acionado e o programa é carregado e executado. Tente gravar e carregar programas, sabendo que um nome de programa deve começar com uma letra e conter até 3Ø caracteres.

Às vezes não é desejável rodar o programa quando este é carregado. Para isso, usamos o comando

```
LOAD SOMA
```

O programa *SOMA* é transferido à memória, e pode ser executado com um

```
RUN
```

Para maiores informações sobre as operações com disco, consulte o apêndice 3.

# V. EXERCÍCIOS PROPOSTOS

3.1 Explique, em poucas palavras, as finalidades dos comandos **HOME, NEW, LIST** e **RUN**.

3.2 Qual a resposta do computador aos programas abaixo:

a)
```
10  PRINT "JOAO"
20  PRINT "   "
30  PRINT "MARIA"
```

b)
```
10  PRINT "JOAO"
20  PRINT "   ";
30  PRINT "MARIA"
```

c)
```
10  PRINT "JOAO";
20  PRINT
30  PRINT "MARIA"
```

3.3 Explique o significado dos comandos abaixo:

a) variável numérica = valor numérico
b) variável *string* = "expressão"
c) **PRINT A$ + B$; A + B**

3.4 Dê a saída do programa abaixo:

```
10  HOME
20 A$ = "(X @ @ X)"
30 B$ = "    XXX"
40 C$ = "  XXXXX"
50 D$ = "  X   X"
60 E$ = " X     X"
70 F$ = " X < > X"
80 G$ = " X  V  X"
90  PRINT C$
100  PRINT E$
110  PRINT A$
120  PRINT G$
130  PRINT F$
140  PRINT D$
150  PRINT B$
```

3.5 Como você procede para apagar uma linha inteira do programa?

3.6 Faça o fluxograma e o programa para resolver a equação $x^2 - x - 12 = 0$ e cuja saída deve ser:

```
RAIZES DA EQUACAO X^2 - X - 12 = Ø
X1 = ___
X2 = ___
```



3.7 Faça o fluxograma e o programa para resolver a equação $x^4 - 13x^2 + 36 = 0$, tendo como saída a mensagem:

```
RAIZES DA BIQUADRADA: X^4 - 13X^2 + 36 = Ø
X1 = ____
X2 = ____
X3 = ____
X4 = ____
```

3.8 Uma caixa d'água, em forma de cilindro reto, possui raio = 3m e altura = 5m. Desenvolva um programa que forneça o volume, em $m^3$ e em litros, e ainda a sua área lateral, em $m^2$. A saída deve ser:

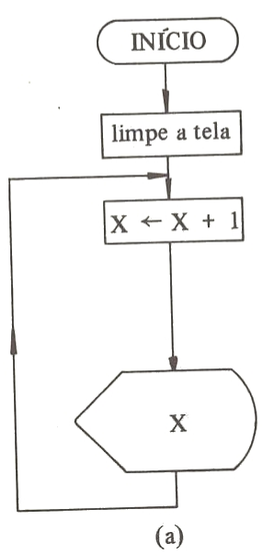
```
CAIXA D'AGUA CILINDRICA
RAIO = 3M    ALTURA = 5M
VOLUME (M^3) = ___
VOLUME (L) =
AREA LATERAL (M^2) = ___
```

3.9 Suponha que desejamos pintar a caixa d'água do problema anterior, com um número suficiente de mãos para que a camada de tinta tenha uma espessura de 4mm. A massa específica da tinta é 3,8 g/$cm^3$.

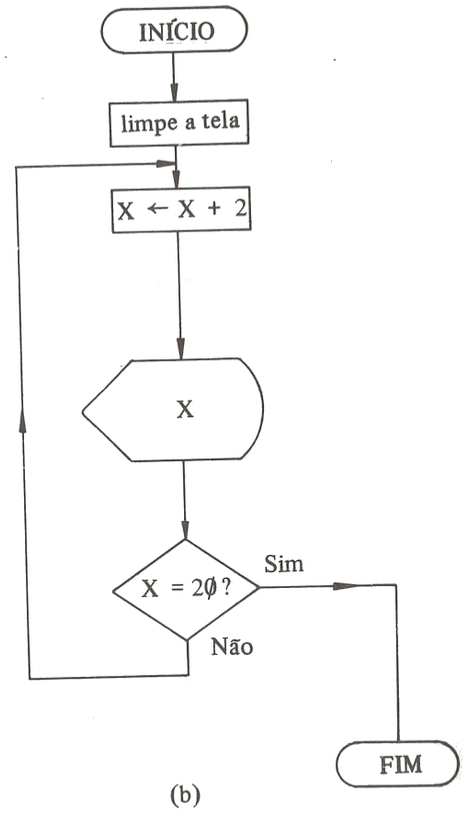
Adote g = 9,8 m/$s^2$. Faça um programa que apresente os dados abaixo:

```
CAIXA D'AGUA CILINDRICA
RAIO = 3M    ALTURA = 5M
VOLUME (M^3) = ____
VOLUME (L) = ____
AREA LATERAL (M^2) = ____
VOLUME DE TINTA (L) = ____
ACRESCIMO DE PESO (KGF) = ___
```

3.10 Explique o que indica cada fluxograma abaixo:




(a)




(b)

OBS.: As variáveis do computador são inicializadas, isto é, o Applesoft coloca zeros em todas as variáveis ao ser ligado o computador.

3.11 Por que as linhas de programa são normalmente numeradas de 1Ø em 1Ø, a partir do número 1Ø?

3.12 São dadas 8 pérolas idênticas em forma e tamanho. Uma delas é falsa e mais leve que as demais, as quais possuem o mesmo peso. Dispõe-se de uma balança mágica de dois pratos, que se desintegra após a segunda pesagem, isto é, uma terceira é impossível. Escreva o algoritmo que descobre a pérola falsa e desenhe seu fluxograma.



## VI. RAXAKUKA

3.1 Progredir não significa morte sem desonra, mas retroceder não significa desonra sem morte.

Logo:

a) retroceder significa morte sem desonra.

b) progredir poderia significar desonra sem morte.

c) progredir poderia significar morte sem desonra.

Responda quais são proposições falsas e quais são as verdadeiras.

3.2 Um homem estava morrendo e sua mulher estava para ter criança. No testamento, deixou 2/3 dos seus bens para o filho (se fosse homem) e 1/3 para sua mulher. Se a criança fosse mulher receberia apenas 1/3 e a esposa 2/3. Após sua morte, a mulher deu à luz gêmeos, um menino e uma menina. Como pode o juiz dividir o dinheiro, de acordo com os desejos do morto?

3.3 Dispõe-se de uma quantidade ilimitada de água em um reservatório, de uma lata de 5 litros e de outra lata de 4 litros. Escreva um algoritmo para se colocar, exatamente, 2 litros de água na lata de 5 litros.

# Formatação da tela e entrada de dados

*"Ai de mim! O vosso amigo e servo Galileu tem estado no último mês desesperadamente cego, de modo que este céu, esta Terra, este Universo, que eu, por maravilhosos descobrimentos e claras demonstrações, alarguei cem mil vezes além da crença dos sábios da antigüidade, se reduzem, daqui por diante, para mim, a um diminuto espaço preenchido pelas minhas próprias sensações corpóreas."*

*Galileu Galilei*
*(1564-1642)*

# I. O COMANDO PRINT

Já vimos, em estudos anteriores, algumas particularidades do comando **PRINT**; outras veremos a seguir. O ponto de interrogação? pode ser usado como abreviação para o comando **PRINT**; numa listagem, esse ponto de interrogação aparece como **PRINT**.

Sem qualquer mensagem a ser impressa, o computador, ao executar o comando **PRINT**, coloca uma linha em branco no terminal de vídeo. Digite:

```
10  HOME
20  PRINT "PRIMEIRA MENSAGEM"
30  PRINT
40  PRINT "SEGUNDA MENSAGEM"
```

P4.1

Digite **RUN**.

**R-1**

Qual a função da instrução 30?

**R-2**

## I.1 – O PRINT com Ponto e Vírgula

Vimos até agora que cada **PRINT** começa a imprimir numa linha nova. Isso acontece porque o **PRINT** anterior, após ser executado, mandou à tela um caractere *RETURN*, que faz o BASIC pular para a próxima linha de vídeo. Esse *RETURN* final pode ser "anulado", colocando no final da instrução um ponto-e-vírgula (;). O mesmo símbolo serve de separador, quando temos mais de um *string* ou expressão



numérica no mesmo **PRINT**. Assim sendo, ao colocarmos um ponto-e-vírgula após um elemento de um **PRINT**, estamos avisando ao BASIC que o próximo caractere deverá ser impresso logo após o último, já colocado na tela.

Faça, como exemplo:

```
A$ = "APPLE"
PRINT "O BASIC DO ";A$;" E' O ";A$;"SOFT."
```

**R-3**

```
PRINT "10 + 7 = ";10+7
```

**R-4**

```
PRINT A$;"Ø";16*3;"K"
```

**R-5**

```
NEW

10  PRINT "O ";
20  PRINT "BRASIL"
30  PRINT "E'";
40  PRINT "|TRICAMPEAO"
```

P4.2

**R-6**

Por que a saída do programa P4.2 deu-se em duas e não em quatro linhas?

**R-7**

## I.2 – O PRINT com Vírgula

Através de outra variação do **PRINT**, podemos criar tabelas de duas ou três colunas de forma rápida e simples. O BASIC separa as quarenta posições horizontais de vídeo em três colunas de 16,16 e 8 caracteres, respectivamente. A tabulação é feita pela vírgula: ao encontrar uma vírgula, o BASIC pula para o início do próximo campo (ou coluna) livre, indo à próxima linha se necessário.

Digite o seguinte programa:

```
5  REM  TESTE DA VIRGULA
1Ø NO$ = "JOSE' DA SILVA"
2Ø TE$ = "549-217Ø"
3Ø ID$ = "31"
4Ø  HOME
5Ø  PRINT "NOME","IDADE","FONE"
6Ø  PRINT
7Ø  PRINT NO$,ID,TE$
```

P4.3

Qual a saída?

**R-8**

Mude agora as linhas 1Ø e 2Ø:

```
1Ø NO$ = "MANUEL ANTONIO DE SOUZA"
3Ø ID$ = "TRINTA E UM"
```



Rodando o programa, como você explica a saída?

**R-9**

O primeiro campo de tabulação compreende as 16 primeiras posições, a partir da esquerda, na linha de vídeo. O segundo *tab field* ocupa as 16 posições seguintes (da 17 até a 32) e está disponível somente se o primeiro *tab field* não estiver ocupado até a posição 16. O terceiro *tab field* consiste nas 8 posições seguintes (da 33 até a 4Ø) e está disponível somente se nada ocupar as posições 24 até 32.

Digite:

```
PRINT "123456789Ø12345","12345","12345"
PRINT "12345678Ø123456","12345","12345"
PRINT "12345678Ø123456","12345678Ø123456","12345"
```

Comente a diferença das três situações acima.

**R-1Ø**

Coloque agora no computador o programa P4.4:

```
5  HOME
1Ø A = 5
2Ø B = 2
3Ø SOMA = A + B
4Ø SUBTRACAO = A - B
5Ø DIVISAO = A / B
6Ø  PRINT "A = ";A,"B = ";B
7Ø  PRINT "SUBTRACAO","SOMA","DIVISAO"
8Ø  PRINT SU,SO,DI
```

P4.4

Antes de rodá-lo, tente prever a saída do programa:

**R-11**

Faça, agora, o programa P4.5:

```
1Ø  PRINT "TABUADA DO 4"
2Ø  PRINT 4 * Ø,4 * 1,4 * 2,4 * 3,4 * 4,4 * 5,4 * 6,4 *
 7,4 * 8,4 * 9,4 * 1Ø
```

P4.5

O que você obteve após o **RUN**?

**R-12**

Como exercício, elabore um programa que apresente a seguinte saída, sendo que o número de anos deve estar gravado na variável ANOS e o número de meses e dias calculado:

| ANOS | MESES | DIAS |
|---|---|---|
| 4 | 48 | 146Ø |

**R-13**

## I.3 – O Comando TAB

Para colocarmos um caractere em uma determinada posição da linha, da posição 1 à posição 4Ø (na realidade de 1 a 256, que é o tamanho máximo de uma "linha lógica" do computador), usamos o comando **TAB** (tabulador).



**TAB (X)** deve ser usado *sempre* com o comando **PRINT** e uma expressão numérica deve ser colocada entre os parênteses, definindo a posição de tabulação, a partir da margem esquerda da linha.

O comando **TAB** coloca o cursor sobre a posição especificada pela expressão aritmética ou pelo número de tabulação, contando a partir da esquerda. Caso você ultrapasse o máximo ou o mínimo da tabulação, a mensagem

```
?ILLEGAL QUANTITY ERROR
```

aparecerá no vídeo. É importante observar que o comando **TAB** *nunca faz retornar o cursor* e que a instrução **PRINT TAB (Ø)** coloca a impressão na posição 256. Digite o seguinte programa:

```
5  HOME
1Ø  PRINT "ALO AQUI"
2Ø  PRINT  TAB( 33);"ALO ALI"
3Ø  PRINT
4Ø N = 8
5Ø  PRINT  TAB( N);N; TAB( 2 * N);2 * N; TAB( N + 22);N
 + 22; TAB( N * 3);N * 3; TAB( 87);87; TAB( Ø);Ø
```

P4.6

Faça **RUN** e escreva o resultado abaixo.

**R-14**

Por que **ALO AQUI** e **ALO ALI** não estão escritos na mesma linha no terminal de vídeo?

**R-15**

Qual a alteração que você faria no programa para obter **ALO AQUI** e **ALO ALI** escritos na mesma linha no monitor?

**R-16**

Estruture e teste um programa que apresente a seguinte saída, usando a tabulação para "formatar" a saída:

```
ANOS        MESES       DIAS        HORAS

5           60          1825        43800
```

O número de anos é armazenado na variável A, e os outros valores são calculados.

**R-17**

Faça agora um programa para obter a saída abaixo, sabendo que as raízes da equação $Ax^2 + Bx + C = 0$ são dadas por $X_1 = (-B + (B * B - 4 * A * C) \hat{} 0.5)/2 * A$ e $X_2 = (-B - (B * B - 4 * A * C) \hat{} 0.5)/2 * A$

Saída:

```
RAIZES DA EQUACAO: X^2 + 8*X - 20 = 0

X1 = ____          X2 = ____
```

## II. TABULAÇÃO E FORMATAÇÃO

Comumente queremos determinar com exatidão em que posição as mensagens ou frases devem ser visualizadas no vídeo, que tem 40 colunas e 24 linhas. Para tanto usamos até então o comando **TAB**, associado ao comando **PRINT**.

Formatação horizontal é facilmente conseguida pelo **TAB**, embora este não consiga voltar o cursor para a esquerda, na mesma linha, e para pular linhas são necessários vários **PRINT**s "em branco"; ainda, para espaçar duas palavras, devemos colocar o caractere ♭ (espaço) tantas vezes quantas forem necessárias.



Digite:

```
10  HOME
20  PRINT
30  PRINT
40  PRINT
50  PRINT
60  PRINT
70  PRINT
80  PRINT
90  PRINT
100  PRINT
110  PRINT
120  PRINT  TAB( 20);"SAO PAULO"
```

Faça **RUN** e anote a saída.

**R-18**

### II.1 – HTAB

O comando **HTAB** é um comando independente, isto é, não está associado ao **PRINT** ou a outro comando. Apresenta a forma **HTAB X**, movendo o cursor do mesmo modo que o comando **TAB**, porém sem a limitação de não poder deslocar o cursor para a esquerda.

### II.2 – VTAB

A tela que o computador define no monitor de vídeo apresenta 24 linhas; a linha superior tem número 1, e 24 é o número da última. **VTAB X** move o cursor para a linha designada pelo número X.

Exemplos de **HTAB** e **VTAB**.

Digite o programa abaixo:

```
1Ø  HOME
2Ø  HTAB 2Ø
3Ø  VTAB 1Ø
4Ø  PRINT "SAO PAULO"
```

P4.8

A saída do programa P4.8 será:

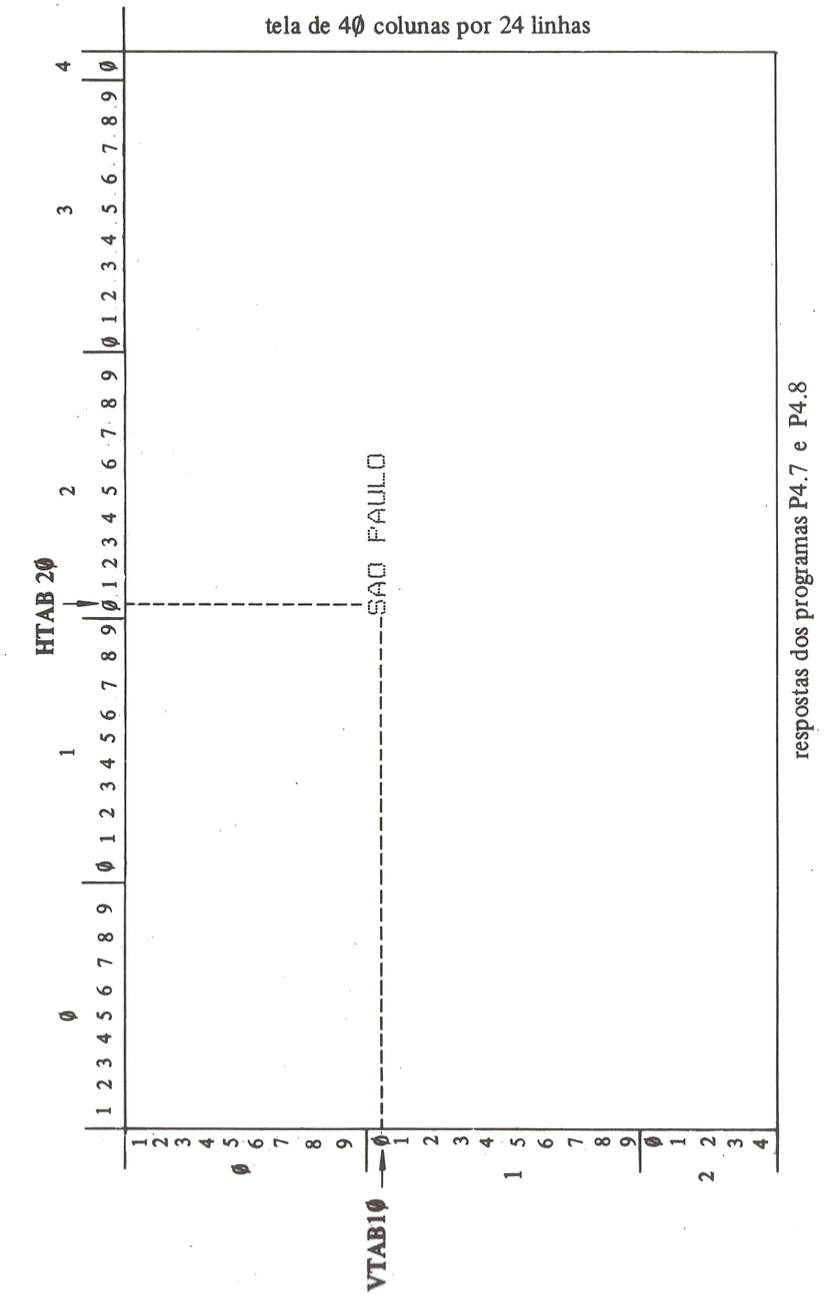
**R-19**

Compare as saídas dos programas P4.7 e P4.8. Qual a sua conclusão?

**R-2Ø**

Na página seguinte, temos uma diagramação da tela do terminal de vídeo em 4Ø colunas e 24 linhas.





## II.3 – SPC

O comando *space*, **SPC (X)**, não é um comando independente e, portanto, deve ser usado "dentro" de um **PRINT**. Esse comando coloca um número de espaços definido pelo número X, entre o último item e o próximo a ser impresso, provocando um deslocamento relativo do cursor. O número X deve estar compreendido entre Ø e 255; esse comando aceita, também, variáveis ou expressões aritméticas para definirem o espaçamento.

Que saída produzirá o comando abaixo?

```
PRINT "SAO";SPC(10);"PAULO"
```

**R-21**

E esta série de instruções?

```
10  HOME
20 A = 40
30 B = (3 * 3) + 1
40  PRINT "OLA"; SPC( A / B);"PESSOAL!"
```

P4.9

**R-22**



## II.4 – Vídeo Inverso

Carregue o programa P4.1Ø e rode-o, a seguir.

```
10  HOME
20  INVERSE
30  HTAB 5
40  VTAB 4
50  PRINT "VIDEO INVERSO"
60  NORMAL
70  VTAB 12
80  HTAB 7
90  PRINT "VIDEO NORMAL"
100  VTAB 20
110  HTAB 28
120  FLASH
130  PRINT "FLASH"
140  NORMAL
150  END
```

P4.1Ø

A resposta de P4.1Ø é:

**R-23**

O que faz o comando **INVERSE** na linha 2Ø?

**R-24**

O que faz o comando **FLASH**?

**R-25**

O que faz o comando **NORMAL**?

**R-26**

O comando **INVERSE** faz com que toda subseqüente saída do computador, e somente a saída via **PRINT**, seja visualizada em caracteres invertidos, isto é, preto no branco (o normal é branco no preto).

O comando **FLASH** faz com que toda subseqüente saída do computador, de forma semelhante ao **INVERSE**, seja visualizada com caracteres piscando, ora preto no branco, ora branco no preto.

**NORMAL** retorna o computador para o modo normal de saída ou visualização.

Digite, a seguir:

```
INVERSE
TESTE
```

Qual a resposta?

**R-27**

Por que **TESTE** apareceu em modo normal e a mensagem **? SYNTAX ERROR** em modo inverso?

**R-28**



Digite:

```
PRINT TAB(1Ø);"ESPACO INVERSO"
```

**R-29**

Para eliminarmos os espaços em **INVERSE**, usamos os comandos **HTAB** e **VTAB**, e para voltar tudo ao normal digite:

```
NORMAL
```

## III. O COMANDO INPUT

Analisemos o programa-exemplo abaixo.

```
1Ø  HOME
2Ø X = 8
3Ø Y = 3
4Ø  PRINT "X = ";X,"Y = ";Y
5Ø  PRINT
6Ø  PRINT "X + Y = ";X + Y,"X - Y = ";X - Y
7Ø  PRINT
8Ø  PRINT "X * Y = ";X * Y,"X / Y = ";X / Y
```

P4.11

Ao digitarmos **RUN**, teremos como resposta:

```
X = 8           Y = 3
X + Y = 11      X - Y = 5
X * Y = 24      X / Y = 2.66666667
```

e, todas as vezes que rodarmos esse programa, teremos sempre a mesma resposta.

Para alterarmos os valores de X ou Y, temos que agir diretamente na memória de programação do computador, reescrevendo as instruções 2Ø ou 3Ø, o que não é nada prático ou dinâmico, além de dificultar o uso do programa por um usuário que não saiba programação.

O comando **INPUT** permite alterarmos ou introduzirmos valores às variáveis já definidas e relacionadas no programa, ou ainda definir novas variáveis. Por meio desse comando, valores numéricos ou alfanuméricos (*strings*) são introduzidos rápida e externamente ao programa, sem ter a necessidade de se alterar linhas ou instruções na programação.

Ao encontrar o comando **INPUT**, o computador pára e apresenta um ponto de interrogação juntamente com o cursor. Nessa ocasião, o operador deve introduzir o dado solicitado e pressionar *RETURN*.

Digite o programa P4.12.

```
5  HOME
10  PRINT  TAB( 10);"QUATRO OPERACOES"
20  PRINT
30  PRINT "QUAL O VALOR DE X?"
40  INPUT X
50  PRINT
60  PRINT "QUAL O VALOR DE Y?"
70  INPUT Y
80  PRINT
90  PRINT "X = ";X,"Y = ";Y
100  PRINT
110  PRINT "X + Y = ";X + Y,"X - Y = ";X - Y
120  PRINT
130  PRINT "X * Y = ";X * Y,"X / Y = ";X / Y
```

P4.12

Carregue o computador com o programa acima e, após o **RUN**, teremos:

```
QUATRO OPERACOES

QUAL O VALOR DE X?

?■
```

cursor

sinal do INPUT

Nessa situação, o computador encontra-se à espera de algum dado de entrada. Por exemplo: digite 18 e pressione *RETURN*.

```
QUAL O VALOR DE Y?

?■
```

Novamente, o computador fica à espera do valor de uma variável, que neste caso é a Y.



Por exemplo: faça Y assumir o valor 6. Coloque, no espaço abaixo, a resposta do programa P4.12.

**R-3Ø**

Assim, toda vez que rodarmos esse programa, o computador ficará à espera das definições das variáveis X e Y, por meio do comando **INPUT**.

Acione, novamente, o programa P4.12 e coloque, na entrada X, *letras* quaisquer. O que o computador respondeu?

**R-31**

Qual seu significado?

**R-32**

Repare que, enquanto o operador não especificar a entrada, o computador continua apresentando? ■, a interrogação e o cursor.

Caso você queira sair dessa pergunta ou do programa nesse ponto, pressione *simultaneamente* as teclas *CTRL* e *C*, e em seguida a tecla *RETURN*.

Na tela, após um "bip", aparecerá a mensagem:

```
BREAK IN 40
```

especificando a linha ou instrução onde ocorreu a interrupção do programa.

Quando damos um **INPUT** após um **PRINT**, o sinal de interrogação do **INPUT** é colocado na linha seguinte à da informação visualizada pelo **PRINT**. Já sabemos como evitar que o computador pule para a próxima linha: basta colocar um ponto-e-vírgula no fim do comando **PRINT**.

Reescreva as linhas 3Ø e 6Ø do programa P4.12, suprimindo os "?" e colocando, após as aspas finais, um ponto-e-vírgula:

```
30  PRINT "QUAL O VALOR DE X";
60  PRINT "QUAL O VALOR DE Y";
```

Rode o novo P4.12. Qual foi o efeito da alteração?

**R-33**

Tenha em mente que o ponto de interrogação na mensagem

```
QUAL O VALOR DE X?
```

foi colocado, não pelo **PRINT**, mas pelo **INPUT**, automaticamente, seguido pelo cursor para avisar que o computador está esperando por dados.

## III.1 – INPUT e Strings

O comando **INPUT** pode ser usado, também, para manusear *strings*.
Veja o exemplo P4.13:

```
10  HOME
20  PRINT "QUAL A PRIMEIRA FRASE";
30  INPUT A$
40  PRINT
50  PRINT "QUAL A SEGUNDA FRASE";
60  INPUT B$
70  HOME
80  PRINT A$ + B$
```

P4.13

Faça a primeira frase ser

```
O BRASIL E' TRICAMPEAO
```

e a segunda frase

```
DE FUTEBOL
```



No monitor de vídeo, aparecerá:

**R-34**

Rode, outra vez, o programa P4.13 e faça a primeira frase ser

```
O GATO SUBIU
```

e a segunda

```
NO TELHADO
```

e o computador apresentará:

**R-35**

Na segunda sentença, os termos **SUBIU** e **NO** ficaram juntos; isto porque não foi deixado um espaço.
Na primeira sentença ocorreu esse mesmo fato? Por quê?

**R-36**

Acione, novamente, o programa P4.13 e coloque **111** como sendo a primeira frase e 222 como sendo a segunda.
Qual o resultado?

**R-37**

Por que a resposta não foi **333**, uma vez que a instrução 8Ø é **PRINT A $ + B $**?

**R-38**

Qual a diferença entre valor numérico e valor alfanumérico ou *string*?

**R-39**

## III.2 – Mais INPUTs num INPUT

O comando **INPUT** aceita simultaneamente duas ou mais variáveis, numéricos ou *string*, desde que estejam separadas por vírgulas.

Analise o exemplo P4.14.

```
5  HOME
1Ø  PRINT "ESCREVA DOIS NUMEROS"
2Ø  PRINT "(SEPARADOS POR VIRGULA)."
3Ø  INPUT A,B
4Ø  PRINT
5Ø  PRINT "OS NUMEROS SAO A = ";A,"B = ";B
6Ø  PRINT
7Ø  PRINT "A SOMA E' ";A + B
8Ø  PRINT
9Ø  PRINT "... E O PRODUTO E' ";A * B
```

P4.14

**RUN**ize o programa acima.

```
ESCREVA DOIS NUMEROS

?■
```

← à espera dos dados



Faça, por exemplo, os números serem 6 e 2. Digite 6,2 e, em seguida, pressione *RETURN*.

Lembre-se de que os dados de entrada são separados por uma vírgula; desse modo, a resposta do computador é:

**R-4Ø**

Acione, novamente, o programa P4.14.

```
ESCREVA DOIS NUMEROS

?■
```

Digite 6 e, em seguida, pressione *RETURN*. O que surge na tela do monitor de vídeo?

**R-41**

Qual o significado da mensagem ?? ?

**R-42**

Coloque uma palavra como sendo o segundo número. Qual a mensagem do computador?

**R-43**

Qual o seu significado?

**R-44**

## III.3 – Um PRINT dentro de um INPUT

Quando definimos, por meio dos comandos **PRINT** e **INPUT**, a entrada de uma variável, as instruções normalmente têm este aspecto:

```
40  PRINT "QUAL O VALOR DO RAIO?"
50  INPUT R
```

O comando **INPUT** aceita, simultaneamente, a mensagem de explicação da variável e a própria variável, desde que estejam separadas por ponto-e-vírgula.

Nessa proposta, o cursor surge, sem a interrogação característica, logo após a mensagem, pois a interrupção do programa é feita, obviamente, nesse ponto, sem que o BASIC vá para a próxima linha. Analise essa idéia com o programa P4.15.

```
10  HOME
20  PRINT  TAB( 5);"COMPRIMENTO DA CIRCUNFERENCIA"
30  PRINT  TAB( 20);"E"
40  PRINT  TAB( 5);"AREA DO CIRCULO QUE ELA DEFINE"
50  PRINT
60  INPUT "QUAL O RAIO? ";R
70 C = 2 * 3.14 * R
80 A = 3.14 * R * R
90  PRINT
100  PRINT
110  PRINT "O COMPRIMENTO E' ";C
120  PRINT "E"
130  PRINT "A AREA E' ";A
```

P4.15

O **INPUT** com **PRINT** ainda admite mais de uma variável, desde que estejam separadas por vírgulas; as variáveis serem numéricas ou *strings*, ou uma combinação dos dois tipos.



**RUN**ize, como exemplo, o programa abaixo.

```
10  HOME
20  INPUT "ESCREVA SEU NOME E IDADE ";N$,I
30 M = I * 12
40 D = I * 365
50  HOME
60  PRINT  TAB( 12);N$
70  PRINT
80  PRINT "VOCE JA' VIVEU ";M;" MESES"
90  PRINT
100  PRINT "OU JA' VIU O SOL NASCER ";D;" VEZES"
```

P4.16

Substitua a linha 20 por

```
20 INPUT "";N$,I
```

Observe que o ponto de interrogação não aparece, como aconteceria se fizéssemos

```
20 INPUT N$,I
```

# IV. GET

Podemos dizer, em primeira aproximação, que o comando **GET** é um **INPUT**, automático e dinâmico, de um único caractere ou algarismo.

Para entendermos melhor o comando **GET**, carregue o computador com o programa P4.17.

```
10  HOME
20  PRINT
30  PRINT  TAB( 3);"DIGITE O SEU NOME";
40  GET A$
50  VTAB 10
60  HTAB 6
70  PRINT "A PRIMEIRA LETRA DO SEU NOME E': "
80  FLASH
90  VTAB 20
100  HTAB 20
110  PRINT A$
120  NORMAL
130  END
```

P4.17

Rode o programa P4.17 e responda:

Por que você não conseguiu digitar o seu nome completo?

**R-45**

O cursor aparece logo após o **PRINT** da linha 3Ø. Por quê?

**R-46**

Por que não é mostrada, no lugar onde estava o cursor, a letra que você pressionou?

**R-47**

O **GET** pode também ser usado junto com uma variável numérica, mas isso raramente é feito, pelo fato do Applesoft responder com uma mensagem de erro quando não é pressionada uma tecla numérica (Ø a 9 ou os símbolos +, -, E) em resposta ao **GET**. Mesmo assim, os símbolos +, - e E retornam o valor zero na variável. Normalmente, esse problema é resolvido colocando-se uma variável alfanumérica no **GET**, transformando depois o *string* obtido num número, através de funções que ainda serão vistas.



## V. EXERCÍCIOS PROPOSTOS

4.1 O comando **INPUT** diz ao computador para esperar uma informação a partir do teclado. Diga, para cada uma das situações abaixo, a exata informação esperada:

a) **INPUT A**
b) **INPUT D $**
c) **INPUT A, AC**
d) **INPUT A $, B $**
e) **INPUT D $, Z $, A, BC**

4.2 O que o computador responderá aos comandos abaixo?

a) **PRINT A**
b) **PRINT A $**
c) **PRINT**

4.3 Suponha que um determinado programa apresenta as linhas

```
50 INPUT X$
60 PRINT X$
```

Se o operador digitar as informações abaixo, em resposta à linha 5Ø, o que ele imprimirá na linha 6Ø?

a) **PAI**
b) **"PAI"**
c) **1982**
d) **"1982"**
e) **SILVA, JOSE' DA**
f) **"SILVA, JOSE' DA"**

4.4 Descreva o que este programa faz:

```
10  HOME
20  PRINT "ESCREVA SEU NOME"
30  INPUT N$
40  HOME
50  PRINT "ESCREVA A DATA DE HOJE (D/M/A)
60  INPUT D$
70  HOME
80  PRINT "PREZADO ";N$; SPC( 10);D$
90  PRINT
100  PRINT "PRAZER EM CONHECE-LO, ";N$
110  PRINT
120  PRINT "LEMBRAREI SEMPRE DESTE DIA ";D$
130  PRINT
140  PRINT "TENHA UM BOM DIA!"
150  PRINT  TAB( 25);"SINCERAMENTE,"
160  PRINT  TAB( 35);"APPLE"
```

4.5 Faça o fluxograma e escreva o programa que converte temperaturas, dadas em Celsius, para Fahrenheit, usando a fórmula:

$$F = (95)\,C + 32$$

A saída deve ser

```
CONVERSAO DE TEMPERATURAS C - F

C                    F

tC                   tF
```

4.6 Escreva um programa para calcular a média anual de um aluno. O programa deve solicitar, separadamente, as médias bimestrais e deve ter a saída abaixo. A formatação deve ser por meio de **TAB**. Os pesos respectivos de cada bimestre, na média final, são de $\frac{1}{8}$, $\frac{2}{8}$, $\frac{2}{8}$ e $\frac{3}{8}$. A média, portanto, é calculada pela expressão:

$$MA = (B_1 + 2 * B_2 + 2 * B_3 + 3 * B_4)/8$$

```
                MEDIA ANUAL

B1        B2        B3        B4        MA

___       ___       ___       ___       ____
```

4.7 Faça um programa para resolver uma equação do 2º grau. Faça, também, o fluxograma e admita que o discriminante seja sempre maior que zero. O computador deve solicitar os coeficientes a, b e c e deve ter uma saída igual à saída abaixo e formatada por meio de **VTAB** e **HTAB**.

```
RAIZES DA EQUACAO AX^2 + BX + C = Ø

DELTA = ____      X1 = ____          X2 = ____
```

4.8 Construa o fluxograma e a codificação BASIC de um programa que calcula o salário de um funcionário com entradas e saídas definidas abaixo. A tela de entrada deve ser impressa por inteiro, para que depois o cursor seja posicionado para cada entrada individual.



Tela de entrada:

```
XYZ S/A

NOME DO
FUNCIONARIO: ■

CARGO:

                    SAL/HORA:

                    HORAS TRABALHADAS:


                DIGITE OS DADOS
```

Tela de saída:

```
XYZ S/A


NOME: ___

CARGO: ___

                      SALARIO BRUTO: ___

                    DESCONTOS (12%): ___

                    ---------------------

                    SALARIO LIQUIDO: ___
]■
```

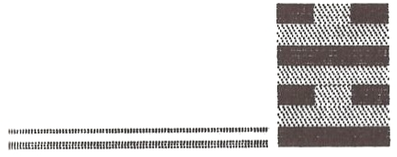
4.9 Refaça o programa do problema 4.4 usando **HTAB**, **VTAB** e **SPC**.

## VI. RAXAKUKA

4.1 Paulo e Roberto estavam indo de bicicleta de sua cidade para uma outra, 20 km distante. Após viajarem 4 km, a bicicleta de Roberto quebrou. Eles queriam chegar rápido, juntos, andando o mínimo a pé. A pé, podiam andar à velocidade de 4 km/hora e de bicicleta à de 8 km/hora. Combinando andar a pé e de bicicleta, como atingiriam o seu objetivo, para cobrir os 16 km restantes?

4.2 Três homens querem atravessar um rio. O barco que possuem tem a capacidade máxima de 150 quilos. Eles pesam 50, 75 e 120 quilos. Como podem atravessar, sem afundar o barco?

4.3 Todos os sábados, à tarde, encontro-me com uma amiga. A hora prevista para o encontro é meio-dia. Da primeira vez ela chegou às 12:30 hs.; no sábado seguinte às 13:20 hs.; no outro sábado às 14:30 hs. e, no outro, ainda às 16:00 hs. A que horas chegará na próxima semana?



# Desvios e decisões

*"Sou um indivíduo que, como filho de sua época e equipado com os métodos lógicos e experimentais de seu tempo, na verdade apenas conseguiu erguer-se sobre os ombros de seus predecessores, vendo talvez um pouco além do que haviam visto antes dele..."*

*Walther Nernst*
*(1864-1941)*

# I. O COMANDO GOTO

Esse comando é usado para provocar desvios, durante a execução, para uma determinada linha ou instrução do programa. Esse desvio é feito *incondicionalmente*, isto é, o trajeto de execução é desviado sem ser previamente estabelecida uma condição para tal. O número da linha, para a qual a execução é desviada, é denominada argumento do **GOTO**.

Carregue o computador com o programa P5.1:

```
10  HOME
20  PRINT " APPLE ";
30  GOTO 20
```

P5.1

**RUN**ize esse programa.

O que você está observando?

**R-1**

Qual o procedimento para parar a execução do programa? (Lembra-se de como saíamos do **INPUT**?)

**R-2**

Se você não se lembra, segure a tecla *CTRL* enquanto você aperta com outro dedo a tecla *C*. O *RETURN*, neste caso, não é necessário.

Qual a mensagem que aparece na tela? Qual o seu significado?

**R-3**



Junto com o comando **GOTO**, os comandos **STOP**, **END** e **CONT** são automaticamente descutidos. Analise o programa P5.2:

```
5  REM  VERIFICACAO DO COMANDO GOTO
10  HOME
20  HTAB 6
30  PRINT "VERIFICACAO DO COMANDO GOTO"
40  GOTO 160
50  INVERSE
60  PRINT "VOLTEI A 50 E 60"
70  GOTO 200
80  SPEED= 0
90  PRINT "ESTOU QUASE NO FIM"
100  VTAB 20
110  HTAB 25
120  NORMAL
130  SPEED= 255
140  PRINT "LINHA 140: FIM"
150  END
160  PRINT
170  PRINT "CHEGUEI NA 170"
180  PRINT
190  GOTO 50
200  FLASH
210  PRINT "AGORA NA 210"
220  GOTO 80
```

P5.2

Rode o programa P5.2 e escreva a resposta obtida:

**R-4**

O que faz o comando **SPEED**?

**R-5**

O comando **SPEED** regula a velocidade de *visualização* dos caracteres no vídeo. A menor velocidade é dada por **SPEED** = **0** e a maior por **SPEED** = **255**. Valores fora desses limites geram a mensagem

```
?ILLEGAL QUANTITY ERROR
```

Em vez do comando **END** na instrução 15Ø, coloque o comando **STOP**, digitando

```
15Ø STOP
```

Rode o programa P5.2 modificado. Qual a resposta obtida?

**R-6**

Compare as respostas obtidas pelas duas versões do P5.2. Qual a diferença entre os comandos **STOP** e **END**?

**R-7**

Digite **CONT**. Qual a resposta no monitor de vídeo?

**R-8**

Se a execução de um programa for interrompida por um **STOP**, **END** ou *CTRL-C*, ou mesmo por *RESET*, o comando **CONT**inue reinicia a execução do programa a partir da instrução que segue a última executada.

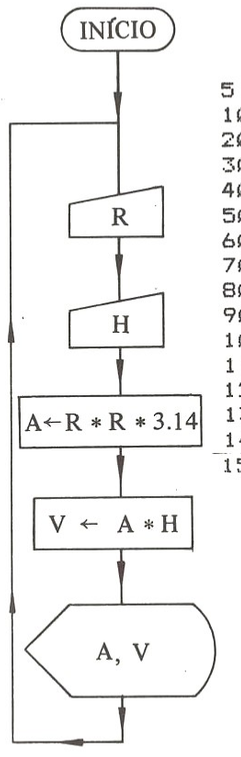
Qual a diferença entre os comandos **CONT** e **RUN**?

**R-9**



Para cálculos repetitivos, o comando **GOTO** tem uma utilidade incrível. Como exemplo, vamos analisar o programa P5.3, que calcula a área da base de cilindros quaisquer e seus respectivos volumes.

Fluxograma




Codificação

```
5  REM  AREA E VOLUME DE CILINDROS
10  HOME
20  PRINT  TAB( 8);"AREA E VOLUME DE CILINDROS"
30  PRINT
40  INPUT "QUAL O RAIO? ";R
50  INPUT "QUAL A ALTURA? ";H
60 A = R * R * 3.14
70 V = A * H
80  HOME
90  PRINT
100  PRINT  TAB( 8);"AREA E VOLUME DE CILINDROS"
110  PRINT
120  PRINT "RAIO = ";R,"AREA DA BASE = ";A
130  PRINT
140  PRINT "ALTURA = ";H,"VOLUME = ";V
150  GOTO 30
```

P5.3

Rode o programa P5.3 e complete o quadro abaixo.

| RAIO | ALTURA | ÁREA | VOLUME |
|---|---|---|---|
| 1.0 | 2.0 | | |
| 2.0 | 10.0 | | |
| 1.38 | 17.14 | | |
| 4.5 | 6.9 | | |

Modifique as instruções 6Ø e 7Ø do P5.3 para* :

```
6Ø A = ( INT (R * R * 3.14 * 1ØØ)) / 1ØØ
7Ø V = ( INT (A * H * 1ØØ)) / 1ØØ
```

Após as alterações acima, rode o novo P5.3 e complete o quadro abaixo.

| RAIO | ALTURA | ÁREA | VOLUME |
|---|---|---|---|
| 1.0 | 2.0 | | |
| 13.389 | 147.3519 | | |
| 0.38 | 10.71574 | | |

Quais as diferenças entre as respostas dos programas P5.3 original e modificado?

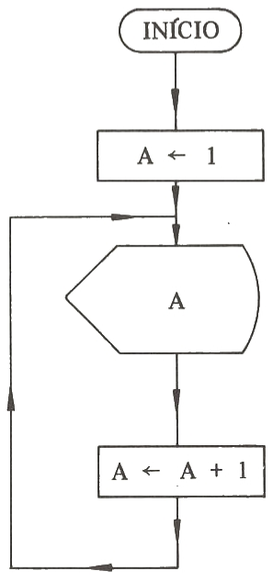
**R-1Ø**

(*) As novas instruções 6Ø e 7Ø do P5.3 mostram um modo de se obter respostas, com até dois algarismos depois da vírgula e sem arredondamento, por meio da função **INT**, que será discutida adiante com mais detalhes.



Com o comando **GOTO**, podemos criar um contador simples. Analisemos o programa P5.4.

Fluxograma




Codificação

```
1Ø  HOME
2Ø  LET A = 1
3Ø  PRINT A
4Ø A = A + 1
5Ø  GOTO 3Ø
```

P5.4

Carregue o computador com o exemplo P5.4 e analise o resultado desse programa.

Por que não pára nunca?

**R-11**

Lembre-se de que, se quisermos interromper o processamento, devemos acionar *CTRL-C*.

O computador caiu num *loop incondicional* e contínuo, isto é, o programa produziu um ciclo repetitivo das operações envolvidas e não existe nenhum comando ou instrução, dentro desse ciclo, que faça ou permita a parada do processamento.

Qual a modificação que faríamos, para imprimir os números em linha?

**R-12**

Vamos introduzir, por enquanto, um truque. Adiante discutiremos como age o comando **FOR ... NEXT**.

Faça:

```
35  FOR I = 1 TO 5ØØ: NEXT I
```

Liste o novo P5.4, anotando-o abaixo.

```
LIST
```

**R-13**

Rode o programa. Qual a modificação na resposta do computador?

**R-14**

Usando **35 SPEED = Ø**, obteríamos o mesmo efeito? Tente fazê-lo.

**R-15**



Com o programa P5.4, que é denominado contador, podemos incrementar ou decrementar uma variável, de uma constante qualquer; porém, esse contador não tem fim. É necessário fazê-lo parar em algum ponto c; para isso, o computador deverá tomar uma decisão.

## II. A HORA DA VERDADE

Antes de poder tomar uma decisão, uma pessoa deve ser capaz de distinguir entre o verdadeiro e o falso, o certo e o errado. No BASIC, como na matemática, podemos relacionar duas expressões numéricas através dos chamados *operadores relativos*, o que pode resultar numa afirmação verdadeira ou falsa. Por exemplo, como 9 é menor que 15, a expressão **9 < 15** é uma verdade. O computador representa uma verdade pelo número **1**. Qual será a saída do comando abaixo? Tente-o no computador.

```
PRINT 9<15
```

**R-16**

Analogamente, o valor zero representa uma afirmação falsa. Faça:

```
PRINT 8=1Ø
```

**R-17**

Os símbolos usados para relacionar dois números ou expressões numéricas no **BASIC** do APPLE são ligeiramente diferentes daqueles usados na matemática. Veja a tabela 1.

Tabela 1

| | |
|---|---|
| = | é igual a |
| > | é maior que |
| < | é menor que |
| > = | é maior ou igual a (≥) |
| < = | é menor ou igual a (≤) |
| < > | é diferente de (≠) |

Tente prever o resultado de cada afirmação, testando-as depois com o **PRINT**:

```
8 = 8
```

**R-18**

```
6 > 7
```

**R-19**

```
1 > 1
```

**R-2Ø**



```
(-3) >= (-3)
```

**R-21**

```
(-3) > 4
```

**R-22**

```
5 <> 5
```

**R-23**

```
123 = 231
```

**R-24**

```
741 > 99
```

**R-25**

```
(5+2) >= 6
```

**R-26**

```
(8*3) <= (25-1)
```

**R-27**

```
Ø = (4-4/2)
```

**R-28**

```
(-125) <> 125
```

Pense um pouco e faça o mesmo com os comandos:

```
PRINT 1=1=1
```

**R-29**

```
PRINT 2=2=2
```

**R-3Ø**



```
PRINT Ø=1=Ø
```

**R-31**

```
PRINT 6<5=Ø
```

**R-32**

Como você explica as saídas obtidas?

**R-33**

Se você teve alguma dificuldade com essas expressões, acompanhe a análise do segundo exemplo. O BASIC, para efetuar uma operação dessa, divide a expressão em partes: primeiro, ele faz 2 = 2, uma verdade, que dá 1 como resultado; a seguir, esse 1 é igualado ao 2, resultando no valor zero (falso). Por isso, a expressão 2 = 2 = 2, que à primeira vista parece verdadeira, resulta ser falsa.

Analise, da mesma forma, a última expressão, 6 < 5 = Ø.

**R-34**

Lembre-se que as expressões podem conter (ou serem constituídas apenas por) variáveis.

Desse jeito, a seqüência:

```
X=4

Y=2

PRINT X*Y < (5-Y)*X
```

devolve o valor **1**, verdade.

Continuando com nosso estudo dos operadores lógicos no BASIC, veremos três operadores que trabalham não com os números em si, como aconteceu até agora, mas com os "estados" verdadeiro e falso, os valores 1 e Ø.

O primeiro desses operadores é o **NOT**, que significa *não* em inglês. Ele inverte o resultado de uma expressão, ou seja, retorna "falso" quando a expressão for verdadeira, ou "verdadeiro" quando for falsa. Isto porque o que *não* é verdadeiro é falso, e o que *não* é falso resulta ser verdadeiro. Efetue, usando o **PRINT**:

```
NOT Ø
```

**R-35**

```
NOT 1
```

**R-36**

```
NOT (5=5)
```

**R-37**



```
NOT (6>66)
```

**R-38**

```
NOT (1=2)
```

**R-39**

```
NOT 1 = 2
```

**R-4Ø**

Você deve ter percebido, pelos últimos dois exemplos, que deve haver alguma hierarquia entre os operadores lógicos, como já vimos que há entre os operadores aritméticos. Essa hierarquia está resumida no final da unidade.

O **AND** (*e*, em inglês) relaciona duas afirmações, ambas verdadeiras, ambas falsas, ou uma verdadeira e uma falsa. Se tivermos a expressão **A AND B**, sendo **A** e **B** valores *verdadeiro* ou *falso* (1 ou Ø), o resultado da operação **AND** está resumido na tabela 2.

Tabela 2

| A | B | A AND B |
|---|---|---|
| Ø | Ø | Ø |
| Ø | 1 | Ø |
| 1 | Ø | Ø |
| 1 | 1 | 1 |

Vemos que a saída será verdadeira somente se a primeira E a segunda afirmação forem verdadeiras.

Antes de prosseguir com exemplos, veja a tabela 3 para o operador **OR** (*ou*, em inglês):

Tabela 3

| A | B | A OR B |
|---|---|---|
| Ø | Ø | Ø |
| Ø | 1 | 1 |
| 1 | Ø | 1 |
| 1 | 1 | 1 |

Portanto, a saída do operador **OR** será verdadeira quando a primeira for verdadeira OU a segunda for verdadeira, sendo falsa apenas quando ambas as afirmações forem falsas. Tente prever o resultado das seguintes afirmações, testando-as depois com **PRINT**s:

```
1 AND 1
```

**R-41**

```
Ø AND 1
```

**R-42**

```
1 AND Ø
```

**R-43**



```
Ø AND Ø
```

**R-44**

```
(2>=1) AND (7=7)
```

**R-45**

```
(8<9) AND Ø
```

**R-46**

```
(5+3=53) AND (23=30-7)
```

**R-47**

```
1 OR 1
```

**R-48**

```
Ø OR 1
```

**R-49**

```
1 OR Ø
```

**R-5Ø**

```
Ø OR Ø
```

**R-51**

```
(2>=2) OR Ø
```

**R-52**

```
((4<3) OR (324>42)) AND (NOT (2Ø<4*3))
```

**R-53**



```
(Ø AND 1) OR NOT (Ø AND 1)
```

**R-54**

```
(1 OR 1) AND Ø
```

**R-55**

```
1 OR 1 AND Ø
```

**R-56**

Pelos últimos dois exemplos vemos, mais uma vez, que existe uma hierarquia entre os operadores lógicos, como nos operadores aritméticos. Como ambos os tipos de operadores trabalham sobre números, essa hierarquia envolve todos. A tabela 2 de hierarquia vista no capítulo 1 (página 24) deve, portanto, ser completada para incluir os operadores lógicos. Os parênteses têm a maior prioridade, enquanto o operador **OR** tem a menor. Veja tabela 4.

Tabela 4

| |
|---|
| ( ) |
| + - **NOT** (operando sobre um único número) |
| ^ |
| * / |
| + - |
| < = < > > = = < > |
| **AND** |
| **OR** |

Mesmo conhecendo a hierarquia entre os operadores, é sempre uma boa prática colocar parênteses nas expressões um pouco mais complexas, especialmente quando estas envolvem **AND**, **OR** ou **NOT**.

Faça, agora:

```
PRINT NOT 528
```

**R-57**

```
PRINT NOT 2
```

**R-58**

Percebemos que o computador considera como "verdadeiro" qualquer valor *diferente de zero*, considerando "falso" apenas o valor zero.

Da mesma forma que fizemos com números, podemos usar operadores lógicos com cadeias de caracteres. Neste caso, os símbolos relativos ganham novo significado. Veja a tabela 5.

Tabela 5

| | |
|---|---|
| = | é idêntico a |
| > | sucede alfabeticamente (vem depois de) |
| < | precede alfabeticamente (vem antes de) |
| > = | sucede alfabeticamente ou é idêntico a |
| < = | precede alfabeticamente ou é idêntico a |
| < > | não é idêntico a (é diferente de) |



Fazendo **A$ = "BRASIL"**, use o **PRINT** para verificar os resultados das expressões:

```
A$ = "BRASIL"
```

**R-59**

```
"BRASILIA" = A$
```

**R-60**

```
"BRAS" < A$
```

**R-61**

```
A$ <> A$
```

**R-62**

```
A$ >= A$
```

**R-63**

```
"A" < "B"
```

**R-64**

```
"AAA" >= "A"
```

**R-65**

```
"BRASIL" > "CANADA" OR A$ = "AUSTRIA"
```

**R-66**

```
("XYZ" < "ZYX") AND ("B" <> "BE")
```

**R-67**

Devemos, porém, ter muito cuidado para não escrevermos expressões do tipo

```
A$ > "BRASIL" = A$
```

processadas pelo computador como

```
Ø = A$
```

e resultando num erro do tipo

```
?TYPE MISMATCH ERROR
```

já que não tem qualquer sentido a comparação de um *string* com um número.



Por último, perceba que a expressão

```
"X" >= "Z"
```

é, mesmo que não o pareça, uma expressão *numérica*. Lembre-se de que uma expressão é classificada como numérica ou *string* pela sua *saída* e não pelos elementos que a compõem.

## III. DECISÕES – O COMANDO IF ... THEN

No BASIC do APPLE, Applesoft o comando **IF ... THEN** nos permite, durante a execução de um programa, seguir um entre dois caminhos. Ele tem a forma geral:

**IF** *expr. num.* **THEN** *comando*

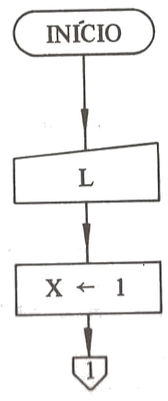
O *comando* que segue o **THEN** pode ou não ser executado pelo BASIC, dependendo do valor da expressão numérica entre o **IF** e o **THEN**. Se esta for zero (falso), tudo o que segue o **THEN** é ignorado por completo, apenas sendo executado o *comando* quando *expr. num.* for diferente de zero (verdadeiro). Portanto, no comando

```
IF X=Y THEN PRINT "IGUAL"
```

a mensagem somente aparecerá se a expressão **X = Y** for verdadeira, isto é, se **X** for igual a **Y**. Tente esse comando direto para alguns valores de **X** e **Y**.

A seguir, digite o seguinte programa:

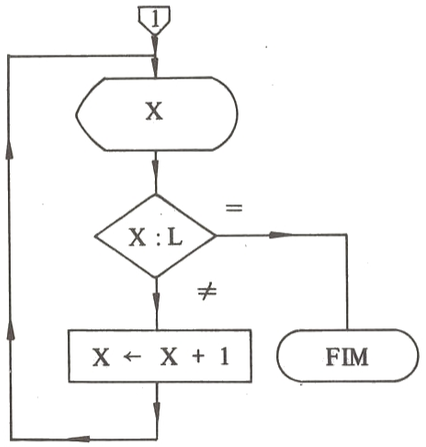
Fluxograma




Codificação

```
5  REM  CONTADOR COM LIMITE
10  HOME
20  PRINT  TAB( 16);"CONTADOR"
30  PRINT
40  INPUT "QUAL O LIMITE? ";L
50 X = 1
60  PRINT
65  REM  INICIO DO LOOP
70  PRINT X,
75  REM  ACABOU?
80  IF X = L THEN  END
85  REM  NAO, INCREMENTAR E VOLTAR
90 X = X + 1
100  GOTO 70
```

P5.5




Rode o programa P5.5, fazendo L = 1Ø.
Modifique o programa P5.5, inserindo as instruções

```
8Ø  IF X = L THEN  GOTO 11Ø
11Ø  PRINT
12Ø  PRINT "ACABEI DE CONTAR DE 1 ATE' ";L;"  UFA!"
```

Liste o programa e **RUN**ize-o, fazendo **L = 5Ø**.

**R-68**

Enquanto o **GOTO** da linha 1ØØ é chamado desvio incondicional, na linha 8Ø ele é um *desvio condicional*. Por quê?

**R-69**



Rode o P5.5 novamente, atribuindo a **L** o valor 12.35. Qual o resultado?

**R-7Ø**

Lembre-se que *CTRL-C* interrompe a execução do programa. Por que ele não pára sozinho ao passar pelo 12.35?

**R-71**

Na verdade, o computador fez exatamente o que nós lhe pedimos: só deveria parar quando **X** fosse *exatamente igual* à 12.35, o que nunca iria acontecer nesse nosso contador unitário. O que realmente queríamos era que o computador parasse assim que **X** ultrapassasse o limite **L**, ou seja, quando **X** fosse *maior ou igual* a **L**.
Modifiquemos, portanto, a linha 8Ø para:

```
8Ø  IF X >= L THEN GOTO 11Ø
```

Digitando **RUN**, teste o programa para **L = 9.4**:

**R-72**

O que segue o **THEN** pode ser qualquer comando BASIC. Veja os seguintes exemplos:

```
IF A > 1Ø THEN PRINT A
IF A = B THEN STOP
IF X - Y > R^2 THEN X = 3.14
IF M <= Ø THEN GOTO 1ØØ
IF B*B - 4*A*C >= 5 - Z^3 THEN N = (5*PI)/Z
IF R$="SIM" AND X<>Y THEN INPUT "PRESSIONE <RETURN>";Q$
```

Analise a instrução abaixo.

```
IF A > Ø THEN IF A <= 5 THEN PRINT "ESTA' CONTIDO"
```

O que ela faz?

**R-73**

Como a representaríamos em fluxograma?

**R-74**

Analise, a seguir, o programa P5.6. O que ele faz? Como exercício, faça o seu fluxograma.

Codificação

```
5  REM  ESTRELAS
1Ø  HOME
2Ø  INPUT "QUANTAS ESTRELAS VOCE QUER? ";N
3Ø  PRINT
4Ø X = 1
5Ø  PRINT " * ";
6Ø  IF X = N THEN 1ØØ
7Ø X = X + 1
8Ø  GOTO 5Ø
9Ø  REM  TERMINADO
1ØØ  PRINT
11Ø  PRINT "MAIS ESTRELAS (S/N)? ";
12Ø  GET SN$
13Ø  IF SN$ = "S" THEN 1Ø
```

P5.6

Rode o programa para verificar se a sua previsão estava correta.

Note o detalhe das linhas 6Ø e 13Ø, onde o **GOTO**, que deveria estar logo após o **THEN**, não foi colocado, *mas está implícito*. É o único comando que apresenta esse tipo de abreviação, já que o **GOTO** é o mais usado no **IF ... THEN**.



Carregue e rode o programa P5.7 no computador.

Codificação

```
1Ø  HOME
2Ø  PRINT
3Ø  PRINT  TAB( 15);"PALAVRAS"
4Ø  PRINT
5Ø  INPUT "ESCREVA UMA PALAVRA QUALQUER ";P1$
6Ø  PRINT
7Ø  INPUT "ESCREVA OUTRA PALAVRA ";P2$
8Ø  IF P1$ < P2$ THEN  GOTO 11Ø
9Ø  IF P1$ > P2$ THEN 14Ø
1ØØ  IF P1$ = P2$ THEN 17Ø
11Ø  PRINT
12Ø  PRINT P1$;" ANTECEDE ";P2$
13Ø  GOTO 5ØØ
14Ø  PRINT
15Ø  PRINT P1$;" SUCEDE ";P2$
16Ø  GOTO 5ØØ
17Ø  PRINT
18Ø  PRINT P1$;" E ";P2$;" SAO PALAVRAS IDENTICAS"
5ØØ  PRINT
51Ø  INPUT "VOCE CONTINUA (S/N)? ";R1$
52Ø  IF R1$ = "S" THEN  GOTO 1Ø
53Ø  HOME
54Ø  END
```

P5.7

Analise e discuta as linhas 8Ø, 9Ø e 1ØØ.

**R-75**

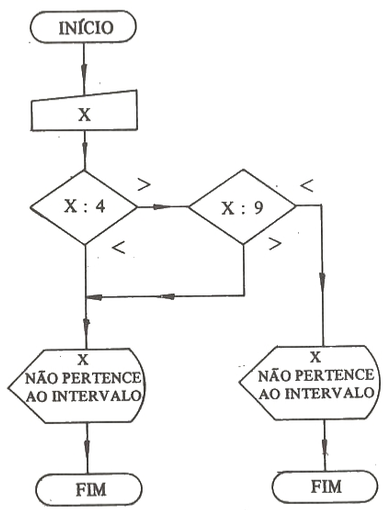
O que aconteceria se eliminássemos a linha 8Ø? Por quê?

**R-76**

Tirando a linha 8Ø não alteramos em nada o funcionamento do programa. Se **P1$** não for maior que ou igual a **P2$** (linhas 9Ø e 1ØØ), a execução "flui" naturalmente para a linha 11Ø.

Finalizando o capítulo, veja como os operadores **AND** e **OR** podem simplificar as comparações com **IF... THEN**. Analise o programa P5.8.

Fluxograma




Codificação

```
1Ø  HOME
2Ø  INPUT "DIGITE UM NUMERO: ";X
3Ø  IF X < 4 THEN  GOTO 7Ø
4Ø  IF X > 9 THEN  GOTO 7Ø
5Ø  PRINT X;" PERTENCE AO
    INTERVALO 4-9"
6Ø  GOTO 8Ø
7Ø  PRINT X;" NAO PERTENCE
    AO INTERVALO 4-9"
8Ø  END
```

P5.8

O programa poderia ser reescrito como:

```
1Ø  HOME
2Ø  INPUT "DIGITE UM NUMERO: ";X
3Ø  IF X < 4 OR X > 9 THEN 7Ø
5Ø  PRINT X;" PERTENCE AO INTERVALO 4-9"
6Ø  GOTO 8Ø
7Ø  PRINT X;" NAO PERTENCE AO INTERVALO 4-9"
8Ø  END
```

P5.9



ou ainda:

```
1Ø  HOME
2Ø  INPUT "DIGITE UM NUMERO: ";X
3Ø  PRINT X;
4Ø  IF X < 4 OR X > 9 THEN  PRINT " NAO";
5Ø  PRINT " PERTENCE AO INTERVALO 4-9"
6Ø  END
```

P5.1Ø

Experimente com suas variações sobre os programas deste capítulo.

## IV. EXERCÍCIOS PROPOSTOS

5.1 Estruture um contador que apresente as perguntas:

```
VALOR INICIAL? ■

VALOR FINAL?

INCREMENTO?
```

e apresente os resultados em linha (os números devem estar separados).

5.2 Construa o fluxograma e esquematize o programa para se calcular $X^2$; $X^{1/2}$; $X^3$ e $X^{1/3}$, desde zero até um valor a ser determinado pelo usuário.

A saída deve ser:

```
X       X^2     X^(1/2)     X^3     X^(1/3)
Ø       Ø       Ø           Ø       Ø
1       1       1           1       1
2       4       1.4142      8       1.2599
.       .       .           .       .
:       :       :           :       :
```

5.3 Estruture um programa para tabelar a função y = 2x + 3, usando o contador do problema 5.1; a saída deve ser:

```
TABELA DA FUNCAO: Y = 2X + 3

X               Y

Ø               3
1               5
2               7
.               .
:               :
```

5.4 Faça o estudo completo para resolver uma equação do 2º grau $ax^2 + bx + c = 0$. O computador deve "perguntar" os valores de a, b e c, além de enviar mensagens sobre o discriminante e as raízes.

5.5 Faça o programa para a determinação da soma dos n primeiros termos da série: $1 + \frac{1}{2} + \frac{1}{3} + \frac{1}{4} + \ldots$

5.6 Um determinado imposto, para pessoas que recebem acima de ..... Cr$ 150.000,00, é 20%. Para pessoas que recebem desde Cr$ 80.000,00 até ..... Cr$ 150.000,00, o imposto é 15%. Abaixo de Cr$ 80.000,00, o imposto é 8% e isentos são os recebimentos até Cr$ 40.000,00. O computador deve solicitar o número do empregado (2 dígitos), o sobrenome (até 15 caracteres) e o recebimento bruto; deve apresentar o registro (número), o nome, o recebimento bruto e o líquido.

5.7 Estruture um programa que determine a média aritmética de n parcelas dadas, tendo sido fornecido antes o número de parcelas.

5.8 Faça o algoritmo, o teste com "lápis e borracha" e o programa para executarmos uma divisão por meio de subtrações sucessivas.

5.9 Faça o algoritmo e o programa para executarmos uma multiplicação por meio de adições sucessivas.

5.10 Estruture um programa que efetua n! nas duas situações, isto é,

$$n! = n\ (n-1)\ (n-2) \ldots 1$$

e

$$n! = 1 \cdot 2 \cdot 3 \cdot \ldots\ (n-2)\ (n-1)\ n$$

Apresente os fluxogramas e os respectivos testes numéricos.

5.11 Faça o estudo completo de um programa que coloca três números quaisquer em ordem crescente.

5.12 Faça o estudo completo de um programa que coloca três "strings" quaisquer em ordem alfabética.

5.13 Considere o programa do exercício 5.11 com repetição (ou "-ções"). O computador deve avisar quantas e quais são as propostas repetidas.



5.14 Dê a resposta do programa abaixo.

```
1Ø  LET A =  - 1ØØ
2Ø  IF A > Ø AND A <  = 1Ø THEN  PRINT "OK",A
3Ø  LET A = A + 1
4Ø  GOTO 2Ø
```

5.15 Refaça o problema 5.6 usando nas comparações os operadores lógicos.

5.16 O que faz a função INT (X)? (veja p. 55)

5.17 Faça um programa para determinar se um número é inteiro ou não e se é par ou ímpar.

5.18 Sendo **X = NOT ( ( (A AND NOT B) OR C) AND NOT C)**, faça um programa para tabelar, no vídeo, as respostas em função das combinações dos estados de A (Ø ou 1), B (Ø ou 1) e C (Ø ou 1).

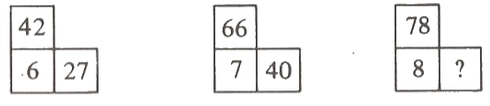
| A | B | C | X |
|---|---|---|---|
| Ø | Ø | Ø | ___ |
| Ø | Ø | 1 | ___ |
| Ø | 1 | Ø | ___ |
| Ø | 1 | 1 | ___ |
| 1 | Ø | Ø | ___ |
| 1 | Ø | 1 | ___ |
| 1 | 1 | Ø | ___ |
| 1 | 1 | 1 | ___ |

## V. RAXAKUKA

5.1 Escreva, dentro dos parênteses, as palavras que faltam:

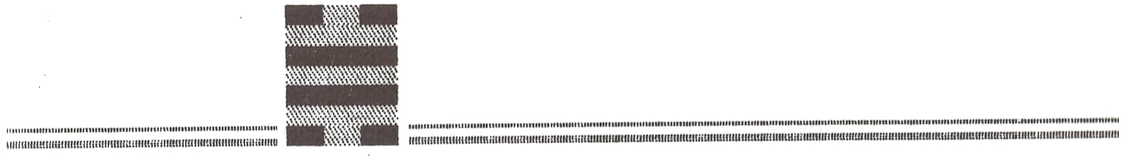
BRANCO (NELA) LUZES  RUMO (ERMA) PENA
FIASCO (.....) LICOR  ISTO (.....) CRIO

5.2 Coloque o número que falta:



5.3 Paulo e Ernani possuem a mesma quantia. Paulo, entretanto, tem mais dinheiro que Jorge e Jorge mais dinheiro que Carlos. Um outro homem, Antonio, tem menos dinheiro que Paulo e mais dinheiro do que Carlos, mas não tem tanto quanto Jorge. Ernani tem menos dinheiro que seu amigo José.

Se a diferença entre o dinheiro de cada um é de Cr$ 1.250,00 e o menos rico tem Cr$ 5,00, quanto cada um possui?



## FOR... NEXT... STEP

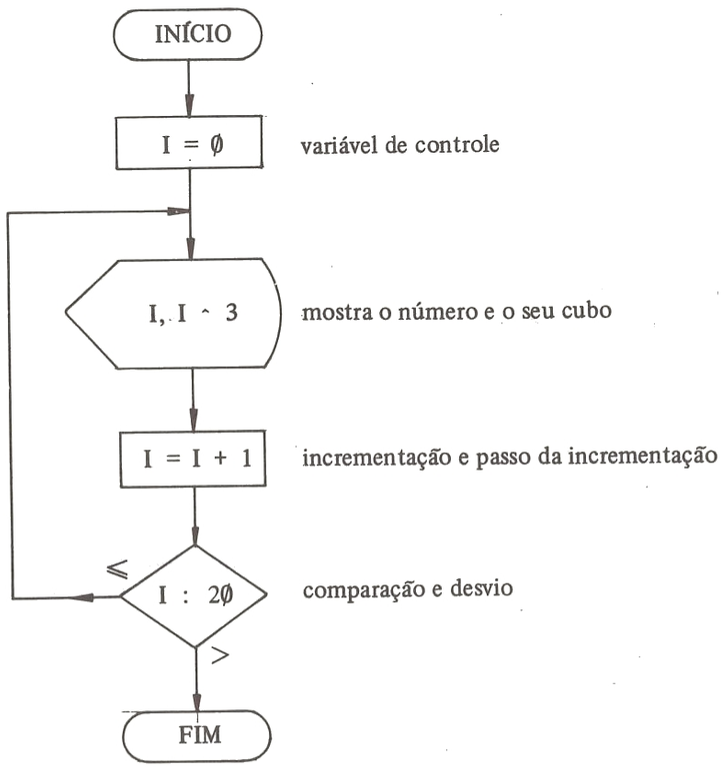
*"Aprender*
*é descobrir que você já sabe.*
*Fazer*
*é demonstrar que você o sabe.*
*Ensinar*
*é lembrar aos outros que*
*eles sabem tanto quanto você.*
*Nós*
*somos todos aprendizes,*
*fazedores,*
*professores."*

*Richard Bach*

## I. APRESENTANDO O FOR... NEXT

A grande maioria das aplicações reais no mundo do processamento de dados exige que uma determinada seqüência de instruções seja repetida várias vezes para se chegar ao resultado esperado. Esses processos repetitivos ou *loop*, que vimos até agora usando os comandos **IF... THEN** e **GOTO**, podem ser implementados de uma forma muito mais prática com a *dupla* formada pelos comandos **FOR** e **NEXT**. Eles reúnem todas as funções básicas encontradas num contador finito: a definição do valor inicial da contagem, incrementação da variável de controle em passos preestabelecidos, comparação desta com o limite do *loop* e desvio condicional para o início do ciclo.

Analise o fluxograma abaixo:






A seguir digite o programa P6.1:

```
** 1∅  HOME: PRINT "TABELA DE CUBOS": PRINT
b∅  I = ∅
3∅  PRINT I, I ^ 3
4∅  I = I + 1
5∅  IF I < = 2∅ THEN 3∅
6∅  END
```

variável de controle — valor inicial

incrementação da variável de controle — valor do passo (incremento)

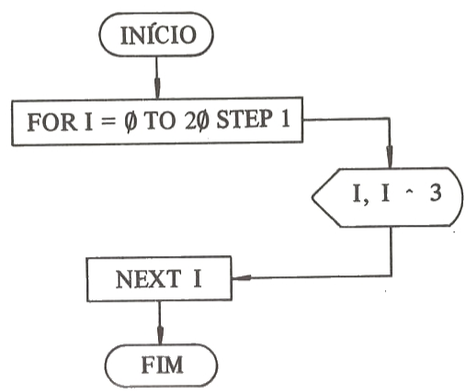
comparação e desvio — valor final — endereço do desvio

P6.1

(**) Na instrução 1∅ é usado o símbolo " : " como separador de comandos usados numa mesma linha. Isso é permitido no Applesoft BASIC.

Rodando P6.1, ele produzirá a saída esperada: uma tabela de cubos dos números de ∅ a 2∅, de um em um. Veja o mesmo programa com **FOR... NEXT**.

Primeiro o fluxograma:



Agora a codificação:

```
1Ø  HOME: PRINT "TABELA DE CUBOS": PRINT

2Ø  FOR I = Ø TO 2Ø STEP 1

3Ø  PRINT  I, I ^3

4Ø  NEXT I

5Ø  END
```

variável de controle

valores inicial e final

valor do passo ou incremento

incrementação comparação e desvio

P6.2

Execute o novo programa. Qual a sua saída? É a mesma que a anterior?

**R-1**

Cite algumas vantagens do uso do **FOR... NEXT** para controlar processos iterativos (repetitivos, *loops*).

**R-2**

Aqui estão algumas das vantagens do **FOR... NEXT** sobre a abordagem feita no capítulo 5, com **IF... THEN**:



1) o programa fica mais fácil de entender, especialmente quando o *loop* se torna mais complexo;
2) há uma economia de espaço, pois uma linha inteira pode ser eliminada do programa (compare P6.1 com P6.2);
3) a velocidade de processamento aumenta significativamente;
4) o programador não precisa se preocupar com comparações e desvios, que são feitos internamente com o **FOR... NEXT.**

## II. INTRODUZINDO PAUSAS EM UM PROGRAMA

Analise o programa P6.3.

```
1Ø  REM    *** PAUSA ***
2Ø  HOME
3Ø  VTAB 3
4Ø  HTAB 1Ø
5Ø  PRINT "VOU INICIAR A PAUSA..."
6Ø  FOR A = 1 TO 3ØØ
7Ø  NEXT A
8Ø  VTAB 2Ø
9Ø  PRINT  TAB( 25);"...ACABEI"
```

P6.3

Qual a finalidade do *loop* 6Ø - 7Ø?

**R-3**

O computador leva um certo tempo para completar uma tarefa qualquer, por menor que seja. Se quisermos, portanto, fazer com que haja uma pausa no meio do programa, podemos colocá-lo num *loop* finito; o número de passagens pelo *loop*, entre outros fatores, define o tempo durante o qual o computador ficará *parado*. Isso é o que foi feito pelas instruções 6Ø e 7Ø do P6.3.

Varie o limite superior da contagem, na linha 6Ø. Qual é o efeito dessa modificação?

**R-4**

## III. STEP: O PASSO

Voltando ao P6.2, mude a linha 2Ø para colocar um incremento (ou passo) igual a 2.4:

```
2Ø  FOR I = Ø TO 2Ø STEP 2.4
```

Como ficou a sua saída?

**R-5**

Em que é alterada a execução do programa se retirarmos por completo a *cláusula* **STEP**?

```
2Ø  FOR I = Ø TO 2Ø
```

**R-6**

Há alguma diferença entre este programa e o P6.2 original, quanto à saída?

**R-7**



O que você conclui sobre o passo de um *loop* que não o tenha especificado pelo **STEP**?

**R-8**

Faça o programa abaixo:

Codificação

```
1Ø  HOME
2Ø  FOR A = 1 TO 4Ø STEP 3
3Ø  PRINT A; TAB( A);"#"
4Ø  NEXT A
```

P6.4

Faça o fluxograma correspondente a esse programa, como exercício. O que faz o programa P6.4?

**R-9**

Rode o programa e confirme sua resposta.
A cláusula **STEP** pode ser negativa e, neste caso, teremos um contador regressivo.

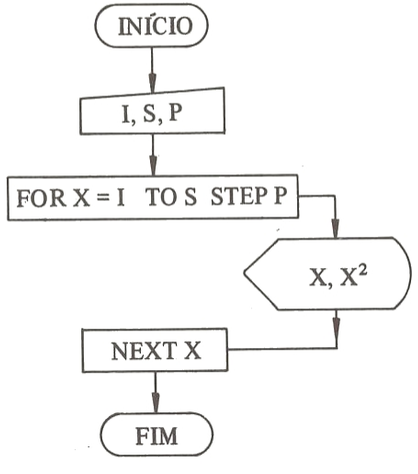
```
1Ø  HOME
2Ø  FOR A = 4Ø TO 1 STEP  - 3
3Ø  PRINT A; TAB( A);"#"
4Ø  NEXT A
```

P6.5

Os valores inicial, final e do passo do comando **FOR... NEXT** podem ser assumidos por variáveis simples ou expressões matemáticas.

Se quisermos esquematizar um contador com limites inferior ou superior, e, ainda, com passo variável, podemos estruturá-lo da forma seguinte:

Fluxograma




Codificação

```
10  REM  * QUADRADOS *
20  HOME
30  VTAB 3: HTAB 16
40  PRINT "QUADRADOS"
50  PRINT
60  INPUT "ESCREVA OS VALORES DOS LIMITES E DO PASSO, S
EPARADOS POR VIRGULAS: ";I,S,P
70  PRINT : PRINT  TAB( 5);"X","X^2"
80  FOR X = I TO S STEP P
90  PRINT  TAB( 5);X,X * X
100  PRINT
110  NEXT X
```

P6.6

Rode o programa P6.6, atribuindo valores a I, S e P e analise o resultado. Por exemplo: I = -2, S = 20 e P = 5, ou I = 10, S = 1, P = 2.

Qual o "porquê" dessas respostas?





## IV. MAIS COMANDOS NUMA LINHA

Podemos simplificar bastante os nossos programas com o auxílio dos dois pontos. Em certos casos é vantajoso; em outros, porém, podem aparecer problemas.

Já vimos que é possível incluir mais de um comando em uma linha de programação, desde que os comandos venham separados por *dois pontos* " : ".

Por exemplo, suponha que você queira pular quatro linhas em uma saída no vídeo. Podemos fazer:

```
10  PRINT
20  PRINT
30  PRINT
40  PRINT
```

porém, com o uso dos *dois pontos,* passamos a ter:

```
10  PRINT : PRINT : PRINT : PRINT
```

Carregue e rode o programa abaixo:

```
10  HOME : INPUT "COLOQUE OS VALORES DE A,B,C ";A,B,C
20  PRINT : PRINT : PRINT "DELTA = ";B * B - 4 * A * C
```

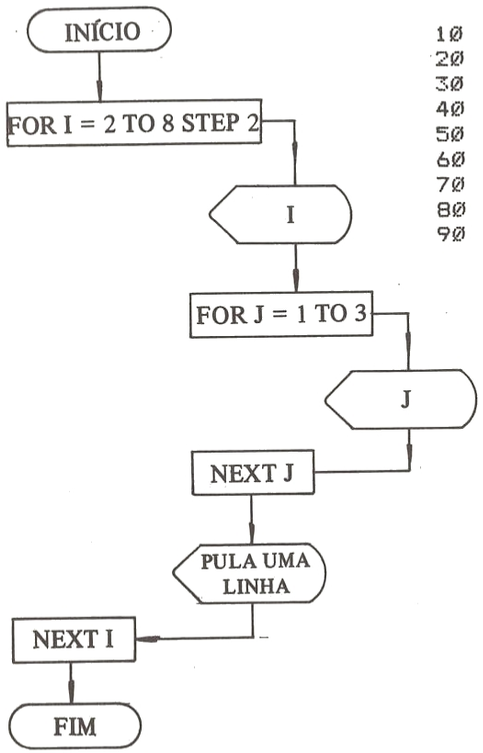
P6.7

A grande desvantagem em usar instruções com múltiplos comandos é a dificuldade que surge em mudar um dado ou variável em um comando intermediário. Devemos perceber, também, que um **GOTO** somente pode passar a execução para o *primeiro* comando da linha a que este se refere; os comandos seguintes serão executados em seqüência.

## V. LOOP DENTRO DE LOOP

Rode o programa P6.8:

Fluxograma




Codificação

```
1Ø  REM  TESTE
2Ø  HOME
3Ø  FOR I = 2 TO 8 STEP 2
4Ø  PRINT "I = ";I
5Ø  FOR J = 1 TO 3
6Ø  PRINT  SPC( 3);"J = ";J
7Ø  NEXT J
8Ø  PRINT
9Ø  NEXT I
```

P6.8

Qual a sua explicação para a saída obtida?

**R-11**



O programa P6.8 consiste em dois *loops* executados simultaneamente. O *loop* interno (linhas 5Ø a 7Ø) é executado por completo a cada iteração do *loop* externo. Por isso, o **PRINT** da linha 4Ø é executado quatro vezes, enquanto o **PRINT** dentro do *loop* interno, na linha 6Ø, é executado 4 x 3, ou 12 vezes.

Liste o programa com a modificação abaixo, formando um terceiro nível nos *loops*:

```
65  FOR TEMPO = 1 TO 1ØØ: NEXT TEMPO
```

Qual foi a alteração na execução, com a inclusão dessa pausa?

**R-12**

Retire a linha 65 e introduza:

```
85  FOR TEMPO = 1 TO 5ØØ: NEXT TEMPO
```

Tente prever o resultado, ao rodar o programa.

**R-13**

### V.1 – Loops Entrelaçados

Faça o programa P6.9, exemplo de uma aplicação incorreta do **FOR... NEXT**:

```
1Ø  HOME
2Ø  FOR N = 1Ø TO 2Ø
3Ø  PRINT N;" ";
4Ø  FOR J = 3Ø TO 4Ø
5Ø  PRINT J;" ";
6Ø  NEXT N
7Ø  NEXT J
```

P.6.9

Rode, agora, P6.9 e explique a sua saída.

Note que o *loop* 2Ø - 6Ø está enlaçado com o *loop* 4Ø - 7Ø. Isto em *BASIC* não é permitido.

## V.2 – Mais Alguns Exemplos

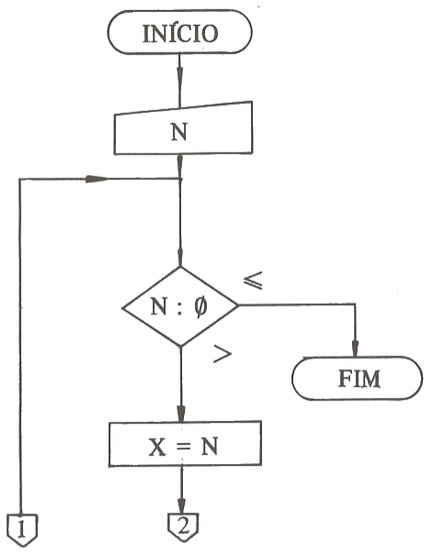
Estruture um programa que peça um número positivo, inteiro e que marque " * ", a partir do número dado, até chegar em um único símbolo.

Sendo N = 4, a saída será:

```
* * * *

* * *

* *

*
```

Resolveremos esse problema de dois modos, o primeiro *não* usando o comando **FOR... NEXT** e o segundo, sim.

Fluxograma

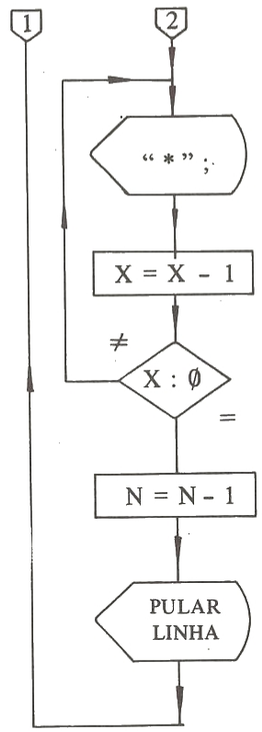



Codificação

```
1Ø  HOME
2Ø  INPUT "QUAL O NUMERO? ";N
3Ø  PRINT
4Ø  IF N <  = Ø THEN  END
5Ø X = N
6Ø  PRINT "* ";
7Ø X = X - 1
8Ø  IF X <  > Ø THEN 6Ø
9Ø N = N - 1
1ØØ  PRINT : PRINT
11Ø  GOTO 4Ø
```

P6.1Ø

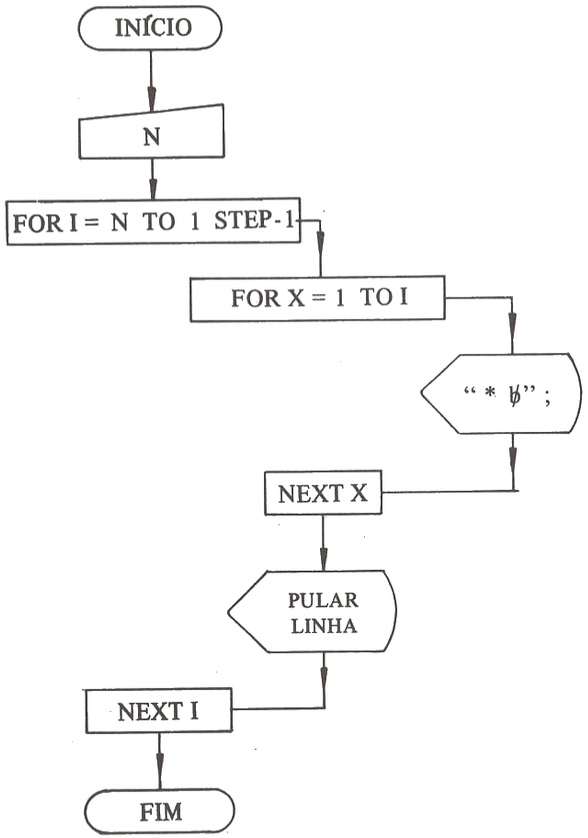





**RUN**ize P6.1Ø e escreva a resposta.

**R-14**

Vamos, agora, refazer P6.1Ø, usando **FOR... NEXT**.




```
10  HOME
20  INPUT "QUAL O NUMERO? ";N
30  PRINT
40  FOR I = N TO 1 STEP  - 1
50  FOR X = I TO 1 STEP  - 1
60  PRINT "* ";
70  NEXT X
80  PRINT : PRINT
90  NEXT I
```

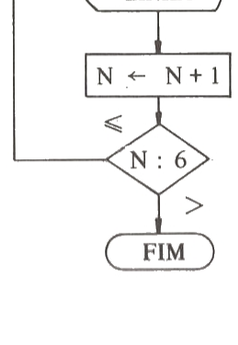
P6.11



Quais as diferenças mais marcantes entre os programas P6.1Ø e P6.11?

**R-15**

Analise o fluxograma e o programa a seguir.




```
1Ø  REM  GRAFICO DE QUADRADOS
2Ø  PRINT "N N^2": PRINT

Valor Inicial de N
3Ø N = 1
4Ø  PRINT N;" ";N ^ 2;

Valor inicial de A
5Ø A = 1
6Ø  PRINT "#";

Passo de A
7Ø A = A + 1

Valor final de A
8Ø  IF A <  = N ^ 2 THEN 6Ø
9Ø  PRINT : PRINT

Passo de N
1ØØ N = N + 1

Valor final de N
11Ø  IF N <  = 6 THEN 4Ø
12Ø  END
```

P6.12

Qual a resposta do programa? Era a esperada?

**R-16**

Refaça P6.12, usando **FOR... NEXT.**

**R-17**

## VI. EXERCÍCIOS PROPOSTOS

6.1 Faça o fluxograma e sua codificação, usando o comando **FOR... NEXT** para calcular a soma dos inteiros de 1 a 5Ø.

6.2 Um trabalhador recebe Cr$ 1,00, no primeiro dia de trabalho; Cr$ 2,00, no segundo dia de trabalho; Cr$ 4,00, no terceiro dia de trabalho; Cr$ 8,00, no quarto dia de trabalho, e assim, sucessivamente, dobrando a cada dia de trabalho durante 30 dias. Faça um programa que calcule quanto irá receber no 30º dia e quanto receberá pelos 30 dias trabalhados. Obs.: é obrigatório o uso de **FOR... NEXT.**

6.3 Refaça o programa anterior, considerando um total de N dias a ser estipulado pelo usuário.

6.4 Faça o fluxograma do programa 6.9 e diga, no fluxograma, o que está errado.

6.5 Faça o programa que apresenta a saída abaixo:

```
1
1  2
1  2  3
1  2  3  4
1  2  3  4  5
```

Obs.: O computador deverá pedir o número limite, que nesse exemplo é 5.



6.6 Escreva o programa para imprimir os inteiros de 1 a 5Ø, em ordem crescente, em 5 linhas com 1Ø colunas, usando **TAB, HTAB, VTAB e FOR... NEXT.**

6.7 Faça o programa que apresenta a seguinte saída:

```
1  *
2  *  *
3  *  *  *
4  *  *  *  *
5  *  *  *  *  *
```

Obs.: O computador deve pedir o número limite, que nesse exemplo é 5.

6.8 Faça o programa que apresenta a seguinte saída:

```
1  2  3  4  5  6  7  8  9
   2  3  4  5  6  7  8
      3  4  5  6  7
         4  5  6
```



Obs.: O computador deverá perguntar qual o número final e o programa deverá conter obrigatoriamente **FOR... NEXT.**

6.9 Refaça o exercício 5.1Ø do capítulo anterior, usando o comando **FOR... NEXT.**

6.10 Faça o fluxograma e o programa, para tabelar a função y = Ax + B.

O computador deve solicitar:

```
VALOR DE A?■
VALOR DE B?
X INICIAL?
X FINAL?
INCREMENTO?
```

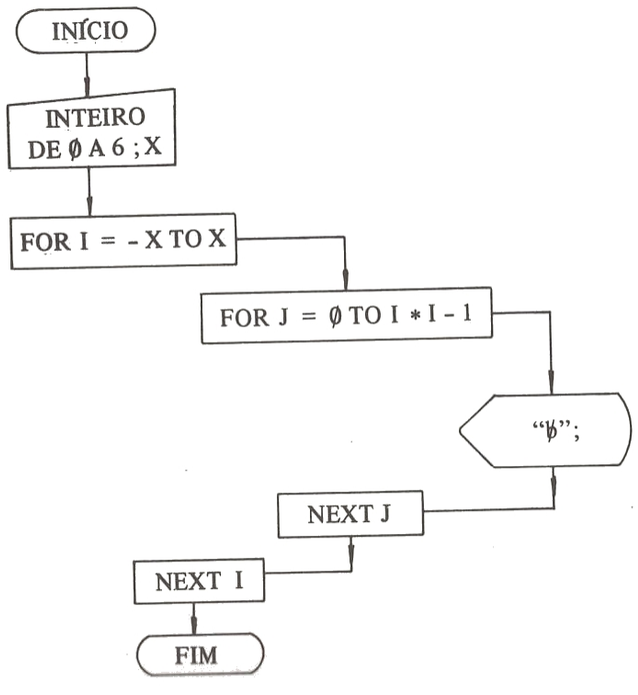
e a saída deve ser:

```
TABELA DA FUNCAO AX + B

X INIC= ___   INCREM= ___   COEF ANG= ___

X FIN = ___                 COEF LIN= ___

                VALORES

          X                    Y
          ___                  ___
          ___                  ___
          .                    .
          :                    :
```

Obs.: Todos os dados tabelados devem apresentar, no máximo, duas casas decimais. Veja o exemplo do programa P5.3.

6.11 Refaça, usando **FOR... NEXT**, os problemas 5.8 e 5.9 do capítulo anterior.

6.12 O que faz o fluxograma abaixo?






6.13 Codifique em *BASIC* o fluxograma do problema 6.12 e confirme sua resposta.

## VII. RAXAKUKA

6.1 Um fazendeiro morreu e deixou um rebanho de 142 vacas para 5 filhos. No testamento, especificou:

para o primeiro filho, 1/3 do rebanho.
para o segundo filho, 1/4 do rebanho.
para o terceiro filho, 1/6 do rebanho.
para o quarto filho, 1/8 do rebanho.
e ao último filho, 1/9 do rebanho.

Os filhos verificaram que não podia ser dividido desse modo, pois cada um receberia frações de vaca e nenhum queria abrir mão de sua quota. Um vizinho inteligente resolveu o problema satisfazendo a todos. Como isso aconteceu?

6.2 Sublinhe a palavra que completa a frase a seguir:

TOHIAL está para XOAJIBAI
assim como
MANHEACI está para:
QOILTACM, XOIZATABAI, CHASTIPLAS, TIPSICHATRI

6.3 Escrever, com quatro quatros e sinais matemáticos, uma expressão que seja igual a um número inteiro. Na expressão não pode figurar (além dos quatro quatros) nenhum algarismo ou letra ou símbolo algébrico que envolva letra, tais como: log, lim, etc.

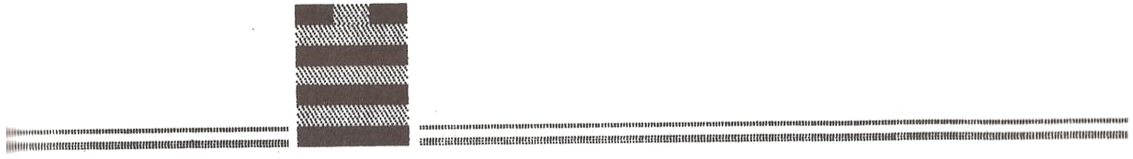
Exemplos:

$$0 = \frac{4}{4} - \frac{4}{4}$$

$$1 = \frac{44}{44}$$

$$97 = 4! \times 4 + \frac{4}{4}$$

Afirmam os pacientes calculistas que será possível escrever, com quatro quatros, todos os números inteiros, desde 0 até 100!!!



# Matrizes e variáveis indexadas

*"A marca da ignorância é a profundidade da crença na injustiça e na tragédia. Ao que a lagarta chama fim do mundo, o mestre chama borboleta."*

*Richard Bach*

## I. INTRODUÇÃO

Para atribuir informações numéricas ou alfanuméricas, usamos, até agora, o comando **LET**; quando desejávamos armazenar, via teclado, uma lista de itens, usávamos o comando **INPUT**, porém, se quisermos armazenar, sob uma variável de mesmo nome, essa mesma lista, devemos lançar mão da *VARIÁVEL SUBSCRITA* ou *INDEXADA*.

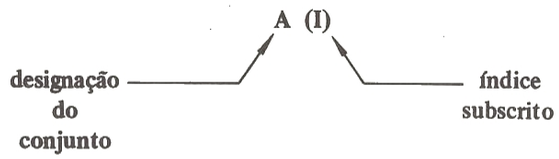
## II. VARIÁVEL SUBSCRITA

Já sabemos que *variável* é um nome que representa um determinado dado ou informação armazenado em uma célula de memória do computador. As variáveis, além das letras do alfabeto, podem receber números, para auxiliar a identificação:

**A0 ; B7 ; A1$ ; Z9$**

Entretanto, podemos endereçar, isto é, enumerar, indiretamente, uma variável, por meio de um índice, *subscrito*, agregado a um contador ou a uma operação algébrica.

A sintaxe da variável subscrita é:




## III. MATRIZES

Matrizes são coleções ou conjuntos de dados organizados e armazenados sob a mesma designação de variável. A menor parte de uma matriz é denominada *elemento.*

Cada elemento é uma informação ou dado, isto é, pode ser uma cadeia alfanumérica ou um número. O elemento da matriz é identificado pelo nome da matriz, seguido pelos índices que o posicionam corretamente. Entretanto, usando uma *variável subscrita, podemos definir a posição de cada elemento de uma matriz.*



### III.1 – Matrizes Unidimensionais

Como o próprio nome diz, matrizes unidimensionais são aquelas que apresentam uma única linha ou uma única coluna.

Analisemos os exemplos abaixo:

**N = [5, 7, 9, -1]** **A$ = [ROLINHA, PARDAL, POMBA]**

$$D\$ = \begin{bmatrix} \text{ANA} \\ \text{PAULA} \\ \text{TERESA} \\ \text{SILVIA} \end{bmatrix} \qquad M = \begin{bmatrix} 1.41 \\ 1.73 \\ 2.78 \\ 4.81 \end{bmatrix} \qquad X\$ = \begin{bmatrix} \text{ANA} \\ 8.5 \\ \text{MARIA} \\ 3.8 \end{bmatrix}$$

As matrizes N e M são matrizes numéricas, D$ e A$ são matrizes *string* e X$ também é uma matriz *string* cujos números são seqüências de caracteres.

Para localizarmos um dos elementos de uma matriz, lançamos mão, como já vimos, da variável subscrita. Assim, da matriz:

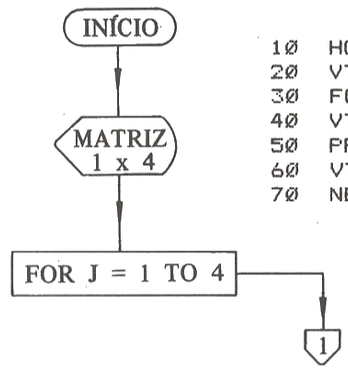
(1) (2) (3) (4)

N = [5 , 7 , 9 , -1]

temos: N(1) = 5
N(2) = 7
N(3) = 9
N(4) = -1

Estruture um programa que faça a introdução dos elementos em uma matriz (1 x 4), isto é, 1 linha com 4 colunas, usando o comando **FOR... NEXT.**

Fluxograma

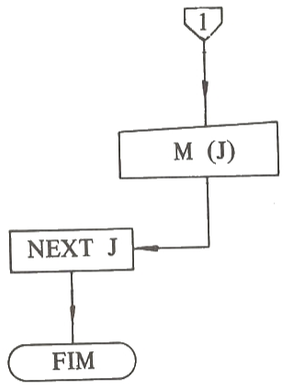



M = [1Ø, 2Ø, 3Ø, 4Ø]

Codificação

```
1Ø  HOME
2Ø  VTAB 3: HTAB 13: PRINT "MATRIZ (1 X 4)"
3Ø  FOR J = 1 TO 4
4Ø  VTAB 7
5Ø  PRINT "QUAL O VALOR DE M";J; SPC( 255)
6Ø  VTAB 7: HTAB 22: INPUT M(J)
7Ø  NEXT J
```

P7.1




Facilmente percebemos que o *loop* é definido pelas instruções 3Ø - 7Ø - 3Ø. Rode o programa P7.1 e atribua a M os elementos 1Ø, 2Ø, 3Ø e 4Ø. Após terminada a definição de M, nada apareceu na tela, uma vez que o programa não apresenta saída. No entanto a matriz M foi definida.

Faça:

```
PRINT M(1), M(2), M(3), M(4)
```

**R-1**

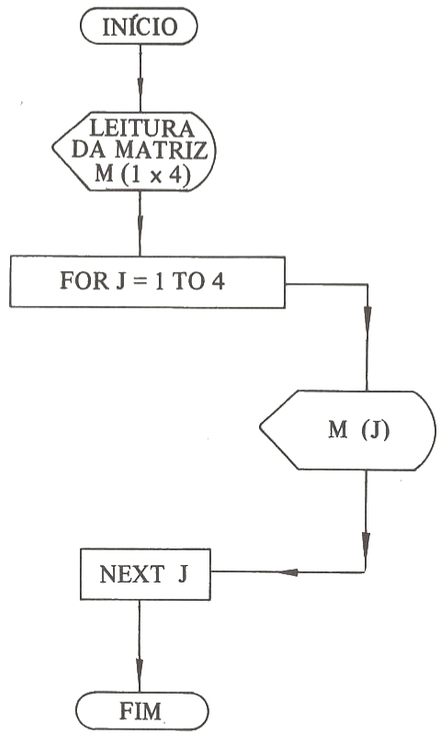
Qual a razão do comando **SPC (255)** na instrução 5Ø?

**R-2**

A seguir, vamos estruturar um pequeno programa para ler a matriz M. Analise o fluxograma a seguir e a sua respectiva codificação.



Fluxograma




Codificação

```
8Ø  FOR X = 1 TO 5ØØ: NEXT X
9Ø  HOME
1ØØ  VTAB 3: HTAB 13: PRINT
     "MATRIZ M (1 X 4)"
11Ø  PRINT : PRINT
12Ø  FOR J = 1 TO 4
13Ø  HTAB 17
14Ø  PRINT "M";J;" = ";M(J)
15Ø  PRINT
16Ø  NEXT J
```

P7.2

Agora, atenção: de acordo com o programa P7.1, a matriz M deve ser *M(1) =1Ø; M(2) = 2Ø; M(3) = 3Ø e M(4) = 4Ø.*

Faça **RUN 8Ø**. Qual a sua resposta? Por quê?

**R-3**

**RUN**ize, em seguida, o programa todo, entrando novamente com M(1) = 1Ø, M(2) = 2Ø, M(3) = 3Ø e M(4) = 4Ø.

Qual a resposta?

**R-4**

Digite **GOTO 8Ø** e, em seguida, pressione *RETURN*.
Qual o resultado?

**R-5**

Qual a diferença entre **RUN 8Ø** e **GOTO 8Ø**?

**R-6**

Agora, façamos um programa que estruture a matriz Q = (1 x 13) abaixo.
Q = [49, Ø5, 18, 27, 33, 42, 57, 68, 19, 35, 94, 1Ø1, Ø1]

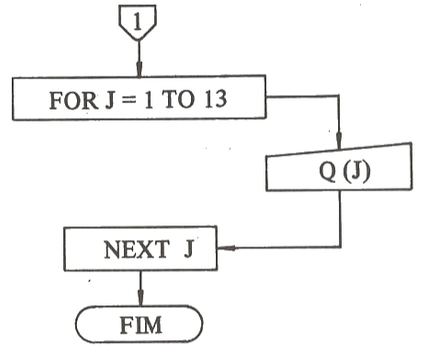
Fluxograma


Codificação

```
1Ø  REM  LEITURA DE Q
2Ø  HOME
3Ø  VTAB 3: HTAB 13: PRINT "MATRIZ (1 X 13)"
4Ø  FOR J = 1 TO 13
5Ø  VTAB 7
6Ø  PRINT "QUAL O VALOR DE M";J; SPC( 255)
7Ø  VTAB 7: HTAB 22: INPUT M(J)
8Ø  NEXT J
```

P7.3






Rode o programa P7.3 e reintroduza os valores dos elementos determinados da matriz anterior.

Você conseguiu definir todos os elementos?

**R-7**

Qual a mensagem enviada pelo computador?

**R-8**

Quando uma matriz é estruturada, o computador admite um máximo de 1Ø itens subscritos.

Na situação descrita anteriormente, devemos usar o comando **DIM** (**DIM** *ENSION*) para reservar o necessário espaço na memória do computador e definir a matriz em questão.

Insira, no programa P7.3, a instrução:

```
5  DIM Q(13)
```

Rode o programa P7.3 e defina a matriz Q.

De um modo geral, é de boa prática dimensionar, sempre, a matriz em questão, por meio da *instrução de declaração DIM*, na qual é definido o nome a ser usado e o número de elementos que será associado a esse nome.

## III.2 – Matrizes Bidimensionais

As matrizes podem apresentar mais de uma dimensão. A figura 7.1 mostra uma matriz bidimensional (3 x 4), com 3 linhas e 4 colunas.

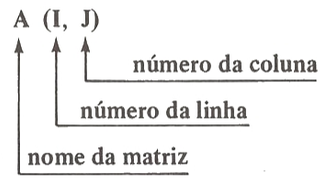
| I ↓ | J → 1 | 2 | 3 | 4 |
|---|---|---|---|---|
| 1 | 1,1 | 1,2 | 1,3 | 1,4 |
| 2 | 2,1 | 2,2 | 2,3 | 2,4 |
| 3 | 3,1 | 3,2 | 3,3 | 3,4 |

fig. 7.1

São exemplos de matrizes bidimensionais:

$$A = \begin{bmatrix} \emptyset & 1 & 2 & 3 \\ 9 & 7 & 6 & 5 \\ -1 & 2 & \emptyset & 4 \end{bmatrix} \qquad M\$ = \begin{bmatrix} \text{SILVA} & \text{JOSE'} & \text{DA} \\ \text{FERREIRA} & \text{CARLOS} & \text{ALBERTO} \\ \text{GOMES} & \text{CARLOS} & \text{JOSE'} \end{bmatrix}$$

Os elementos de uma matriz bidimensional necessitam de dois índices para ficarem definidos:




Se fôssemos usar a matriz A (3 x 4) definida no exemplo acima, num programa, precisaríamos dimensioná-la tanto em linha como em coluna, e nesse caso faríamos:

**DIM A (3, 4)**

que cria uma matriz de 4 linhas e 5 colunas. Adiante veremos o porquê.



Analise o programa P7.4:

```
10  REM  MATRIZ BIDIMENSIONAL
20  FOR I = 1 TO 3
30  FOR J = 1 TO 4
40 A(I,J) = I * 10 + J
50  NEXT J
60  NEXT I
```

P7.4

Rode o P7.4. Qual a matriz que ele definiu?

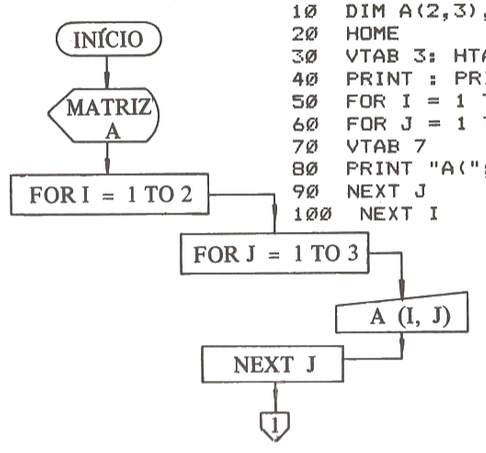
**R-9**

Estruture um programa para mostrar, na tela, a matriz definida pelo programa P7.4.

Lembrando que para somar duas matrizes numéricas devemos somar elementos correspondentes, faça o programa que defina e some as matrizes A e B dadas abaixo:

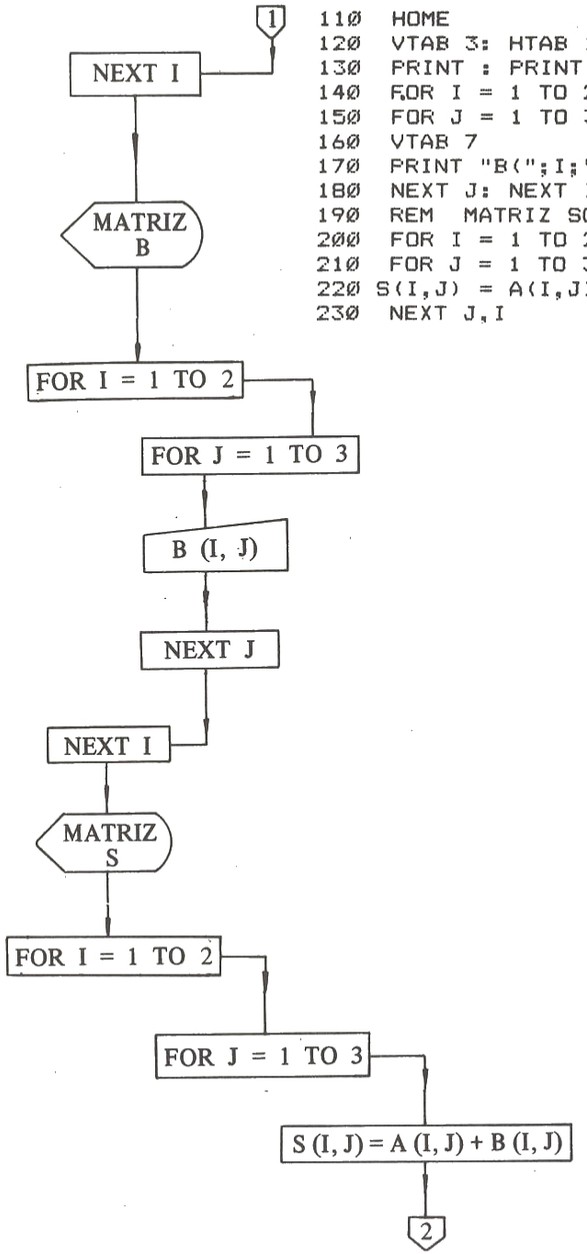
$$A = \begin{bmatrix} 0 & 1 & 2 \\ -1 & 3 & 1 \end{bmatrix} \qquad B = \begin{bmatrix} 3 & 1 & 4 \\ 5 & -1 & 2 \end{bmatrix}$$

Fluxograma



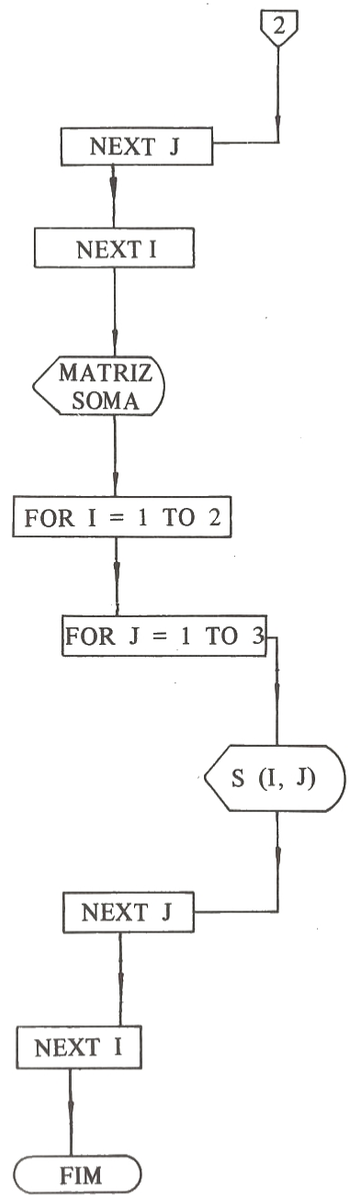

Codificação

```
10  DIM A(2,3),B(2,3),S(2,3)
20  HOME
30  VTAB 3: HTAB 16: PRINT "MATRIZ A"
40  PRINT : PRINT
50  FOR I = 1 TO 2
60  FOR J = 1 TO 3
70  VTAB 7
80  PRINT "A(";I;",";J;") = ";: INPUT A(I,J)
90  NEXT J
100  NEXT I
```




```
110  HOME
120  VTAB 3: HTAB 16: PRINT "MATRIZ B"
130  PRINT : PRINT
140  FOR I = 1 TO 2
150  FOR J = 1 TO 3
160  VTAB 7
170  PRINT "B(";I;",";J;") = ";: INPUT B(I,J)
180  NEXT J: NEXT I
190  REM  MATRIZ SOMA
200  FOR I = 1 TO 2
210  FOR J = 1 TO 3
220 S(I,J) = A(I,J) + B(I,J)
230  NEXT J,I
```






```
240  REM APRESENTACAO FINAL
250  HOME
260  VTAB 5: HTAB 14: PRINT "MATRIZ SOMA"
270 X = 8
280  FOR I = 1 TO 2
290 Y = 14
300  FOR J = 1 TO 3
310  VTAB X: HTAB Y
320  PRINT S(I,J)
330 Y = Y + 5
340  NEXT J
350 X = X + 2
360  NEXT I
370  IF I = 3 AND J = 4 THEN 1000
380  HOME : END
1000  VTAB 20: HTAB 13
1010  INPUT "CONTINUA (S/N)? ";R$
1020  IF R$ = "N" THEN 380
1030  GOTO 20
```

O programa P7.5 deve ser encarado como um exemplo didático e, por isso, abusamos um pouco dos **FOR... NEXT**.

Rode o programa P7.5 e defina as matrizes A e B. Qual é a matriz soma?

**R-1Ø**

Se cada índice admite até um máximo de 1Ø sem declaração de dimensionamento, qual o motivo da instrução 1Ø?

**R-11**

Qual seu comentário sobre as instruções 9Ø e 1ØØ; 18Ø; 23Ø; 34Ø e 36Ø?

**R-12**

A instrução 38Ø será executada alguma vez? Por quê?

**R-13**



## III.3 – Matrizes de Três ou Mais Dimensões

O número máximo de dimensões que se pode declarar para uma matriz depende, de um modo geral, da versão BASIC utilizada pelo computador. No nosso caso, podemos declarar uma matriz de, até, 88 dimensões (!! ?).

Faça o fluxograma e a codificação do programa que define a matriz tridimensional da figura 7.2.

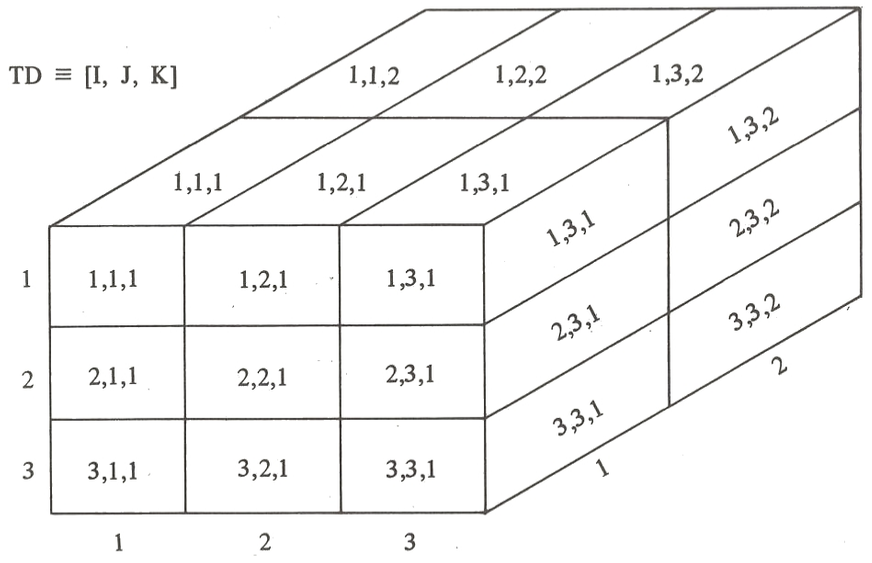



fig. 7.2

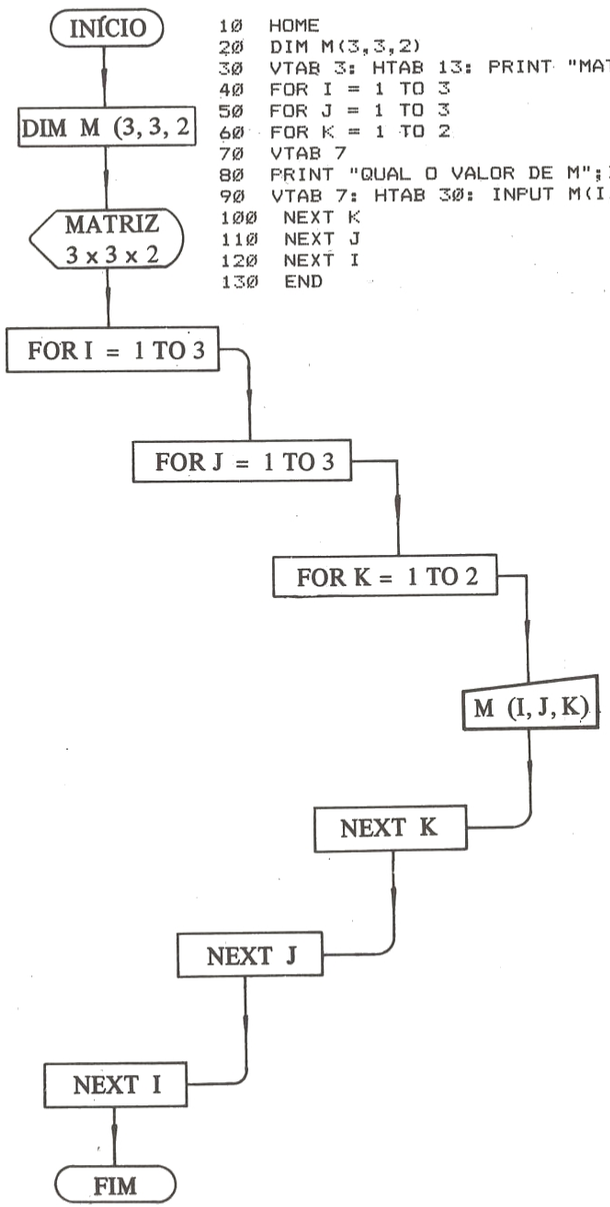
Os elementos que definem a matriz TD (3 x 3 x 2) são:

$a_{1,1,1} = 1$ $a_{1,2,1} = 7$ $a_{1,3,1} = 13$ $a_{1,1,2} = 2$ $a_{1,2,2} = 8$ $a_{1,3,2} = 14$

$a_{2,1,1} = 3$ $a_{2,2,1} = 9$ $a_{2,3,1} = 15$ $a_{2,1,2} = 4$ $a_{2,2,2} = 10$ $a_{2,3,2} = 16$

$a_{3,1,1} = 5$ $a_{3,2,1} = 11$ $a_{3,3,1} = 17$ $a_{3,1,2} = 6$ $a_{3,2,2} = 12$ $a_{3,3,2} = 18$

Fluxograma




Codificação

```
10  HOME
20  DIM M(3,3,2)
30  VTAB 3: HTAB 13: PRINT "MATRIZ (3 X 3 X 2)"
40  FOR I = 1 TO 3
50  FOR J = 1 TO 3
60  FOR K = 1 TO 2
70  VTAB 7
80  PRINT "QUAL O VALOR DE M";I;",";J;",";K; SPC( 255)
90  VTAB 7: HTAB 30: INPUT M(I,J,K)
100  NEXT K
110  NEXT J
120  NEXT I
130  END
```



**RUN**ize o programa P7.6 e atribua os valores definidos à matriz TD (3 x 3 x 2). Para efeito de verificação, após a definição, faça:

```
PRINT TD(1,1,1)
```

**R-14**

```
PRINT TD(2,1,2)
```

**R-15**

```
PRINT TD(3,3,2)
```

**R-16**

```
PRINT TD(4,1,1)
```

**R-17**

## III.4 – Algumas Observações e Particularidades do Applesoft

a) O subscrito ∅ é permissível. Um comando **DIM S(12)** reserva 13 *células* de memória, a saber: **S(∅), S(1), S(2) ... S(12).**

b) O comando **DIM** é automático quando é referenciada uma matriz de até 11 elementos por dimensão (∅ a 1∅).

c) Tente, sempre que possível, dimensionar, corretamente, a matriz em estudo ou em declaração.

d) A instrução **DIM** pode ser colocada em qualquer parte do programa, porém é bom hábito localizá-la no começo.

e) A grandeza definida pelo comando **DIM** pode ser feita por um número ou por uma expressão aritmética ou, ainda, por uma expressão numérica: **DIM S (2 * 3 + 4); DIM S(N); DIM S (A + 2∅).**

f) Usando um único comando **DIM**, pode-se declarar uma ou mais matrizes: **DIM A (1∅), B(2∅), C(15).**

g) Um programa pode conter alguns comandos DIM, mas cada matriz pode ser declarada, somente, uma vez, desse modo:

```
1∅  DIM A (2∅)
14∅ DIM A (18)
```

causam mensagens de erro.

h) A mesma designação de variável pode ser usada num programa: como *variável subscrita e como variável não subscrita*; **S1** e **S(1)** são variáveis diferentes.

i) Cada elemento de uma matriz *string* pode guardar até 255 caracteres.

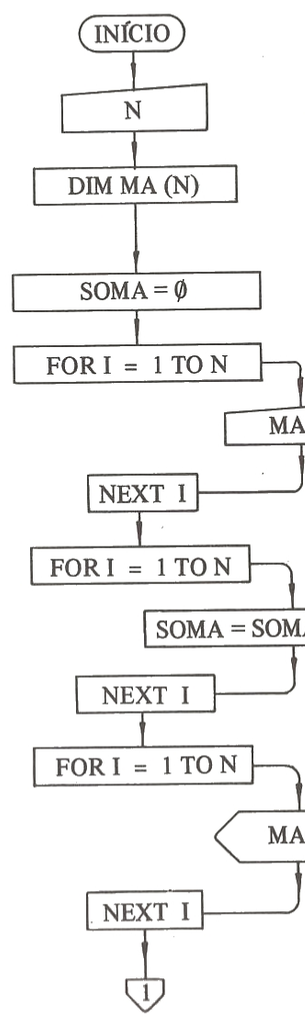
j) Se possível, defina sempre uma matriz de variáveis numéricas inteiras.



# IV. TRÊS APLICAÇÕES

Com uma matriz de uma dimensão, vamos escrever um programa que efetue a média aritmética de N dados de entrada.

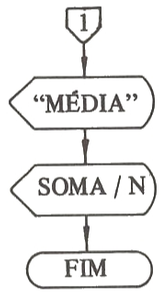
Fluxograma




Codificação

```
10  HOME
20  VTAB 3: HTAB 12: PRINT "MEDIA ARITMETICA"
30  VTAB 5: INPUT "QUANTOS ELEMENTOS? ";N
40  DIM MA(N)
50 SOMA = 0: PRINT
60  FOR I = 1 TO N
70  PRINT "E(";I;") = ";: INPUT MA(I)
80  NEXT I
90  FOR I = 1 TO N
100 SOMA = SOMA + MA(I)
110  NEXT I
120  HOME
130  VTAB 3: HTAB 12: PRINT "MEDIA ARITMETICA"
140  PRINT : PRINT "ELEMENTOS": PRINT
150  FOR I = 1 TO N
160  PRINT MA(I);" ";
170  NEXT I
180  PRINT : PRINT : PRINT "A MEDIA E' ";SOMA / N
190  END
```

P7.7

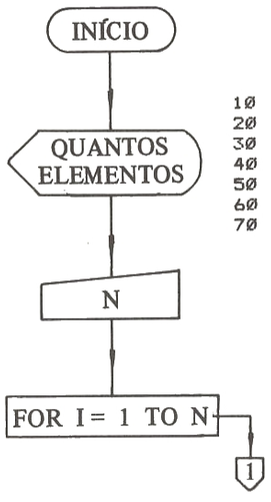



**RUN**ize o programa P7.7 e verifique seu resultado.

**R-18**

Uma outra aplicação simples é determinar qual o maior elemento em uma matriz, por exemplo, unidimensional:

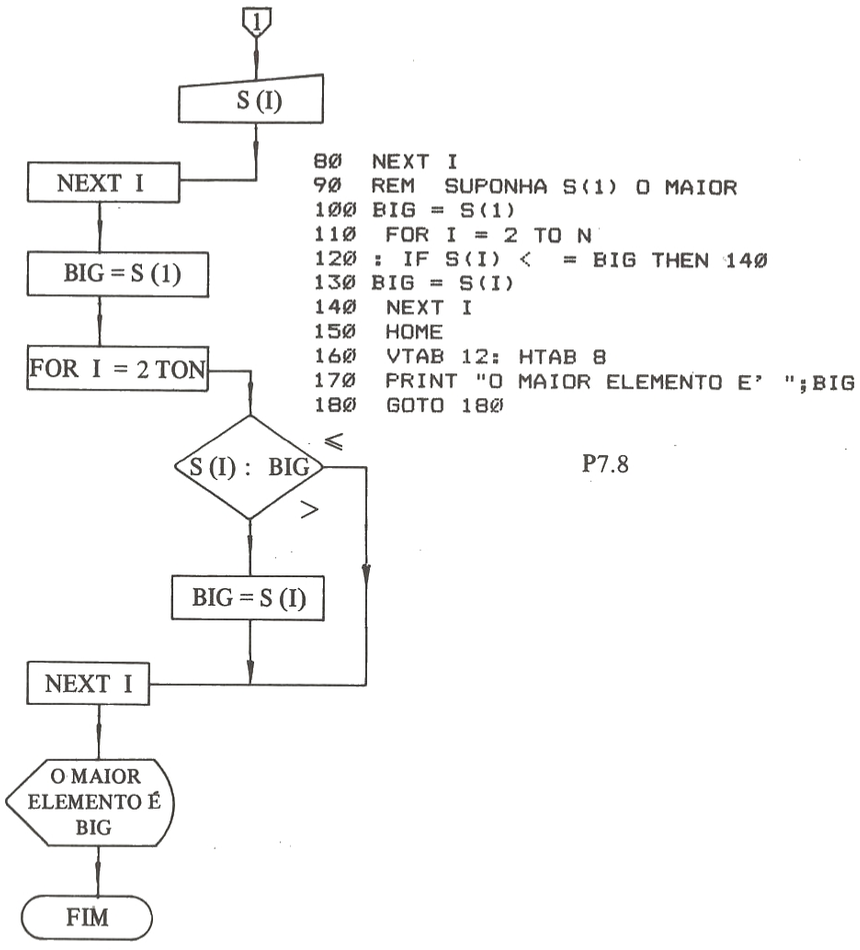
Fluxograma




Codificação

```
10  HOME
20  HTAB 10
30  INPUT "QUANTOS ELEMENTOS? ";N
40  DIM S(N)
50  FOR I = 1 TO N
60  HOME
70  VTAB 10: HTAB 5: PRINT "ELEMENTO N";I;: INPUT S(I)
```






```
80  NEXT I
90  REM  SUPONHA S(1) O MAIOR
100 BIG = S(1)
110  FOR I = 2 TO N
120 : IF S(I) <  = BIG THEN 140
130 BIG = S(I)
140  NEXT I
150  HOME
160  VTAB 12: HTAB 8
170  PRINT "O MAIOR ELEMENTO E' ";BIG
180  GOTO 180
```

P7.8

**RUN**ize e teste o programa P7.8.
Está *OK*?

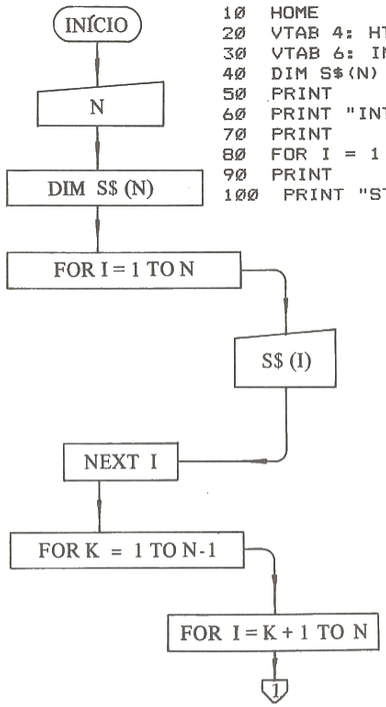
**R-19**

Discuta as instruções 5Ø, 1ØØ, 12Ø, 13Ø.

**R-2Ø**

Vamos ver, em seguida, um programa muito importante e de grande aplicação em muitos outros programas que você poderá desenvolver. Primeiro analise o fluxograma e depois carregue o computador e rode o programa.

Fluxograma

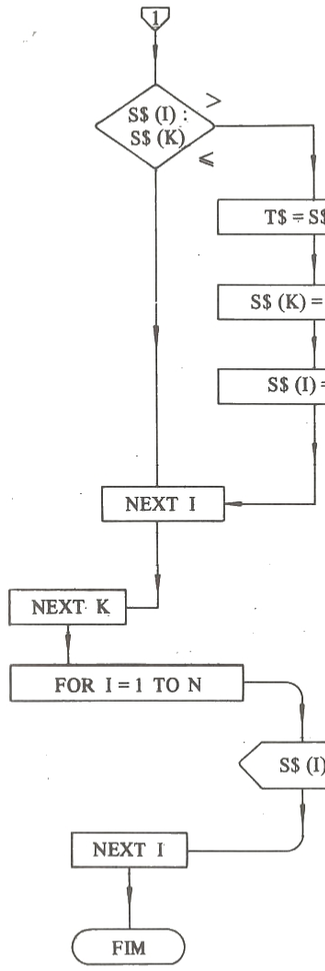



Codificação

```
1Ø  HOME
2Ø  VTAB 4: HTAB 7: PRINT "CLASSIFICACAO DE 'STRINGS'"
3Ø  VTAB 6: INPUT "QUANTOS STRINGS? ";N
4Ø  DIM S$(N)
5Ø  PRINT
6Ø  PRINT "INTRODUZA UM STRING POR VEZ"
7Ø  PRINT
8Ø  FOR I = 1 TO N
9Ø  PRINT
1ØØ  PRINT "STRING(";I;") = ";: INPUT "";S$(I)
```

P7.9






```
11Ø  NEXT I
12Ø  FOR K = 1 TO N - 1
13Ø  REM  SUPONHA S(K) O PRIMEIRO
14Ø  FOR I = K + 1 TO N
15Ø  IF S$(I) <  = S$(K) THEN 2ØØ
16Ø  REM INTERCAMBIO
17Ø T$ = S$(K)
18Ø S$(K) = S$(I)
19Ø S$(I) = T$
2ØØ  NEXT I: NEXT K
21Ø  HOME
22Ø  PRINT "'SORT' DECRESCENTE": PRINT : PRINT
23Ø  FOR I = 1 TO N
24Ø  PRINT S$(I)
25Ø  NEXT I
26Ø  END
```

P7.9

O que faz o programa P7.9?

**R-21**

Aplicado ao P7.9, qual o significado da palavra *SORT*?

**R-22**

Discuta as instruções de 13Ø a 16Ø.

**R-23**

O que fazem as instruções 17Ø, 18Ø e 19Ø?

**R-24**

## V. EXERCÍCIOS PROPOSTOS

7.1 Carregue, com o programa a seguir, o computador. No final, *NÃO DIGITE RUN.*



```
1Ø  HOME
2Ø  DIM A(4Ø)
3Ø  FOR L = 1 TO 3Ø
4Ø A(L) = L
5Ø  NEXT L
```

Digite, a seguir, as instruções, de acordo com a ordem abaixo, dizendo quais as saídas previstas:

a) **PRINT A(2Ø)**
b) **PRINT L**
c) **RUN**
d) **PRINT A(Ø)**
e) **PRINT A(2Ø)**
f) **PRINT A(35)**
g) **PRINT A(-1)**
h) **PRINT A(41)**
i) **PRINT L**

7.2 Escreva a matriz resultante do programa abaixo.

```
1Ø A = 1
2Ø  FOR I = 1 TO 3
3Ø  FOR J = 1 TO 2
4Ø M(I,J) = A
5Ø A = A + 2
6Ø  NEXT J
7Ø  NEXT I
```

7.3 Escreva o fluxograma e a codificação de um programa que armazene, em matriz, os inteiros de 1 a 12, sendo:

a) todos em uma linha
b) um inteiro por linha
c) três inteiros por linha
d) dois planos contendo três inteiros por linha

7.4 Escreva um programa que determine o menor elemento de uma matriz linear.

7.5 Escreva um programa que determine o maior elemento de uma matriz bidimensional, sem repetições de elementos, e indique qual a sua posição.

7.6 Estruture um programa para receber uma matriz linear de 3Ø elementos e separar os positivos dos negativos, fornecendo a soma dos positivos e o maior negativo.

7.7 Modifique o programa anterior, para fornecer, também, as médias simples dos dois campos numéricos.

7.8 Faça um programa que peça a entrada dos valores de uma matriz N x N e, em seguida, verifique se os elementos da diagonal principal são maiores que a soma dos outros elementos de suas linhas. A saída deve ser, por exemplo:

**MATRIZ N x N**

**ESTUDO DA DIAGONAL**

**A(1,1) = XX E' MAIOR QUE: A + B + C + ...**

**A(2,2) = YYY NÃO E' MAIOR QUE: M + O + P + ...**

**A(3,3) = ZZZ E' IGUAL A: R + S + T + ...**

7.9 Duas matrizes A e B apresentam dimensões I x J, em que I = 3 e J = 2. Escreva um programa que compare os correspondentes elementos de A e B e, em seguida, forme uma nova matriz (3 x 2) contendo, em ordem crescente, os maiores elementos. A nova matriz deverá ser denominada **MATRIZ DOS MAIORES A/B EM ORDEM CRESCENTE.**

7.10 Retome o problema anterior, para formar uma nova matriz que compare cada elemento correspondente, das duas matrizes, escolhendo o maior respectivamente. A nova matriz deverá ser denominada **"MATRIZ DOS MAIORES".**

7.11 Faça um programa que coloque, em ordem crescente, os elementos de cada coluna de uma matriz bidimensional.

7.12 Faça um programa que coloque, em ordem decrescente, os elementos de uma matriz tridimensional.

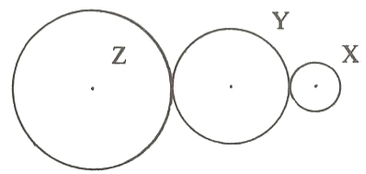
7.13 Faça um programa para multiplicar duas matrizes A (m x n) e B (n x r).



## VI. RAXAKUKA

7.1 Um marinheiro cujo navio afundou viu-se só, em uma ilha, com febre altíssima. Do naufrágio tinham sobrado duas garrafas de medicamento. Numa delas, havia 3/4 do conteúdo e a outra, igual, estava completamente vazia. As instruções para tomar o remédio prescreviam uma dose de metade da garrafa, nem mais nem menos. Como obter exatamente 1/2 garrafa?

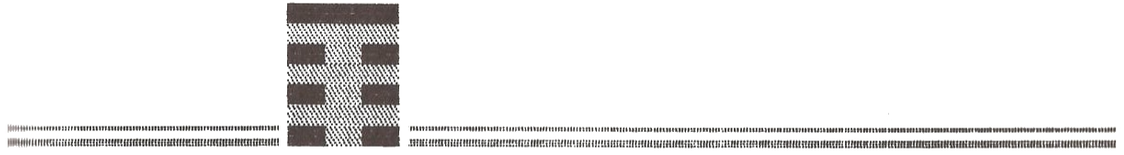
7.2 Considere as engrenagens X, Y e Z com 20, 40 e 100 dentes respectivamente. Se Y completar uma volta horária, X e Z completarão quantas voltas? Em que sentido?




7.3 Alguns bárbaros aprisionaram um sábio e propuseram-lhe:

– Se você disser uma verdade, será queimado, e se disser uma mentira, será afogado. De que maneira prefere morrer?

O sábio deu uma resposta tal que foram obrigados a libertá-lo. Qual foi a resposta?



# Sub-rotinas

*"Eu não abalava a tua fé, como abalaria a fé de uma velha serva da igreja. Lamentava teu destino humilde, que te fazia cega e surda... Mas esta noite, no Saara, sozinho entre a areia e as estrelas, eu te faço justiça."*

*Antoine de Saint-Exupéry*
*(1900-1944)*

## I. INTRODUÇÃO

Normalmente, em programas longos e mais elaborados, existe uma série de informações, instruções ou operações que devem ser feitas repetidas vezes, ao longo do programa. Seria extremamente tediosa essa série, em cada lugar do programa em que isso fosse necessário e, evidentemente, o programa resultante seria extremamente longo.

Por outro lado, durante a preparação de um programa longo, é útil e prático separar, em pequenas partes, esse programa, ficando mais fácil e simples a análise e a depuração.

## II. UMA PRIMEIRA OLHADA

Pelo próprio nome, depreende-se que sub-rotina significa *sub-programa* do programa principal. É uma rotina de trabalho que, quando ativada, executa, sempre, as mesmas operações e instruções.

A sub-rotina é um programa relativamente curto, que desenvolve uma função simples ou uma classe de funções, podendo ainda apresentar possibilidades de ser executada como um programa simples.

Carregue o computador com o programa P8.1; antes, digite **NEW**.

```
1000  REM SETA --->
1010  FOR H = 1 TO 36
1020  HOME
1030  PRINT  TAB( H);"--->"
1040  FOR I = 1 TO 30: NEXT I
1050  NEXT H
1060  END
```

P8.1

**RUN**ize o programa P8.1. O que ele faz?

**R-1**



Carregue o computador, *sem digitar* **NEW**, com o programa P8.2.

```
2000  REM SETA <---
2010  FOR H = 36 TO 1 STEP  - 1
2020  HOME
2030  PRINT  TAB( H);"<---"
2040  FOR I = 1 TO 30: NEXT I
2050  NEXT H
2060  END
```

P8.2

Digite **RUN 2000**. O que aconteceu?

**R-2**

Insira, a seguir, as instruções abaixo.

```
5  REM DEMONSTRACAO DE SUBROTINAS
10  HOME
20  PRINT  TAB( 17);"SETAS"
30  PRINT : PRINT "MOVIMENTO A DIREITA"
40  FOR X = 1 TO 500: NEXT X
50  GOSUB 1000
60  HOME
70  PRINT : PRINT  TAB( 10);"MOVIMENTO A ESQUERDA"
80  FOR X = 1 TO 500: NEXT X
90  GOSUB 2000
100  HOME
110  END
1060  RETURN
2060  RETURN
```

**não é a tecla** *RETURN*

P8.3

Digite **LIST** e anote, abaixo, todo o programa contido no computador.

**R-3**

Veja como o programa P8.3 pode ficar, através do uso de sub-rotinas mais gerais:

```
5  REM DEMONSTRACAO DE SUBROTINAS
10  HOME
20  PRINT  TAB( 17);"SETAS"
30  PRINT : PRINT "MOVIMENTO A DIREITA"
40 P = 500: GOSUB 2000
50 D = 0: GOSUB 1000
60  HOME
70  PRINT : PRINT  TAB( 10);"MOVIMENTO A ESQUERDA"
80 P = 500: GOSUB 2000
90 D = 1: GOSUB 1000
100  HOME
110  END
1000  REM  SETAS
1010  REM D=0 : --->
1020  REM D=1 : <---
1030 A = 1:B = 36:C = 1:S$ = "--->"
1040  IF D THEN A = 36:B = 1:C =  - 1:S$ = "<---"
1050  FOR H = A TO B STEP C
1060  HOME
1070  PRINT  TAB( H);S$
1080 P = 30: GOSUB 2000
1090  NEXT H
1100  RETURN
2000  REM  PAUSA
2010  FOR I = 1 TO P: NEXT I
2020  RETURN
```

P8.4

Rode o programa P8.3. Como foi executado?

**R-4**

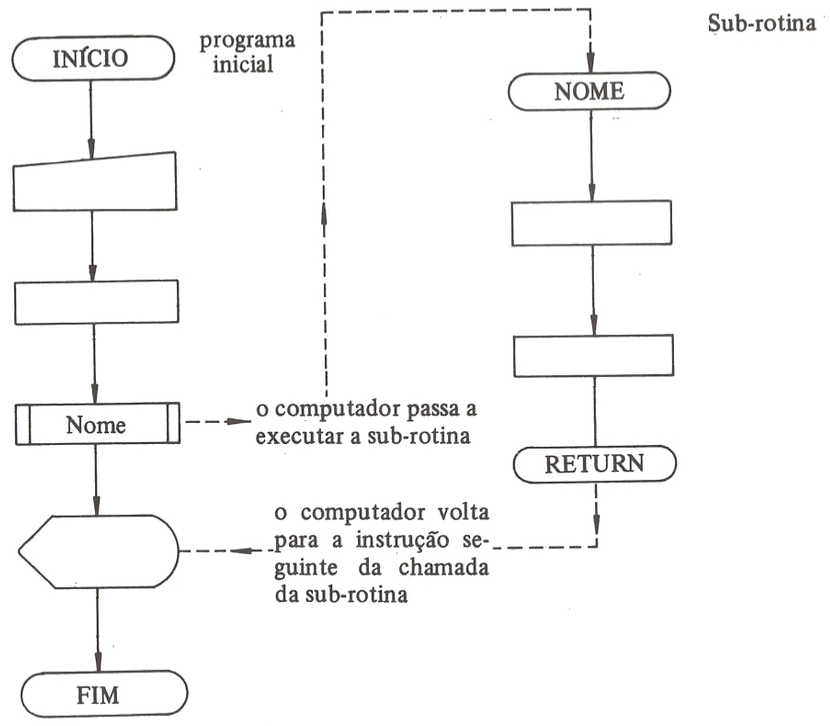
## III. SUB-ROTINAS INCONDICIONAIS

O comando para se executar uma sub-rotina é **GOSUB** e usa como *argumento* o número da instrução em que a sub-rotina se inicia. No entanto, no fluxograma



indicamos a execução, ou saída, de uma sub-rotina pelo símbolo [ nome ] . Analise a figura 8.1, que mostra um fluxograma com sub-rotina e a respectiva codificação.

Fluxograma




Codificação

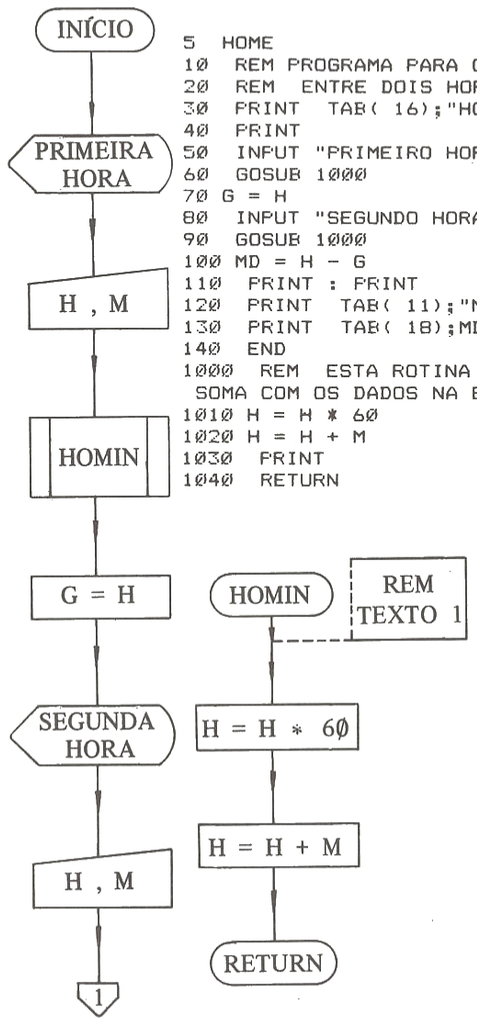
```
10  REM                      1000  REM  ROTINA A
20  HOME : PRINT
30  INPUT ...
 :
230  GOSUB 1000
240  PRINT                   1090  RETURN
 :
700  END
```

fig. 8.1

O comando para executar o retorno para o programa principal, ao final da sub-rotina, é **RETURN**. Não confundir com a tecla *RETURN*.

A seguir temos a fluxogramação e a codificação de um programa que recebe dois horários distintos e calcula o número de minutos decorridos entre eles. A entrada é do tipo *HH, MM* e a saída deve ser *XXX* minutos.

Fluxograma

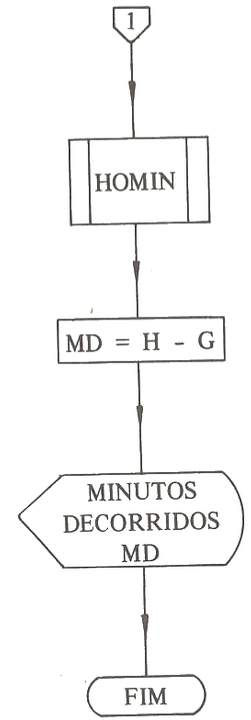



Codificação

```
5  HOME
10  REM PROGRAMA PARA CALCULAR O TEMPO
20  REM  ENTRE DOIS HORARIOS DADOS
30  PRINT  TAB( 16);"HORARIOS"
40  PRINT
50  INPUT "PRIMEIRO HORARIO (HH,MM)? ";H,M
60  GOSUB 1000
70 G = H
80  INPUT "SEGUNDO HORARIO (HH,MM)? ";H,M
90  GOSUB 1000
100 MD = H - G
110  PRINT : PRINT
120  PRINT  TAB( 11);"MINUTOS DECORRIDOS"
130  PRINT  TAB( 18);MD
140  END
1000  REM  ESTA ROTINA TRANSFORMA AS HORAS EM MINUTOS E
 SOMA COM OS DADOS NA ENTRADA
1010 H = H * 60
1020 H = H + M
1030  PRINT
1040  RETURN
```

P8.5






*TEXTO 1:* **ESTA ROTINA TRANSFORMA AS HORAS EM MINUTOS E SOMA-OS AOS MINUTOS DADOS NA ENTRADA.**

**RUN**ize o programa P8.5. Como ele opera?

**R-5**

Faça **RUN 1000** e **GOTO 1000**. O que ocorreu? Por quê?

**R-6**

Em determinados casos, são necessárias sub-rotinas de sub-rotinas, isto é, uma sub-rotina solicitando outra sub-rotina. Nessas condições, dizemos que elas estão enlaçadas. Veja o fluxograma da figura 8.2.

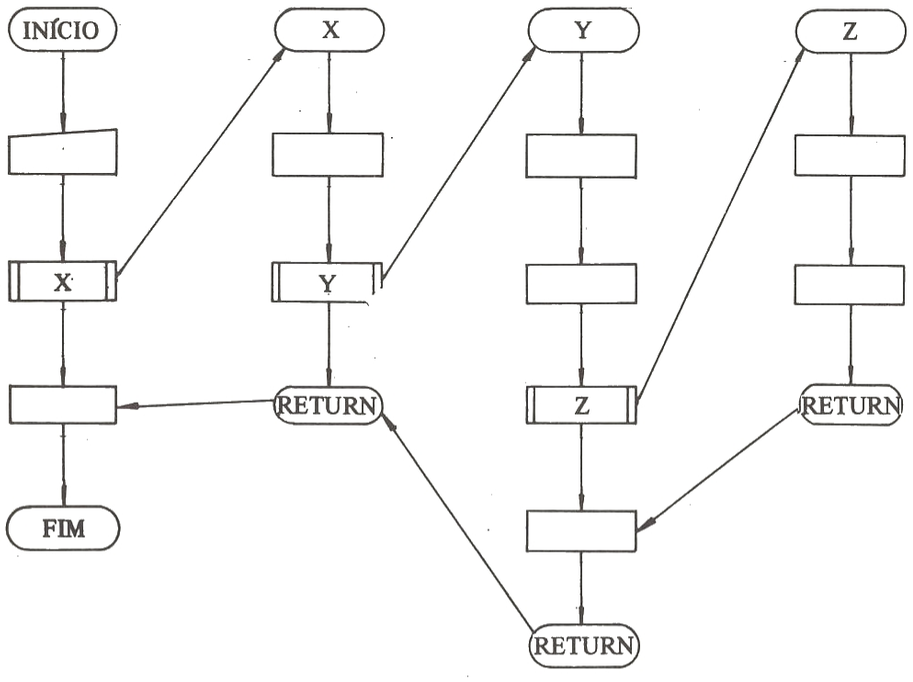
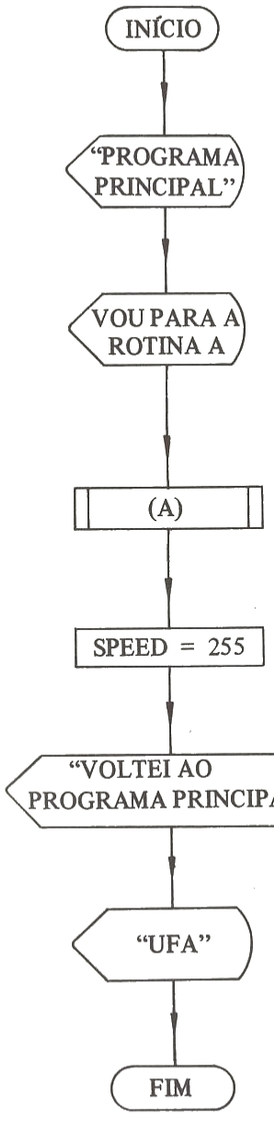



fig. 8.2

**A título de exemplo, analise o fluxograma e o programa P8.6.**



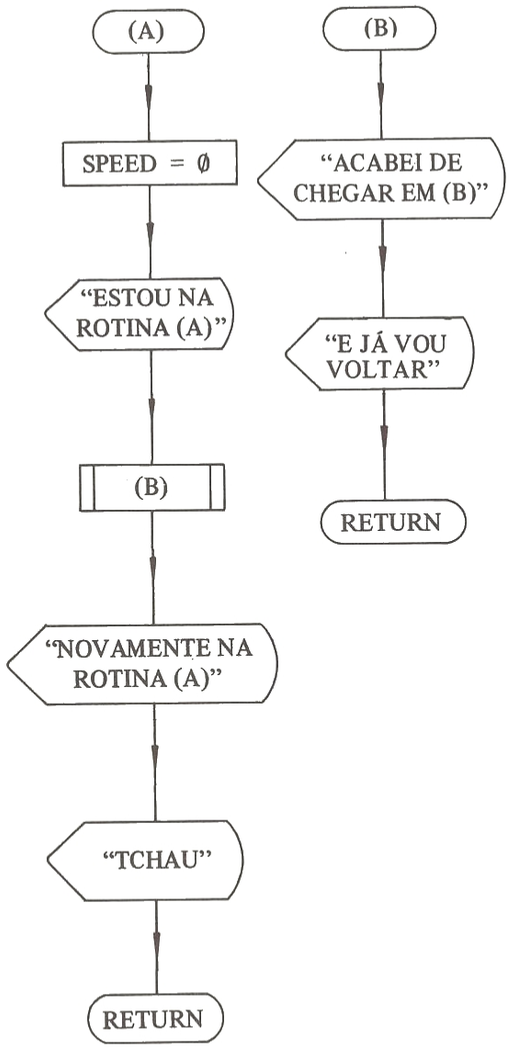
Fluxograma




Codificação

```
10  HOME
20  PRINT "DEMONSTRACAO DE SUBROTINAS ENLACADAS"
30  PRINT : PRINT "ESTE E' O PROGRAMA PRINCIPAL"
35  FOR I = 1 TO 1000: NEXT I
40  PRINT "VOU PARA A ROTINA (A)"
50  PRINT
60  GOSUB 1000
70  SPEED= 255
80  PRINT "VOLTEI AO PROGRAMA PRINCIPAL"
90  PRINT  TAB( 36);"UFA!"
100  END
1000  REM  SUBROTINA (A)
1010  SPEED= 0: INVERSE
1020  PRINT "ESTOU NA ROTINA (A)"
1030  FOR I = 1 TO 100: NEXT I
1040  PRINT : FLASH
1050  GOSUB 2000
1060  PRINT "NOVAMENTE NA ROTINA (A)"
1070  PRINT "TCHAU!"
1080  PRINT : NORMAL
1090  RETURN
2000  REM  SUBROTINA (B)
2010  PRINT "ACABEI DE CHEGAR EM (B)": HTAB 18
2020  PRINT "...E JA' VOU VOLTAR"
2030  PRINT : INVERSE
2040  RETURN
```

P8.6




Nesse programa-exemplo, temos dois enlaces de sub-rotinas. O programa principal consiste nas instruções 1Ø e 1ØØ; a sub-rotina A é definida pelas instruções 1ØØØ a 1Ø9Ø e a sub-rotina B inclui as linhas 2ØØØ a 2Ø4Ø.

Carregue o programa P8.6 no computador e **RUN**ize-o. Qual a resposta?



**R-7**

Em seguida, faça **RUN 1ØØØ**. Qual a resposta?

**R-8**

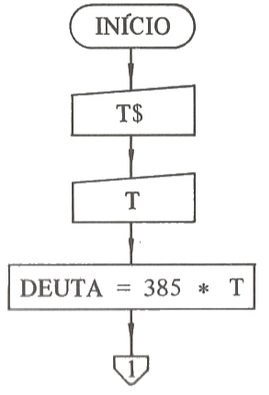
Agora faça **RUN 2ØØØ**. Qual a resposta? Em que mudaria a resposta se fizéssemos **GOTO 2ØØØ**?

**R-9**

Lembre-se: Toda sub-rotina deve terminar com o comando **RETURN**. Experimentemos, agora, um exemplo interessante.

Digite **NEW** e carregue o computador com o programa P8.7.

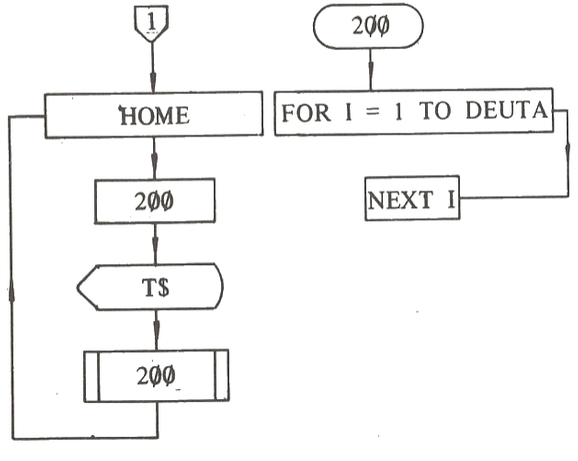
Fluxograma




Codificação

```
100  HOME
110  INPUT "TEXTO A SER VISUALIZADO: ";T$
120  INPUT "TEMPO DE VISUALIZACAO
     (SEGUNDOS): ";T
130 D = 1100 * T
140  HOME
150  GOSUB 200
160  PRINT T$
170  GOSUB 200
180  GOTO 140
200  REM SUBROTINA DO PISCA-PISCA
210  FOR I = 1 TO D: NEXT I
220  RETURN
```

P8.7




Faça o fluxograma correspondente ao programa P8.7; carregue-o e RUNize-o. O que ele faz?

**R-10**

Ao fim de cada visualização, o computador fica disponível? Por quê?

**R-11**

Está ocorrendo algum *loop* contínuo?
Onde?



**R-12**

Qual o procedimento para se obter, novamente, o cursor?

**R-13**

## IV. SUB-ROTINAS CONDICIONAIS

Quando o comando **GOSUB** vier diretamente em uma instrução:

```
10Ø  GOSUB 2ØØØ
```

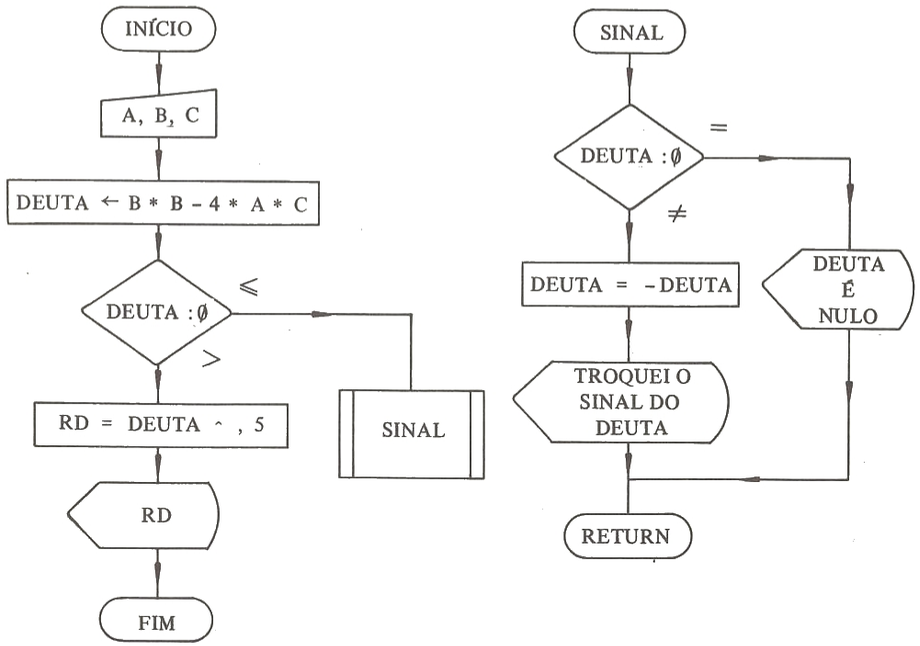
a transferência da execução do programa é feita *incondicionalmente*, isto é, a execução é automaticamente transferida para linha ou instrução de destino (2ØØØ). Porém, em determinados casos, é necessário condicionar o uso da sub-rotina.

```
10Ø  IF A > B THEN  GOSUB 2ØØØ
```

Nessa situação, se A exceder B, a execução é transferida para a rotina 2ØØØ e, ao fim desta, a execução *é recolocada na linha 11Ø.*

Vejamos o programa-exemplo P8.8, cujo fluxograma segue no verso.

Fluxograma




Codificação

```
10  HOME
20  INPUT "DIGITE TRES NUMEROS A,B,C ";A,B,C
30 DEUTA = B * B - 4 * A * C
40  IF DEUTA <  = Ø THEN  GOSUB 1000
50 RD = DEUTA ^ .5
60  PRINT
70  PRINT "RAIZ DO DELTA = ";RD
80  END
1000  REM ROTINA DO SINAL
1010  IF DEUTA = Ø THEN  GOTO 2000
1020 DEUTA =  - DEUTA
1030  PRINT "O SINAL DO DELTA FOI TROCADO"
1040  PRINT
1050  RETURN
2000  PRINT : PRINT  TAB( 25);"O DELTA E' NULO"
2010  GOTO 1040
```

P8.8



Rode P8.8 e escreva o resultado.

**R-14**

Faça **RUN 2000**. Qual a resposta? Era esperada?

**R-15**

## V. DUAS OBSERVAÇÕES FINAIS

Para finalizar esse capítulo convém salientar dois itens importantes em relação às sub-rotinas:

a) Todas as sub-rotinas devem ficar separadas do programa principal e de si mesmas por **END**, **STOP**, **GOTO** ou mesmo pelo próprio **RETURN**, no caso de estar finalizando uma sub-rotina. Digite:

```
10  HOME
20  PRINT "CONVERSOR HEXADECIMAL"
30  FOR N = Ø TO 255: PRINT "$";
40 N% = N / 16
50  GOSUB 120
60 N% = N - N% * 16
70  GOSUB 120
80  PRINT " = ";N;
90  PRINT  SPC( 1 + (N < 100) + (N < 10));
100  NEXT N
110  END
120  PRINT  CHR$(48 + N% + 7 * (N% > 9));
130  RETURN
```

P8.9

Rodando o programa P8.9, teremos uma tabela de números hexadecimais e seus equivalentes decimais. Perceba que o **END** na linha 11Ø não permite que a sub-rotina seja executada sem ter sido dado o comando de desvio **GOSUB**.

Esta colocação você pode ter observado ao analisarmos os programas P8.6 e P8.8.

OBS.: A função **CHR$ (  )** na linha 12Ø será estudada no capítulo 11.

Retire a linha 11Ø. O que acontece? Por quê?

**R-16**

b) Em certos casos é desejável sair de uma sub-rotina através de um desvio absoluto (**GOTO**) e não como é feito, pelo **RETURN** da respectiva sub-rotina. Aí, devemos antes anular o **GOSUB**, mas, é claro, sem voltar ao comando que o segue; esse "**RETURN** sem desvio" é executado pelo comando **POP**.

Este recurso não é muito utilizado, pois uma boa estruturação de sub-rotinas evita saídas no meio da sua execução.

## VI. EXERCÍCIOS PROPOSTOS

8.1 Analise o programa abaixo.

```
1Ø  HOME
2Ø  PRINT "INICIO"
3Ø  GOSUB 7Ø
4Ø  PRINT "DE VOLTA NA ROTINA PRINCIPAL"
5Ø  END
7Ø  PRINT "SUBROTINA 1"
75  RETURN
```

Qual a sua resposta?



8.2 Adicione ao programa do problema 8.1 a instrução:

```
35  GOSUB 7Ø
```

Qual a nova saída? Apague a linha 35.

8.3 Apague a linha 5Ø do problema 8.1. Qual o resultado? Reentre com a instrução 5Ø.

8.4 Adicione as instruções abaixo ao problema 8.1.

```
35  GOSUB 8Ø
8Ø  PRINT "SUBROTINA 2"
85  RETURN
```

Qual a nova saída?

8.5 Adicione ao programa do problema anterior a instrução:

```
72  GOSUB 8Ø
```

Qual a nova saída?

8.6 Apague a linha 85 do programa do problema anterior. Qual a nova saída?

8.7 Usando um contador unitário até 5, como uma sub-rotina e outra sub-rotina contendo um contador decrescente, faça um programa que apresenta a saída:

```
A1    A2    A3    A4    A5

X5    X4    X3    X2    X1

B1    B2    B3    B4    B5

Y5    Y4    Y3    Y2    Y1
```

8.8 Retome o problema anterior e resolva-o, usando, somente, um contador crescente e **VTAB** e **HTAB** para o correto posicionamento.

8.9 Faça uma sub-rotina que, examinando 1Ø números dados, separe e conte os números positivos e apresente a sua soma.

8.10 Usando a idéia do programa P8.7, faça com que o resultado da soma, na saída do problema 1Ø, apareça piscando, aproximadamente, a cada 5 segundos.

8.11 Ainda usando o programa P8.7, estruture um programa para operar de modo semelhante a um relógio digital.

8.12 Faça um programa para resolver equações do 1º e 2º graus e biquadrada. A escolha do "menu" deve ser feita por **GET**; a mensagem, se A = ∅, por sub-rotina e, também, por uma única sub-rotina, o cálculo do *DELTA* para a biquadrada e para a do 2º grau.

8.13 Escreva uma sub-rotina para calcular N!

8.14 Estruture um programa para calcular $e^x$, usando a fórmula:

$$e^x = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \ldots$$

nos casos:

a) com um número finito de termos, a ser definido pelo usuário;

b) com um número infinito de termos (ao pararmos o programa, deve ser possível visualizar o denominador alcançado);

c) fazendo com que a diferença entre o valor atual e o valor atual mais uma fração seja menor que 0,000001 (nessas condições, o programa deve dizer até que expoente do numerador foi feito o cálculo).

## VII. RAXAKUKA

8.1 Seis homens sentaram-se numa mesa redonda, para jogar pôquer. Um deles era irmão de Paulo. Descubra onde eles se sentaram, conhecendo os seguintes fatos:

a) Pedro, que não era parente de Paulo, sentou junto de Jorge.

b) o homem que sentou junto ao homem que sentou do lado oposto ao de Paulo sentou do lado oposto ao irmão de Paulo.

c) Antonio sentou do lado oposto ao de Pedro, que sentou junto ao homem que sentou junto ao homem, que sentou do lado oposto ao de Paulo.

d) Ernâni sentou do lado oposto ao do homem, que sentou junto ao homem, que sentou junto ao homem que sentou do lado oposto ao do irmão de Paulo.

8.2 Na sentença a seguir, determine qual a letra mais distante da primeira letra do alfabeto, na mesma distância em que o segundo O está do primeiro.



## AS MORIBUNDAS NA FLORESTA

8.3 Um elevador sai do térreo com uma pessoa; no andar seguinte entram duas pessoas; no outro, entra uma pessoa e saem duas; no próximo, saem duas e entra uma; no seguinte entram três e, no último, sai uma pessoa. Quantos andares subiu o elevador? Se a carga máxima do elevador é seis pessoas, em que andar houve perigo?

# Arquivo de dados no programa

*"Por conseguinte, ergo a minha taça à razão humana.*
*Que ela nos proteja a todos, negros, amarelos e brancos.*
*Que ela nos tire do vale onde reina a angústia e nos leve a um paraíso de alegrias e de paz."*

*J. M. Simmel*

# I. INTRODUÇÃO

Nós já vimos que, usando **LET** e **INPUT**, atribuímos valores a variáveis numéricas ou alfanuméricas. Em outro capítulo, vimos como essa atribuição era processada mais rápida e facilmente, por meio das variáveis subscritas. Entretanto, em certas circunstâncias, queremos rodar um programa algumas vezes e, em cada situação, com um grupo diferente de dados. O *BASIC* padrão apresenta um conjunto poderoso de comandos para entrada de dados e para atribuição. São eles: **READ**, **DATA** e **RESTORE**.

# II. READ E DATA

O comando **READ** é similar ao comando **INPUT**, porém, as informações (dados) que ele *lê* são fornecidas a partir de uma lista proveniente do comando **DATA** e não proveniente do teclado ou terminal.

Analise o programa abaixo.

```
5  REM  READ/DATA
100  REM  LENDO X,Y,Z
110  READ X,Y,Z
120  HOME
130  VTAB X + 5: PRINT  TAB( X * 10);X
140  VTAB Y + 10: PRINT  TAB( Y * 10);Y
150  VTAB Z + 15: PRINT  TAB( Z * 10);Z
160  GOTO 160
170  DATA 1,2,3
```

P9. 1

Qual a resposta do computador ao programa P9.1?

**R-1**

Podemos dizer, de uma forma simples, que o comando **DATA** guarda, dentro do programa, grandes quantidades de informações que serão lidas pelo comando **READ**, durante a sua execução.



O comando **DATA** não é um comando *executável;* ele serve, apenas, como indicador da lista de dados e pode aparecer em qualquer lugar do programa.

Digite:

```
50  DATA ALFREDO,ABRIL,16
100  READ N$,M$,D
200  HOME
210  VTAB 10
220  PRINT N$: PRINT
230  PRINT "NASCEU EM ";M$;" DE ";1983 - D
240  END
```

P9.2

**RUN**ize o programa P9.2 e analise a sua resposta.

**R-2**

No início da execução do programa, todos os comandos **DATA** são *colecionados* na ordem dos números das linhas e todos os dados ou informações de todos os **DATA**s são colocados numa perfeita seqüência de dados.

Desse modo, na execução de um comando **READ**, são obtidas informações, a partir daquela seqüência de dados, conforme necessário, para as variáveis definidas pelo comando **READ**.

Carregue o computador com o programa P9.3:

```
5  REM  MULTIPLICACOES POR READ/DATA
10  READ D1,D2
20  IF D1 < 0 THEN 200
30  HOME
40  VTAB 5: HTAB 10: PRINT D1;" X ";D2;" = ?"
50  PRINT : INPUT "VALOR? ";R
60  IF R = D1 * D2 THEN  VTAB 20: HTAB 30: PRINT "CERTO
": GOTO 90
70  VTAB 20: PRINT "ERRADO"
80  HTAB 15: PRINT D1;" X ";D2;" = ";D1 * D2
90  FOR I = 1 TO 1000: NEXT I: REM PAUSA
100  GOTO 10
110  DATA 1,5,10,3
120  DATA 4,7,3,2,-1,3
200  END
```

P9.3

Rode o programa P9.3 e verifique seu resultado.

**R-3**

Como o programa escolhe cada par de números?

**R-4**

Como ocorre a auto-interrupção do programa?

**R-5**

Qual a finalidade do último 3 na instrução 12Ø?

**R-6**

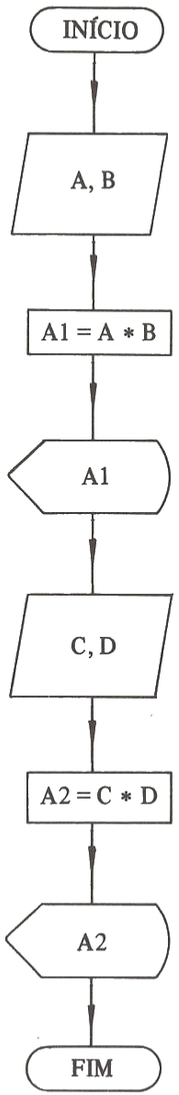
Note que para cada multiplicação apresentada foi escolhido um par de números, a partir do par 1, 5.

O comando **READ** pode ser desdobrado e, ao ser executado, procura, no *arquivo* formado pelos **DATAs**, a respectiva informação.

Faça o fluxograma e a codificação de um programa para calcular a área de dois retângulos, sendo os comprimentos dos lados obtidos por **READ** e **DATA**.



Fluxograma




Codificação

```
5  HOME
1Ø  REM AREA DE 2 RETANGULOS
2Ø  REM LEITURA DOS LADOS A E B
3Ø  READ A,B
4Ø A1 = A * B
5Ø  VTAB 1Ø: HTAB 4
6Ø  PRINT "A AREA DO PRIMEIRO RETANGULO E' ";A1
7Ø  REM LEITURA DOS LADOS C E D
8Ø  READ C,D
9Ø A2 = C * D
1ØØ  VTAB 18: HTAB 4
11Ø  PRINT "A AREA DO SEGUNDO RETANGULO E' ";A2
12Ø  PRINT : PRINT "A1 = ";A;"*";B
13Ø  PRINT : PRINT "A2 = ";C;"*";D
14Ø  DATA 1Ø,3Ø,2Ø,4Ø
```

P9.4

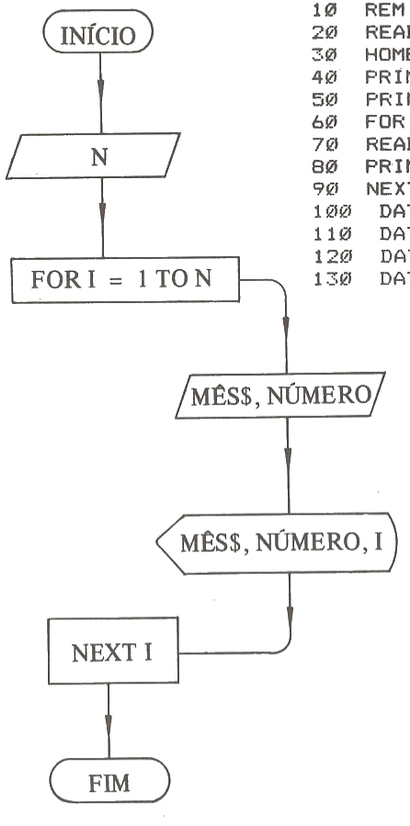
**RUN**ize o programa P9.4. Qual a sua resposta?

É importante salientar que para alterar os valores das informações fornecidas pelo **DATA** devemos modificar, neste programa, a instrução 14Ø.

Os comandos **READ** e **DATA** são muito úteis para a constituição de tabelas com valores fixos, não acontecendo, desse modo, uso excessivo de memória (uso de **LET**).

Analise o programa P9.5.

Fluxograma




Codificação

```
10  REM TABELA ANUAL
20  READ N
30  HOME
40  PRINT "MES","NUMERO DE DIAS"; TAB( 33);"N"
50  PRINT : PRINT
60  FOR I = 1 TO N
70  READ MES$,NUMERO
80  PRINT MES$,NU,I
90  NEXT I
100   DATA 12
110   DATA JAN,31,FEV,28,MAR,31,ABR,30
120   DATA MAI,31,JUN,30,JUL,31,AGO,31
130   DATA SET,30,OUT,31.NOV,30,DEZ,31
```

P9.5



Qual a saída do programa P9.5?

**R-7**

Qual a finalidade do *loop* formado pelas linhas 6Ø - 9Ø - 6Ø?

**R-8**

Note que não foi necessário colocar entre aspas os *strings* do comando **DATA**. No entanto, deve-se tomar cuidado para que eles sejam lidos por variáveis alfanuméricas.

Só é obrigatório colocar os *strings* do **DATA** entre aspas quando estes contiverem "," ou espaços na frente. Exemplo:

**DATA JOAQUIM DA SILVA, "ЖЖЖ SILVA, JOAO DA"**

Os comandos **READ** e **DATA** são muito utilizados para processar informações no teclado, em confronto com um *arquivo* de dados previamente organizado, formando um ótimo conjunto com **FOR ... NEXT** e com as *variáveis subscritas*.

Já vimos que, por meio dos comandos **READ** e **DATA**, podemos organizar um *arquivo* de informações com a vantagem de não usarmos, excessivamente, atribuições de memória. Para podermos ter uma idéia melhor e mais consistente, analisemos o programa P9.6.

```
5  REM CAPITAIS DO NORDESTE
10  HOME
20  PRINT "VOCE SABE AS CAPITAIS DOS ESTADOS DA": PRINT
 "REGIAO NORDESTE?"
30  REM LEITURA DOS ESTADOS E DAS CAPITAIS
40  FOR I = I TO 9
50  READ E$(I),C$(I)
60  NEXT I
70 C = Ø: REM CONTADOR
80  FOR J = 1 TO 9
90  PRINT : PRINT "QUAL A CAPITAL DE ";E$(J);"?"
100   INPUT "";A$
110   IF C$(J) <   > A$ THEN 130
```

```
120 C = C + 1: GOTO 160
130  PRINT : INPUT "NAO, TENTE NOVAMENTE: ";A$
140  IF C$(J) = A$ THEN  PRINT "AGORA VOCE ACERTOU!": G
OTO 160
150  PRINT : PRINT "NAO, A RESPOSTA CORRETA E' ";C$(J)
160  NEXT J
170  PRINT : PRINT : PRINT "VOCE ACERTOU ";C;" EM 9"
180  PRINT : PRINT "ATE' LOGO!!": END
190  DATA MARANHAO,SAO LUIS,PIAUI,TERESINA
200  DATA CEARA,FORTALEZA,RIO GRANDE DO NORTE,NATAL
210  DATA PARAIBA,JOAO PESSOA,PERNAMBUCO,RECIFE
220  DATA ALAGOAS,MACEIO,SERGIPE,ARACAJU,BAHIA,SALVADOR
```

P9.6

Rode o programa acima e verifique o seu resultado.

**R-9**

Qual a finalidade da instrução 7Ø?

**R-1Ø**

O que faz o *loop* 4Ø, 6Ø, 4Ø?

**R-11**

O que faz o *loop* 8Ø, 16Ø, 8Ø?

**R-12**



Toda vez que um comando **READ** é executado, ele procura, por meio do indicador de dados, *DATA POINTER*, a informação em questão. Quando não encontra a informação, o computador gera um erro de execução.

Carregue o computador com o programa P9.7.

```
1Ø  REM ERRO NO DATA
2Ø  READ A$
3Ø  PRINT A$
4Ø  GOTO 2Ø
5Ø  DATA SEG,TER,QUA,QUI
6Ø  DATA SEX,SAB,DOM
```

P9.7

**RUN**ize o programa P9.7. Qual a sua saída?

**R-13**

Qual o significado desse erro?

**R-14**

Como você modificaria o programa P9.7 para eliminar esse erro? Faça a sua listagem.

**R-15**

## II.1 – RESTORE

O comando **READ** vai lendo, em série, as informações da pilha formada pelos comandos **DATA**, como se tivéssemos um ponteiro *DATA POINTER* correndo pela pilha, que indicasse, sempre, a próxima informação a ser lida.

Às vezes é necessário voltar esse ponteiro para o começo da pilha, para que possamos ler novamente, desde o início, a pilha.

Como exemplo, rode o programa P9.8.

```
Ø  REM  REPOSICIONAMENTO
5  HOME
1Ø  READ A,B
2Ø  PRINT A,B
3Ø  RESTORE
4Ø  READ C,D
5Ø  PRINT C,D
6Ø  DATA 1,2,3,4
```

P9.8

Qual a saída? Por quê?

**R-16**

Apague a linha 3Ø e rode, de novo, o programa. Que diferença você observou?

**R-17**

O comando **RESTORE** recoloca o "*ponteiro*" do **READ** no topo (começo) da pilha de informações formada pelos comandos **DATA**.

Imaginemos um programa que tenha, ao todo, 1ØØ informações nos seus comandos **DATA** e, num determinado instante da execução, precisemos da 5Øª ou 5ª informação. O que fazer?

**R-18**



Uma solução seria ler, em seqüência, todas as informações, até chegarmos à desejada. Mas isso pode ser muito lento, principalmente, se houver, no caminho, um elo iterativo (**FOR ... NEXT, IF ... THEN** contador), muitas vezes, para diferentes informações.

Nesses casos, é conveniente, sempre que dispusermos de memória suficiente, transferir todas as informações do **DATA** para matriz cujos elementos possam ser acessados diretamente. (Lembre-se do programa P9.6!)

Analise o programa P9.9.

```
1Ø  REM TABELA ANO X LUCRO
2Ø  DIM LUCRO(1Ø)
3Ø  HOME
4Ø  FOR I = 1 TO 1Ø
5Ø  READ LUCRO(I)
6Ø  NEXT I
7Ø  PRINT : PRINT "DIGITE O ANO PARA SABER O LUCRO": IN
PUT "  (ENTRE 73 E 82): ";ANO
8Ø  IF ANO >  = 73 AND ANO <  = 82 GOTO 1ØØ
9Ø  PRINT : PRINT : PRINT "ANO INVALIDADO": GOTO 7Ø
1ØØ I = ANO - 72
11Ø  PRINT : PRINT "O LUCRO EM ";ANO;" FOI DE CR$";LUCR
O(I)".ØØ"
12Ø  PRINT : GOTO 7Ø
13Ø  DATA 5ØØØØ,1ØØØØØ,12ØØØØ,15ØØØØ,2ØØØØØ,3ØØØØØ,35ØØ
ØØ,4ØØØØØ,8ØØØØØ,1ØØØØØØ
```

P9.9

Qual a função do *loop* das linhas 4Ø - 6Ø - 4Ø?

**R-19**

O que faz a linha 1ØØ e qual a sua função em relação à linha 11Ø?

**R-2Ø**

O que faz a linha 8Ø?

**R-21**

## III. TRACE E NOTRACE

*Debugar* ou depurar é o ato de encontrar è eliminar erros de um programa, sejam eles de sintaxe ou lógica. Para lhe ajudar nessa tarefa, o computador tem o comando **TRACE**.

O comando **TRACE**, realmente, traça a execução do programa, imprimindo o número da linha, quando essa é executada.

Para ver como funciona **TRACE**, digite o programa P9.1Ø.

```
5  HOME
1Ø  PRINT "ESCREVA UM NUMERO DE 1 A 5"
2Ø  INPUT "(6 PARA TERMINAR): ";N: IF N = 6 THEN  END
3Ø  IF N = 1 THEN  PRINT "ONE";: GOTO 9Ø
4Ø  IF N = 2 THEN  PRINT "TWO";: GOTO 9Ø
5Ø  IF N = 3 THEN  PRINT "THREE";: GOTO 9Ø
6Ø  IF N = 4 THEN  PRINT "FOUR";: GOTO 9Ø
7Ø  IF N = 5 THEN  PRINT "FIVE";: GOTO 9Ø
8Ø  END
89  REM VISUALIZAR N ASTERISCOS
9Ø  FOR I = 1 TO N
1ØØ  PRINT " *";
11Ø  NEXT I
12Ø  PRINT : PRINT
21Ø  GOTO 1Ø
```

P9.1Ø

Digite **TRACE** e **RUN** para N = 3, 2 e 6.

Por que algumas instruções (números de linhas) são sempre visualizadas duas vezes seguidas?



**R-22**

Por que as instruções (números) 1ØØ e 11Ø são visualizadas, várias vezes, em cada **RUN**?

**R-23**

Para desligar **TRACE** basta digitar **NOTRACE**.

**TRACE** e **NOTRACE**, também, podem fazer parte do programa.

Graças a **TRACE** e **NOTRACE**, o computador faz a parte que lhe compete na correção de programas; a parte do programador consiste em fazer a depuração.

Como exemplo, veja o programa P9.11.

```
1Ø  REM  TRIANGULO DE ASTERISCOS
2Ø  INPUT "ENTRE UM NUMERO DE UM A DEZ ";N
3Ø  FOR I = N TO 1 STEP  - 1
4Ø  FOR J = 1 TO I
5Ø  PRINT "* ";
6Ø  NEXT J
7Ø  PRINT
8Ø I = I + 1
9Ø  NEXT I
```

P9.11

Rodando-o para N = 4, a saída esperada era:

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

Mas, na verdade, a saída é:

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

Pelo visto, há algum erro no programa. Assim, vamos digitar **TRACE**: **SPEED** = **1ØØ** (para acompanharmos) e **RUN**.

Como sugestão, faça a depuração com lápis e borracha.

| Número da linha do **TRACE** | I | J | N |
|---|---|---|---|
| 1Ø | Ø | Ø | Ø |
| 2Ø | Ø | Ø | 4 |
| 3Ø | 4 | Ø | 4 |
| 4Ø | 4 | 1 | 4 |
| 5Ø | 4 | 1 | 4 |
| 6Ø | 4 | 2 | 4 |
| 5Ø | 4 | 2 | 4 |
| 6Ø | 4 | 3 | 4 |
| 5Ø | 4 | 3 | 4 |
| 6Ø | 4 | 4 | 4 |
| 5Ø | 4 | 4 | 4 |
| 6Ø | 4 | 5 | 4 |
| 7Ø | 4 | 5 | 4 |
| 8Ø | 5 | 5 | 4 |
| 9Ø | 4 | 5 | 4 |

No fim do *loop*, quando é executado o **NEXT I**, o valor de I deveria abaixar de 1, mas continua o mesmo que no começo do *loop*. Antes de ser decrementada, no **NEXT**, a variável I é incrementada na linha 8Ø; portanto, nunca chega a 1 e o *loop* não tem fim; esse é o erro.

Para termos certeza disso podemos inserir uma linha 85:

```
85  PRINT "VALOR DE I ANTES DE SER DECREMENTADO:  ";I
```

Fazemos **NOTRACE** e **RUN**.

Qual o resultado?

**R-24**

Percebemos, pois, que I nunca é decrementado, apesar do **STEP** - 1 da linha 3Ø.



A introdução de **PRINT**s extras, que depois devem ser retirados, é um recurso muito útil no *Debug* dos programas. Outro recurso é colocar um ou mais **STOP** em lugares onde possam estar havendo problemas. Quando o programa dá o **BREAK**, podemos verificar, manualmente, o valor das variáveis, através de **PRINT**.

Retire a instrução 85 e acrescente:

```
35  STOP
```

Faça **RUN** e, quando o programa parar, digite:

```
? I, J
```

**R-25**

A seguir, digite **CONT**.

**R-26**

Repita, várias vezes, o procedimento, até ter certeza de que a variável I não está sendo decrementada como devia.

Você aprendeu algumas técnicas para achar problemas dentro de um programa. Neste programa, basta retirar a linha 8Ø e estará tudo resolvido, mas, geralmente, as soluções não são tão fáceis de se achar.

## IV. EXERCÍCIOS PROPOSTOS

9.1 Desenhe o fluxograma e escreva o programa em *BASIC*, para somar todos os valores de um comando **DATA**. O primeiro valor a ser lido indica o número de valores que devem ser somados a seguir (controle do *loop*).

9.2 Releia o programa P9.8 e imagine que você tenha de fazer um programa para dar a mesma saída e que o computador não tenha memória suficiente para guardar os dados em uma matriz. Faça o fluxograma e a codificação *BASIC* de tal programa. *Dica*: leia a explicação anterior ao programa P9.8.

9.3 Desenhe o fluxograma e escreva um programa em *BASIC* que calcule e imprima a média de todos os valores num bloco **DATA**. Imagine que o último valor do **DATA** seja 999999, servindo, apenas, como aviso do fim de dados e não devendo entrar no cálculo da média.

9.4 Desenhe o fluxograma e escreva um programa em *BASIC* que ordene, em ordem crescente, todos os valores do bloco **DATA**; coloque-os numa matriz indexada e visualize-a. O primeiro valor lido deve indicar o número de informações restantes no bloco **DATA** e devendo fazer parte da matriz.

9.5 Uma importante aplicação dos computadores é no desenho de histogramas (gráficos de barra). Faça o fluxograma e a codificação *BASIC* de um programa para ler 1Ø valores, entre Ø e 2Ø, de um bloco **DATA**, visualizando, para cada valor, o mesmo número de asteriscos na horizontal. A saída deve ser, por exemplo, se **DATA** for 5 e 3:

**GRAFICO**

| **VALOR** | **REPRESENTAÇÃO** |
|---|---|
| 5 | * * * * * |
| 3 | * * * |

OBS.: Coloque números a seu critério (entre Ø e 2Ø).

9.6 Imagine que tenhamos, ao todo, 36 informações numéricas num bloco **DATA**, representando o lucro de cada mês, nos anos de 8Ø, 81, 82. Todos os valores estão em ordem cronológica. Faça o fluxograma e o programa *BASIC* que transfira os dados para uma matriz indexada, colocando as informações sob acesso direto do usuário, bastando que esse dê o ano e o número do mês, para que o computador informe o lucro correspondente.

9.7 O seguinte programa deveria ler os valores do bloco **DATA** e imprimi-los numa matriz 4 x 3, isto é, 4 linhas e 3 colunas.

Usando **TRACE**, teste com lápis e corrija o que achar necessário.

```
1Ø  FOR I = 1 TO 4
2Ø  FOR J = 1 TO 3
3Ø  READ S(J,I)
4Ø  NEXT J
5Ø  NEXT I
6Ø  FOR J = 1 TO 3
7Ø  FOR I = 1 TO 4
```



```
8Ø  PRINT S(I,I);"  ";
9Ø  NEXT I
1ØØ  PRINT
11Ø  NEXT J
12Ø  DATA 23,45,12,34,67,69,34,9Ø,17,56,87,39
```

Exemplo de **RUN**:

**23 67 17 Ø**
**23 67 17 Ø**
**23 67 17 Ø**

9.8 Usando **TRACE**, **PRINTs** e **STOPs**, corrija o programa abaixo, que é um **SORT** numérico de ordem decrescente.

```
1Ø  REM  ORGANIZAR EM ORDEM DECRESCENTE
2Ø  INPUT "NUMERO DE ELEMENTOS ";N
3Ø  FOR I = 1 TO N
4Ø  PRINT "ENTRE ELEMENTO S(";I;") ";
5Ø  INPUT S(I)
6Ø  NEXT I
7Ø  FOR K = 1 TO N - 1
75  REM IMAGINEMOS QUE S(K) SEJA O MAIOR
8Ø  FOR I = K + 1 TO N
9Ø  IF S(J) < S(K) THEN 12Ø
1ØØ  REM E' PRECISO TROCAR
11Ø S(K) = S(I)
12Ø  NEXT I
13Ø  NEXT K
14Ø  PRINT : PRINT "ELEMENTOS EM ORDEM DECRESCENTE"
15Ø  FOR L = 1 TO N: PRINT S(L): PRINT : NEXT L
16Ø  END
```

Exemplo de **RUN**:

**NUMERO DE ELEMENTOS 2**
**ENTRE ELEMENTO S(1) ? 2Ø**
**ENTRE ELEMENTO S(2) ? 4Ø**

**ORDEM DECRESCENTE**
**2Ø**
**4Ø**

## V. RAXAKUKA

9.1 Um sultão das "Mil e Uma Noites" tem 10 sacos, contendo 100 moedas de ouro cada um; mas um dos sacos contém, somente, moedas falsas. A moeda verdadeira tem massa 10g e a moeda falsa 11g. Usando uma balança calibrada em gramas e em uma só aferição, como o sultão pode descobrir o saco de moedas falsas?

9.2 Num jarro, há 7 amebas. Elas se multiplicam tão rapidamente que dobram o seu volume a cada minuto. Se, para encher o jarro, elas levam 40 minutos, quanto tempo levarão para encher metade do jarro?

9.3 Depois de n dias de férias, um estudante observou que:

a) choveu sete vezes; de manhã ou à tarde;
b) quando chove de manhã, não chove à tarde;
c) houve cinco tardes sem chuva;
d) houve seis manhãs sem chuva.

Desse modo, concluímos que o valor de n é _____ dias.

# Juntando IF's e GOTO's

*"Aventurar-se causa ansiedade, mas deixar de arriscar-se é perder a si mesmo... E aventurar-se, no sentido mais elevado, é precisamente tomar consciência de si próprio."*

*Kierkegaard*
*(1813-1855)*

# I. INTRODUÇÃO

O *BASIC (Applesoft)* estruturado e desenvolvido para o nosso computador apresenta comandos de múltipla decisão e comando de decisão após detecção de erros.

Os comandos **ON ... GOTO** e **ON ... GOSUB** permitem economizar vários **IF ... THEN**, pois, como veremos adiante, uma única instrução define várias decisões.

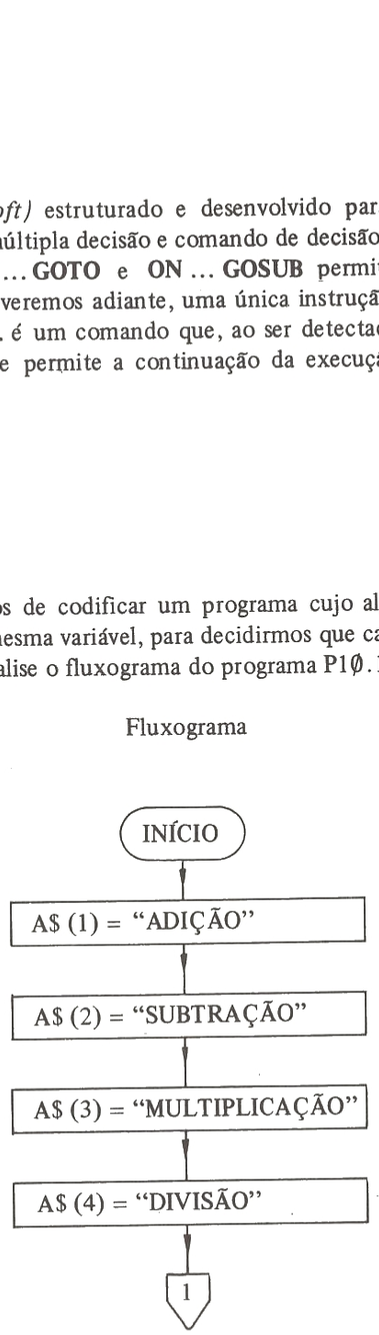
**ON ERR GOTO ...** é um comando que, ao ser detectado um erro, define uma rotina, por exemplo, que permite a continuação da execução do programa, sem o *error break*.

# II. ON ... GOTO

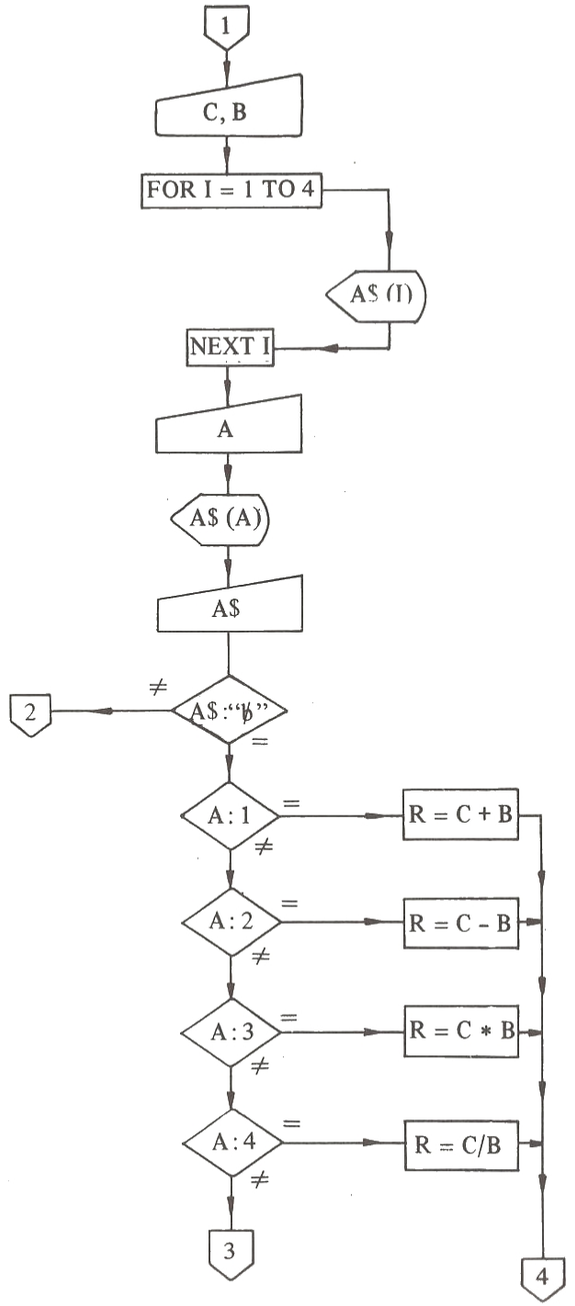
Muitas vezes, temos de codificar um programa cujo algoritmo exige uma série de comparações, com a mesma variável, para decidirmos que caminho seguir.

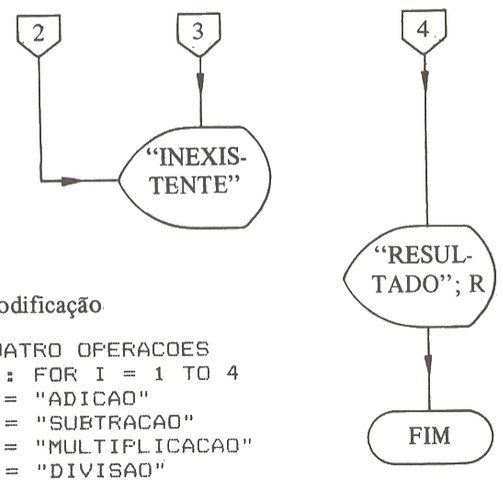
Como exemplo, analise o fluxograma do programa P1Ø.1.

Fluxograma








Codificação

```
5  REM QUATRO OPERACOES
10  HOME : FOR I = 1 TO 4
20 A$(1) = "ADICAO"
30 A$(2) = "SUBTRACAO"
40 A$(3) = "MULTIPLICACAO"
50 A$(4) = "DIVISAO"
60  HOME
70  INPUT "QUAL O PRIMEIRO NUMERO? ";C
80  PRINT
90  INPUT "QUAL O SEGUNDO NUMERO? ";B
95  HOME
100  FOR I = 1 TO 4
110  VTAB I * 2 + 8
120  PRINT I;" - ";A$(I)
130  NEXT I
140  VTAB 20: PRINT "QUE OPERACAO VOCE QUER? ";
150  GET A: PRINT A: IF A > 4 OR A < 1 THEN 240
155  VTAB A * 2 + 8: INVERSE
160  PRINT A;" - ";A$(A): NORMAL
170  VTAB 22: HTAB 5: PRINT "APERTE <ESPACO> PARA CONFI
RMAR ";: GET A$
180  HTAB 1
190  IF A$ <   > " " THEN 95
200  IF A = 1 GOTO 300
210  IF A = 2 GOTO 400
220  IF A = 3 GOTO 500
230  IF A = 4 GOTO 600
240  VTAB 24: PRINT "OPERACAO INEXISTENTE":GOTI = 1 TO
1000: NEXT I: GOTO 95
300 R = C + B: GOTO 700
400 R = C - B: GOTO 700
500 R = C * B: GOTO 700
600 R = C / B
700  INVERSE : HTAB 1: VTAB 24: PRINT "O RESULTADO E' "
;R
710  NORMAL : END
```

**P1Ø.1**



Rode o programa para dois números quaisquer a sua escolha. Qual a saída?

**R-1**

Não há dúvida de que o bloco de instruções das linhas 2ØØ até 23Ø funciona corretamente, mas é, desnecessariamente, comprido e lento.

Substitua as instruções 2ØØ e 23Ø pela instrução:

```
200  ON A GOTO 300,400,500,600
```

Neste caso, a variável A é usada como índice do comando. Se A = 1, a execução é desviada para a linha 3ØØ, se A = 2, a execução vai para a linha 4ØØ, e assim por diante.

Após fazer essa substituição, apagando as linhas 21Ø, 22Ø e 23Ø, rode, novamente, o programa.

Qual a saída? Há alguma diferença em relação ao programa anterior?

**R-2**

Nesse caso, por que usamos variáveis indexadas?

**R-3**

Qual a função do *loop* das linhas 1ØØ - 13Ø - 1ØØ?

**R-4**

O que faz a linha 15Ø?

**R-5**

O que faz a linha 16Ø?

**R-6**

O que faz a linha 2ØØ?

**R-7**

O que faz a linha 17Ø?

**R-8**

Que tecla devemos apertar para confirmar a operação desejada?

**R-9**

Outra tecla serviria? Por quê?

**R-1Ø**



Observações:

I) Se o valor da parte inteira da variável-índice for Ø ou maior que o número de linhas de transferência, mas menor que 256, a execução continua na linha seguinte a **ON ... GOTO**. Se a variável-índice tiver valor menor que Ø ou maior que 255, o computador dará a mensagem:

```
?ILLEGAL QUANTITY ERROR
```

II) **ON ... GOTO** é um recurso próprio do *BASIC*; portanto, não existe um símbolo próprio para sua fluxogramação. Representamos a seqüência **IF ... THEN GOTO**, equivalente.

Rode, a seguir, o programa exemplo P1Ø.2.

```
1Ø  REM LISTA DE PAGAMENTO
2Ø  PRINT "ITEM"; TAB( 8);"QUANTIDADE"; TAB( 2Ø);"VALOR
"; TAB( 3Ø);"TOTAL"
3Ø  READ N
4Ø  FOR J = 1 TO N
5Ø  READ I,Q
6Ø  ON I GOTO 7Ø,9Ø,11Ø,13Ø
7Ø R = 22
8Ø  GOTO 14Ø
9Ø R = 3Ø
1ØØ  GOTO 14Ø
11Ø R = 35
12Ø  GOTO 14Ø
13Ø R = 37
14Ø PR = PR + R * Q
15Ø  PRINT I; TAB( 8);Q; TAB( 2Ø);R; TAB( 3Ø);R * Q
16Ø  NEXT J
17Ø  PRINT
18Ø  PRINT "O TOTAL E' CR$";PR
19Ø  DATA 7
2ØØ  DATA 1,8Ø,3,4Ø,2,1ØØ,1,2Ø,3,2ØØ,2,3Ø,4,2ØØ
```

P1Ø.2

Qual a função do valor lido pelo comando **READ**, na linha 3Ø?

**R-11**

Na linha 2ØØ, os números estão em grupos de 2; o primeiro valor é o código do item de venda e o segundo valor é o número de itens vendidos.

Qual o preço do item de código 1? E dos itens de código 2, 3 e 4?

**R-12**

O que faz a linha 6Ø?

**R-13**

O que faz a linha 14Ø?

**R-14**

## III. ON ... GOSUB

Já vimos como um comando **ON ... GOTO** é usado para desviar a execução do programa, de acordo com o valor de uma certa variável-índice.

Do mesmo modo, podemos usar **ON... GOSUB** para mandar a execução a uma das várias sub-rotinas existentes. A única diferença é que, nesse comando, é executado um **GOSUB** e não um **GOTO**. As observações relativas ao valor das variáveis-índice válidas para **ON ... GOTO** valem, também, para **ON ... GOSUB**.

Exemplo: a seqüência de instruções

```
IF A = 1 THEN  GOSUB 1ØØØ
IF A = 2 THEN  GOSUB 2ØØØ
IF A = 3 THEN  GOSUB 3ØØØ
```



poderia ser substituída por

```
ON A GOSUB 1ØØØ,2ØØØ,3ØØØ
```

Quando a sub-rotina dá o **RETURN**, a execução volta para a instrução seguinte ao **ON ... GOSUB**.

Rode, agora, o programa P1Ø.3.

```
1Ø  REM AREA DE UM RETANGULO
15  PRINT : PRINT
2Ø  INPUT "DE OS LADOS EM POLEGADAS (L1,L2): ";L1,L2
3Ø  PRINT
4Ø  PRINT "SELECIONE A SAIDA EM:"
5Ø  PRINT "1.  CM^2"
6Ø  PRINT "2.  PES^2"
7Ø  PRINT "3.  M^2"
75  PRINT
8Ø  PRINT "QUAL? ";
9Ø  GET A
1ØØ  PRINT A
11Ø  ON A GOSUB 2ØØ,3ØØ,4ØØ
12Ø  REM CALCULO DA AREA
13Ø AREA = L1 * L2: PRINT
14Ø  PRINT "A AREA E' ";AREA;" ";U$
15Ø  END
2ØØ  REM SUBROTINA PARA AREA EM CM^2
21Ø L1 = L1 * 2.54
22Ø L2 = L2 * 2.54
23Ø U$ = "CM^2"
24Ø  RETURN
3ØØ  REM SUBROTINA PARA AREA EM PES^2
31Ø L1 = L1 / 12
32Ø L2 = L2 / 12
33Ø U$ = "PES^2"
34Ø  RETURN
4ØØ  REM SUBROTINA PARA AREA EM M^2
41Ø L1 = L1 * 2.54 / 1ØØ
42Ø L2 = L2 * 2.54 / 1ØØ
43Ø U$ = "M^2"
44Ø  RETURN
```

P1Ø.3

Rode o programa e diga o que acontece, passo a passo, para cada opção escolhida.

**R-15**

O que é U$ ?

**R-16**

Se você selecionar a opção 4, o que acontecerá? Por quê?

**R-17**

Observações:

I) Uma técnica importante, usada tanto em **ON ... GOTO** como em **ON ... GOSUB**, é preparar a variável para ser usada nos comandos.

Digamos que você tenha a seguinte seqüência de comparações com A:

```
IF A = 1Ø GOTO 1ØØØ
IF A = 11 GOTO 2ØØØ
IF A = 12 GOTO 3ØØØ
IF A = 13 GOTO 4ØØØ
```

Para ser usado **ON ... GOTO**, precisamos transformar numa seqüência, começando com o número 1, as comparações.

Assim, fazemos:

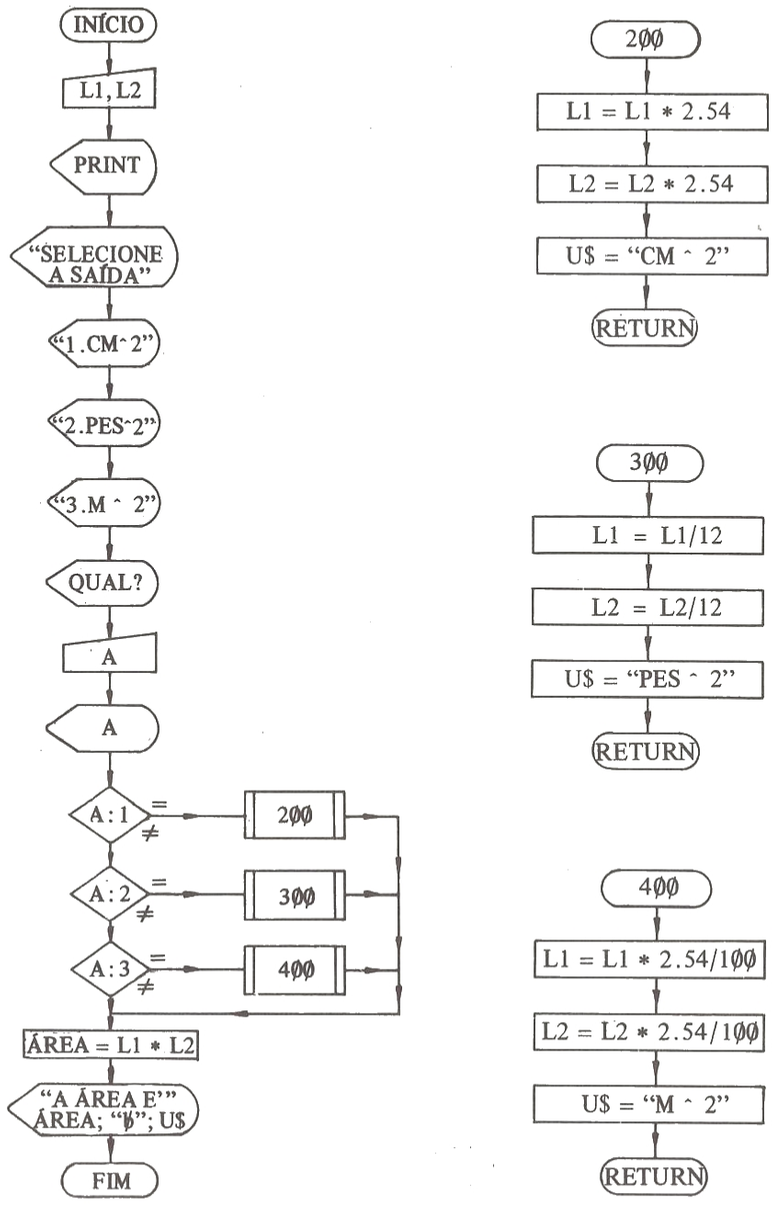
```
ON A - 9 GOTO 1ØØØ,2ØØØ,3ØØØ,4ØØØ
```

Nesse caso, quando A for 1Ø, A - 9 é 1. Portanto, a execução é desviada para a linha 1ØØØ. Quando A for 11, a execução é desviada para a linha 2ØØØ e assim por diante.

II) Assim como **ON ... GOTO**, **ON ... GOSUB** também não tem símbolo próprio de fluxograma. Usamos, então, a seqüência de **IF ... THEN GOSUB**, equivalente.



Fluxograma do programa P1Ø.3



Tendo em mente a primeira das observações da página 232, analise e rode, a seguir, o programa P1Ø.4.

```
1Ø  REM  CALCULO DO DESCONTO NO SALARIO
15  HOME
2Ø  INPUT "QUAL O SALARIO DO FUNCIONARIO? ";SAL
3Ø  IF SAL > 399999 THEN P = 5: GOTO 5Ø
4Ø  ON SAL / 1ØØØØØ + 1 GOSUB 1ØØ,2ØØ,3ØØ,4ØØ
5Ø D = SAL * P / 1Ø
55  PRINT : PRINT
6Ø  PRINT "O DESCONTO NO SALARIO SERA' DE CR$";D
65  PRINT : PRINT "O SALARIO LIQUIDO E' CR$";SAL - D
7Ø  END
1ØØ  REM DESCONTO PARA QUEM GANHA ATE' 99999
11Ø P = 1: RETURN
2ØØ  REM DESCONTO PARA QUEM GANHA DE 1ØØØØØ A 199999
21Ø P = 2: RETURN
3ØØ  REM DESCONTO PARA QUEM GANHA DE 2ØØØØØ A 299999
31Ø P = 3: RETURN
4ØØ  REM DESCONTO PARA QUEM GANHA DE 3ØØØØØ A 399999
41Ø P = 4: RETURN
```

P1Ø.4

Qual o maior desconto que pode ser dado? A partir de que salário é aplicado?

**R-18**

O que faz a linha 4Ø? Para que serve?

**R-19**

O **GOTO 5Ø** da linha 3Ø é necessário? Explique.

**R-2Ø**



Se você der como entrada um salário de 253482 cruzeiros, em que valor ele é transformado na linha 4Ø? O que acontecerá então?

**R-21**

Tanto o comando **ON ... GOTO** como o **ON ... GOSUB** levam em conta, apenas, a parte inteira da variável ou expressão-índice, pois isso economiza trabalho; caso contrário, teríamos de isolar a parte inteira do índice, antes de usá-lo.

Acompanhe, agora, um programa utilizando **ON ... GOTO**, para resolver testes no computador.

```
1Ø  HOME
2Ø  PRINT "QUEM INVENTOU A LAMPADA? "
3Ø  PRINT : PRINT "1 - ISAAC NEWTON"
4Ø  PRINT "2 - MICHAEL FARADAY"
5Ø  PRINT "3 - THOMAS EDISON"
6Ø  PRINT : PRINT "ESCOLHA 1, 2 OU 3: ";
7Ø  INPUT "";A
8Ø  PRINT : ON A GOTO 1ØØ,14Ø,18Ø
1ØØ  PRINT "NAO, ELE ERA MECANICO"
11Ø E = E + 1: GOTO 21Ø
14Ø  PRINT "NAO, ELE INVENTOU O CAPACITOR"
15Ø E = E + 1: GOTO 21Ø
18Ø  PRINT "CORRETO"
19Ø C = C + 1
21Ø  REM CONTINUA COM TESTE NUMERO 2...
```

P1Ø.5

O programa acima pode continuar indefinidamente, com mais testes. Estes, é claro, poderiam ser armazenados em **DATA**, utilizando uma única rotina para fazer as perguntas e dar a resposta.

Como você faria, para aumentar, de 3 para 5, o número de alternativas?

**R-22**

Modifique o programa, de modo que nenhum número escolhido cause erro.

**R-23**

Em que variável é guardado o número de erros? E o número de acertos?

**R-24**

O programa a seguir faz uso de **ON ... GOSUB** e **ON ... GOTO**, para testar a capacidade do aluno ao fazer pequenas multiplicações.

```
1Ø  REM  TESTE DE MULTIPLICACAO
15  HOME
2Ø  INPUT "QUAL.O SEU NOME? ";N$
3Ø  PRINT "MUITO BEM, ";N$;".  HOJE NOS VAMOS TREINAR A
LGUMAS MULTIPLICACOES; APOS A PERGUNTA, DIGITE SUA RESP
OSTA E APERTE <RETURN>."
4Ø  PRINT
5Ø  INPUT "QUANTO E' 5 * 7? ";A
6Ø  IF A = 35 THEN L = 1
7Ø  ON L + 1 GOSUB 1ØØØ,2ØØØ
8Ø  INPUT "QUANTO E' 7 * 6? ";A
9Ø  IF A = 42 THEN L = 1
1ØØ  ON L + 1 GOSUB 1ØØØ,2ØØØ
11Ø  INPUT "QUANTO E' 8 * 4? ";A
12Ø  IF A = 32 THEN L = 1
13Ø  ON L + 1 GOSUB 1ØØØ,2ØØØ
14Ø  ON C GOTO 2ØØ,3ØØ,4ØØ
15Ø  PRINT "E' UMA PENA QUE VOCE NAO TENHA ACERTADO NEN
HUMA.  VOCE DEVE ESTUDAR MELOR A TABUADA!": END
2ØØ  PRINT "VOCE SO ACERTOU UMA QUESTAO, ESTUDE SUA TAB
UADA!": END
3ØØ  PRINT "2 CERTOS EM 3.  ESTA' BOM, MAS PROCURE ACER
TAR TODOS DA PROXIMA VEZ!": END
4ØØ  PRINT "PARABENS, VOCE ACERTOU TODOS!!": END
1ØØØ  PRINT "CONCENTRE-SE MAIS, ";N$;", TENHO CERTEZA D
E QUE VOCE PODE FAZER MELHOR."
1Ø1Ø  PRINT : PRINT
1Ø2Ø  RETURN
2ØØØ  PRINT "PARABENS, ";N$;"; VOCE ACERTOU!!"
2Ø1Ø C = C + 1
2Ø2Ø L = Ø: PRINT : PRINT
2Ø3Ø  RETURN
```

P1Ø.6



Por que fazemos **ON L + 1 GOSUB 1ØØØ, 2ØØØ**?

**R-25**

Por que fazemos **L = Ø** na linha 2Ø2Ø?

**R-26**

Qual a função da variável C?

**R-27**

Para o usuário, quais as principais diferenças, entre esse programa e o P1Ø.5?

**R-28**

O programa P1Ø.6 apresenta uma grande diferença em relação ao P1Ø.5 – é interativo, pois menciona o nome do aluno, dá-lhe parabéns e estimula-o a estudar.

Programas interativos são muitos úteis, quando o usuário não conhece o funcionamento do computador.

## IV. ON ERR GOTO ...

Vamos nos lembrar do capítulo sobre **READ, DATA** e dos programas com que trabalhamos. Em todos eles, sabíamos, exatamente, quantos valores íamos ler através do comando **READ**. Vejamos, agora, o programa P1Ø.7.

```
1Ø  REM  TABELA ANO X LUCRO
2Ø  HOME
3Ø  PRINT "ANO","LUCRO"
4Ø  READ ANO,LUCRO
5Ø  PRINT
6Ø  PRINT ANO,LUCRO
7Ø  GOTO 4Ø
8Ø  DATA 75,5ØØØØ,76,8ØØØØ,77,9ØØØØ
9Ø  DATA 78,12ØØØØ,79,18ØØØØ,8Ø,3ØØØØØ
1ØØ  DATA 81,5ØØØØØ,82,1ØØØØØØ
```

P1Ø.7

Esse programa tem suas linhas de **DATA**, sempre, mudadas e, portanto, não dá para saber o número exato de informações.

Rode o programa.

Qual a mensagem de erro? Qual o seu significado?

**R-29**

Por que aconteceu o erro?

**R-3Ø**

Nesse caso, o único inconveniente foi a mensagem de erro, mas, se o programa acima fosse parte de um programa maior, este teria sua execução interrompida.

Podemos evitar mensagens e interrupções através do comando:

**ON ERR GOTO número de linha.**

Quando executado, esse comando aciona um *flag* (indicador) interno da máquina, que diz ao computador para ir até a linha especificada, caso ocorra algum erro.

Introduza as seguintes linhas ao programa anterior:

```
25  ONERR  GOTO 11Ø
11Ø  PRINT
12Ø  PRINT "FIM DA TABELA"
13Ø  END
```



Rode o programa. Que diferença você observou? Por quê?

**R-31**

O comando **ON ERR** pode ser usado sempre que se espera que algum erro ocorra, mas devemos ter ciência do erro e certeza de que não ocorrerão outros erros comuns, como? **SYNTAX ERROR**, pois esses, também, desviarão a execução do programa para a linha especificada.

Um comando que pode ser executado junto com **ON ERR** é o **RESUME**. Esse comando recomeça a execução do programa, a partir da linha onde ocorreu o erro.

Rode o programa P1Ø.8, que calcula o valor de 1/x, dando x.

```
1Ø  REM CALCULO DO RECIPROCO
2Ø  ONERR  GOTO 7Ø
3Ø  PRINT : INPUT "QUAL O VALOR DE X? ";X
4Ø R = 1 / X
5Ø  PRINT "O RECIPROCO DE ";X;" E' ";R
6Ø  GOTO 3Ø
7Ø  IF X = Ø THEN X = 1E - 37
8Ø  PRINT "ATENCAO:  Ø FOI SUBSTITUIDO POR 1E-37!!"
9Ø  RESUME
```

P1Ø.8

Rode o programa para X = Ø. Qual a saída? O que fazem as linhas 2Ø, 7Ø e 8Ø?

**R-32**

Interrompa o programa no **INPUT** da linha 3Ø com um *CTRL-C*. O que aconteceu?

**R-33**

*CTRL-C* é considerado um erro de interrupção de programa. Se não houver *flag* (**ON ERR acionado**), ele interrompe a execução do programa, mostrando o número da linha em questão.

Adicione a linha:

```
75  IF  PEEK (222) = 255 THEN  PRINT "TERMINADO": END
```

Rode o novo P1Ø.8. Qual a mensagem para **CTRL-C**?

**R-34**

A função **PEEK (222)** devolve o código do erro ocorrido na execução do programa. A tabela 1 mostra os códigos e suas respectivas mensagens de erro:

Tabela 1

| CODIGO | MENSAGEM | CODIGO | MENSAGEM |
|---|---|---|---|
| Ø | NEXT WITHOUT FOR | 12Ø | REDIMENSIONED ARRAY |
| 16 | SYNTAX | 133 | DIVISION BY ZERO |
| 22 | RETURN WITHOUT GOSUB | 163 | TYPE MISMATCH |
| 42 | OUT OF DATA | 176 | STRING TOO LONG |
| 53 | ILLEGAL QUANTITY | 191 | FORMULA TOO COMPLEX |
| 69 | OVERFLOW | 224 | UNDEFINED FUNCTION |
| 77 | OUT OF MEMORY | 254 | BAD RESPONSE TO AN INPUT |
| 9Ø | UNDEFINED STATEMENT | 255 | CTRL C INTERRUPT ATTEMPTED |
| 1Ø7 | BAD SUBSCRIPT | | |

O comando **ON ERR** também pode interceptar erros do **DOS**, quando estiver ativado. A tabela 2 mostra os códigos das mensagens de erro do **DOS**.



Tabela 2

| CODIGO | MENSAGEM | CODIGO | MENSAGEM |
|---|---|---|---|
| 1 | LANGUAGE NOT AVAILABLE | 9 | DISK FULL |
| 2,3 | RANGE ERROR | 1Ø | FILE LOCKED |
| 4 | WRITE PROTECTED | 11 | SYNTAX ERROR |
| 5 | END OF DATA | 12 | NO BUFFERS AVAILABLE |
| 6 | FILE NOT FOUND | 13 | FILE TYPE MISMATCH |
| 7 | VOLUME MISMATCH | 14 | PROGRAM TOO LARGE |
| 8 | I/O ERROR | 15 | NOT DIRECT COMMAND |

Adicione ao programa P1Ø.8 a instrução

```
25  PRUNT
```

Rode o programa. Qual a nova saída? Por quê?

**R-35**

Retire a linha 2Ø e rode, novamente, o programa. O que ocorreu?

**R-36**

Um erro comum em programas matemáticos é divisão por zero. Obviamente, não existe divisão por zero, mas, em muitos casos, é conveniente substituir o Ø por um número muito pequeno, para que a divisão possa continuar.

*Devemos fazer o programa avisar sempre que tal substituição for feita.*

Analise o programa P1Ø.9:

```
1Ø  REM TESTE DO ONERR
2Ø  ONERR  GOTO 1ØØØ
3Ø  DIM A(5)
4Ø  HOME
5Ø N% = 1
6Ø  READ S$
7Ø  PRINT S$;": ";
8Ø  INPUT "";A(N%)
9Ø N% = N% + 1: GOTO 6Ø
1ØØ  DATA IDENTIDADE,TELEFONE,IDADE,DIA DE NASCIMENTO,N
UMERO DA SORTE
1ØØØ P =  PEEK (222): PRINT "ERRO: ";P
1Ø1Ø  IF P = 255 THEN  PRINT "FIM": END
1Ø2Ø  IF P = 254 THEN  PRINT "RESPOSTA ILEGAL": RESUME
1Ø3Ø  IF P = 42 THEN  PRINT : PRINT "FIM DAS PERGUNTAS"
1Ø4Ø  END
```

P1Ø.9

Rode o programa e responda com erros eventuais (p. ex., só pressionar *RETURN*, *CTRL-C*).

Quais as funções das linhas 1Ø1Ø, 1Ø2Ø, 1Ø3Ø?

**R-37**

A rotina de erro prevê três tipos de erro:

– interrupção por *CTRL-C*;
– resposta ilegal a um **INPUT** numérico;
– fim de dados.

Altere a linha 3Ø para:

```
3Ø  DIM A(2)
```

e rode o programa novamente. Qual o novo comportamento do programa?



**R-38**

Qual o significado do erro 1Ø7?

**R-39**

## V. EXERCÍCIOS PROPOSTOS

10.1 Uma loja vende cinco produtos diferentes, cujos preços estão mostrados na seguinte tabela:

| número do produto | nome | preço |
|---|---|---|
| 1 | sabão | 2ØØ.ØØ |
| 2 | tinta | 4ØØ.ØØ |
| 3 | pasta | 7ØØ.ØØ |
| 4 | solda | 2ØØØ.ØØ |
| 5 | fio especial | 5ØØØ.ØØ |

Estruture um fluxograma e o programa em *BASIC* que faça o **INPUT** de uma série de pares de números, a saber:

1. o número do produto
2. a quantidade vendida

Como saída, deseja-se uma tabela com o número, nome do produto, quantidade vendida, total de vendas de cada produto e total geral de vendas.

O computador deverá fornecer, a pedido do usuário, a quantidade em estoque de cada produto e avisar, se possível, a quantidade pedida ou quando terminar. Obs. A quantidade inicial dos produtos é 1ØØØ unidades.

10.2 Faça o fluxograma e o programa *BASIC* para imprimir a equação horária de movimento (*MRUV*) de um móvel, utilizando as unidades metro e segundo. Os dados de entrada são $s_0$, $v_0$ e a. As unidades de entrada para comprimento são: centímetro, polegada e pé; para tempo são: segundo, minuto, hora.

O computador deve perguntar a unidade desejada para resposta e pedir o instante, fornecendo posição e velocidade ou pedir a posição fornecendo velocidade e instante.

É obrigatório o uso de **ON ... GOSUB**, para a conversão, e **ON ... ERR** para não parar o programa.

10.3 Faça o fluxograma e o programa *BASIC* que, ao ler os valores de um bloco **DATA**, valores inteiros de 1 a 5, some-os com as seguintes variáveis:

A se o valor do **DATA** for 1
B se o valor do **DATA** for 2
C se o valor do **DATA** for 3
D se o valor do **DATA** for 4
E se o valor do **DATA** for 5

No final, deverá imprimir o valor de cada variável, bem como o número de valores somados em cada uma delas. Use **ON ... GOTO**. Você não sabe quantas variáveis serão lidas, mas deve evitar mensagens de erro e interrupção do programa.

10.4 Faça um programa que calcule, em litros, o volume de um certo sólido. A fórmula de seu volume é:

$$V = \frac{1}{3}\pi \cdot r^2 \cdot \frac{h}{2} + \frac{4}{6}\pi \cdot r^2 + \pi \cdot r^2 \cdot g$$

em que r, h e g são as variáveis do problema e podem ser dadas em polegadas, pés, centímetros, jardas ou braças. Use **ON ... GOSUB**.

Dados: 1 polegada = 25.4 mm
1 pé = 12 polegadas
1 jarda = 3 pés
1 braça = 2 jardas

10.5 Faça um fluxograma e o programa em *BASIC* que estabeleça a função:

$$Y = \frac{(x^2 + 4x + 4)(x - 2)}{(x - 2)}$$



de $-1\emptyset$ até $+1\emptyset$. Evite mensagens de erros e interrupções, e, usando **ON ERROR**, verifique se o erro ocorrido foi o esperado.

10.6 Faça uma sub-rotina para que, após detectar somente erros de divisão por zero ou raiz quadrada de número negativo, o computador coloque uma mensagem na tela e apresente ao operador possibilidade de continuar ou não o processamento.

## VI. RAXAKUKA

10.1 O piloto de um avião levantou vôo de um certo lugar e voou, para o sul, 657 km, parando num aeroporto de uma cidade. Dali, voou para oeste, na direção de uma cidade a 1.078 km. Depois, voou para o sul, de volta ao lugar de onde tinha saído. Como isso é possível e qual foi o seu ponto de partida?

10.2 Dois povos utilizam calendários diferentes, nos quais os dias têm a mesma duração, mas o ano não. Um deles comemora a "passagem do ano" a cada 360 dias e o outro a cada 324 dias. Supondo que o início da "marcação dos tempos" dos dois povos tenha coincidido, após quantos dias desse início coincidiu, pela primeira vez, o dia de comemoração da "passagem do ano" para os dois povos?

10.3 O guarda-freios, o maquinista e o foguista de um trem chamam-se Davier, Smith e Jones, não respectivamente. No mesmo trem, viajam três passageiros, que têm nomes idênticos: Mr. Jones, Mr. Smith e Mr. Davier.

Sabe-se que:

a) Mr. Davier mora em Detroit;

b) o guarda-freios mora no meio do caminho, entre Detroit e Chicago;

c) Mr. Jones ganha, exatamente, 2.000 dólares por mês;

d) Smith vence o foguista no bilhar;

e) o mais próximo vizinho do guarda-freios, um dos passageiros, ganha, exatamente, três vezes mais que o guarda-freios;

f) o passageiro que tem o mesmo nome que o guarda-freios mora em Chicago.

Pergunta-se: Qual o nome do maquinista?

# Processamento de caracteres e manipulação de **strings**

*"... uma emoção que parece nascer quando uma experiência vem desmentir um mundo de concepções já suficientemente arraigadas em nós. Sempre que uma tal contradição é sentida com força e intensidade, experimentamos uma reação decisiva na maneira de interpretar o mundo. O desenvolvimento dessa interpretação é, em certo sentido, como um vôo contínuo a partir da surpresa."*

*Albert Einstein*
*(1879-1955)*

# I. INTRODUÇÃO

O grande poder do computador reside, sem dúvida, na habilidade de não só processar informações numéricas mas também de processar e manipular informações alfanuméricas (caracteres).

Informações alfanuméricas consistem em combinações de caracteres numéricos, alfabéticos, de pontuação e de simbologia. Um *string* é uma seqüência específica de alguns caracteres.

Internamente o computador gera cada caractere como sendo um número, entre Ø e 255, para então ser decodificado e, a seguir, visualizado.

Para uma padronização internacional do código de geração de caracteres (imagine que confusões ocorreriam se cada computador tivesse um código em particular) foi criado o ASCII, que é usado na maioria dos computadores.

A sigla ASCII significa *American Standard Code for Information Interchange* e o código ASCII usado no nosso computador segue na tabela 1.

Tabela 1

| DEC | HEX | CHAR | DIGITE |
|---|---|---|---|
| Ø | ØØ | NULL | ctrl @ |
| 1 | Ø1 | SOH | ctrl A |
| 2 | Ø2 | STX | ctrl B |
| 3 | Ø3 | ETX | ctrl C |
| 4 | Ø4 | ET | ctrl D |
| 5 | Ø5 | ENQ | ctrl E |
| 6 | Ø6 | ACK | ctrl F |
| 7 | Ø7 | BEL | ctrl G |
| 8 | Ø8 | BS | ctrl H ou ← |
| 9 | Ø9 | HT | ctrl I |
| 1Ø | ØA | LF | ctrl J |
| 11 | ØB | VT | ctrl K |
| 12 | ØC | FF | ctrl L |
| 13 | ØD | CR | ctrl M ou RETURN |
| 14 | ØE | SO | ctrl N |
| 15 | ØF | SI | ctrl O |
| 16 | 1Ø | DLE | ctrl P |
| 17 | 11 | DC1 | ctrl Q |
| 18 | 12 | DC2 | ctrl R |
| 19 | 13 | DC3 | ctrl S |



| DEC | HEX | CHAR | DIGITE |
|---|---|---|---|
| 2Ø | 14 | DC4 | ctrl T |
| 21 | 15 | NAK | ctrl U ou → |
| 22 | 16 | SYN | ctrl V |
| 23 | 17 | ETB | ctrl W |
| 24 | 18 | CAN | ctrl X |
| 25 | 19 | EM | ctrl Y |
| 26 | 1A | SUB | ctrl Z |
| 27 | 1B | ESCAPE | ESC |
| 28 | 1C | FS | n/d |
| 29 | 1D | GS | ctrl shift-M |
| 3Ø | 1E | RS | ctrl ^ |
| 31 | 1F | US | n/d |
| 32 | 2Ø | SPACE | espaço |
| 33 | 21 | ! | ! |
| 34 | 22 | " | " |
| 35 | 23 | # | # |
| 36 | 24 | $ | $ |
| 37 | 25 | % | % |
| 38 | 26 | & | & |
| 39 | 27 | ' | ' |
| 4Ø | 28 | ( | ( |
| 41 | 29 | ) | ) |
| 42 | 2A | * | * |
| 43 | 2B | + | + |
| 44 | 2C | , | , |
| 45 | 2D | – | – |
| 46 | 2E | . | . |
| 47 | 2F | / | / |
| 48 | 3Ø | Ø | Ø |
| 49 | 31 | 1 | 1 |
| 5Ø | 32 | 2 | 2 |
| 51 | 33 | 3 | 3 |
| 52 | 34 | 4 | 4 |
| 53 | 35 | 5 | 5 |
| 54 | 36 | 6 | 6 |
| 55 | 37 | 7 | 7 |
| 56 | 38 | 8 | 8 |
| 57 | 39 | 9 | 9 |
| 58 | 3A | : | : |
| 59 | 3B | ; | ; |

| DEC | HEX | CHAR | DIGITE |
|---|---|---|---|
| 6Ø | 3C | < | < |
| 61 | 3D | = | = |
| 62 | 3E | > | > |
| 63 | 3F | ? | ? |
| 64 | 4Ø | @ | @ |
| 65 | 41 | A | A |
| 66 | 42 | B | B |
| 67 | 43 | C | C |
| 68 | 44 | D | D |
| 69 | 45 | E | E |
| 7Ø | 46 | F | F |
| 71 | 47 | G | G |
| 72 | 48 | H | H |
| 73 | 49 | I | I |
| 74 | 4A | J | J |
| 75 | 4B | K | K |
| 76 | 4C | L | L |
| 77 | 4D | M | M |
| 78 | 4E | N | N |
| 79 | 4F | O | O |
| 8Ø | 5Ø | P | P |
| 81 | 51 | Q | Q |
| 82 | 52 | R | R |
| 83 | 53 | S | S |
| 84 | 54 | T | T |
| 85 | 55 | U | U |
| 86 | 56 | V | V |
| 87 | 57 | W | W |
| 88 | 58 | X | X |
| 89 | 59 | Y | Y |
| 9Ø | 5A | Z | Z |
| 91 | 5B | [ | n/d |
| 92 | 5C | \ | n/d |
| 93 | 5D | ] | ] (shift-M) |
| 94 | 5E | ^ | ^ |
| 95 | 5F | _ | n/d |

OBSERVAÇÕES:

**DEC**: código ASCII decimal.
**HEX**: código ASCII hexadecimal.

**CHAR**: caractere ASCII.
**n/d**: não disponível diretamente do teclado do Apple.



Códigos ASCII na faixa entre 96 e 255 gerarão, no Apple, caracteres que repetem aqueles na lista anterior (primeiro os da coluna dois, depois a lista inteira novamente). Mesmo sendo os caracteres 65 e 193 o símbolo "A", o Applesoft não reconhece os dois como sendo o mesmo caractere quando são comparados com operadores lógicos, e uma impressora os imprimiria de maneiras diferentes.

## II. O CÓDIGO ASCII

Para visualizarmos diretamente um caractere do teclado, basta digitarmos o comando **PRINT** seguido do caractere entre aspas desejado.

```
PRINT "A"
```

**R-1**

Porém esse mesmo caractere pode ser visualizado por meio da função **CHR$ ( )**. Digite:

```
PRINT CHR$(65)
```

**R-2**

Faça agora:

```
PRINT ASC("A")
```

**R-3**

A função **CHR$ ( )** apresenta o caractere especificado pelo código numérico ou pelo resultado de uma expressão numérica, que deve vir entre parênteses. O conteúdo dos parênteses é denominado *argumento* da função.

A função inversa do **CHR$ ( )** é o **ASC ( )**. Ela apresenta o correspondente numérico do código ASCII somente do primeiro caractere do *string* considerado.

Verifique a saída dos seguintes comandos:

```
PRINT CHR$(ASC("A"))
```

**R-4**

```
PRINT ASC(CHR$(65))
```

**R-5**

```
I=65
PRINT CHR$(I)
```

**R-6**

```
A$ = "POR DENTRO DO APPLE"
B$ = "P"
PRINT ASC(A$), ASC(B$)
```



**R-7**

Como uma primeira interação com as funções **CHR$ ( )** e **ASC ( )**, analise o programa P11.1.

```
5  REM  LISTAGEM DE "ASC" E "CHR$"
10  HOME
20  FOR I =  ASC (" ") TO  ASC ("Z")
30  PRINT I;" --> "; CHR$ (I),
40  FOR T = 1 TO 200: NEXT T
50  NEXT I
60  END
```

P11.1

Descreva a saída do P11.1 antes de rodá-lo.

**R-8**

Rode o programa P11.1. Use *CTRL-S* para interromper sua saída. Usando o programa P11.1 e a tabela 1, diga quais os caracteres que não estão presentes no teclado mas que podem ser visualizados através da função **CHR$ ( )**.

**R-9**

Você deve ter obtido como resposta os caracteres:

| | |
|---|---|
| [ | ASCII - 91 |
| \ | ASCII - 92 |
| _ | ASCII - 95 |

Digite o seguinte comando direto:

```
HOME : PRINT "POR DENTRO DO"; CHR$(32); CHR$(65);
   CHR$(80); CHR$(80); CHR$(76); CHR$(69); CHR$(32);
   CHR$(93); CHR$(91)
```

Pela tabela ASCII facilmente percebemos que os códigos de Ø a 31 correspondem aos *caracteres de controle.* O código Ø corresponde a digitar *CTRL-@* ; o código 1 corresponde a *CTRL-A* ; 2 a *CTRL-B* e assim por diante.

As letras que aparecem na coluna **CHAR** da tabela 1 são observações de funções normalmente exercidas por esses códigos.

É importante notar que os códigos 28 e 31 não estão presentes no teclado e que o código 27 (ESCAPE) é acessível através da tecla *ESC.*

## III. CTRL - CONTROL

Os caracteres de controle que no momento analisaremos são: G (7); H (8); M (13); J (1Ø); U (15) e X (18).

Digite:

```
PRINT CHR$(7)
```

**R-1Ø**

A sigla BEL na coluna **CHAR** (*CTRL-G*) é a abreviação de *Bell.*
Faça agora:

```
PRINT ""
```
(*CTRL-G* deve estar entre aspas)

**R-11**

```
B$ = ""
```
(*CTRL-G* deve estar entre aspas)
```
PRINT B$
```



**R-12**

Em vez de digitarmos **PRINT CHR$** (*código*), podemos também fazer **PRINT** "*CTRL letra*", pois o efeito será o mesmo. Utilizaremos, de um modo geral, a primeira forma por ser mais clara, e em certos programas definiremos o *CTRL letra* por meio de uma variável *string.*

Rode, a seguir, o programa P11.2.

```
5  REM  TESTE DE "CTRL-H"
1Ø  HOME
2Ø  VTAB 1Ø
3Ø  PRINT "ABCDEFG";
4Ø  GET A$
5Ø  PRINT  CHR$ (8);
6Ø  PRINT A$
7Ø  END
```

P11.2

Qual a saída fornecida pelo computador?

**R-13**

Qual a saída se não houvesse a linha 5Ø?

**R-14**

Retire a linha 5Ø e confirme sua resposta.
**RUN**ize a seguir o programa P11.3.

```
10  REM  TESTE DE "CTRL-J"
20  PRINT "ABCDEFG";
30  FOR I = 1 TO 6
35  PRINT I; CHR$ (10);
40  NEXT I
45  SPEED= 255
50  PRINT "HIJKLM"
70  END
255  SPEED= 0
```

P11.3

O que faz *CTRL-J* ou **CHR$ (10)** ?

**R-15**

As letras LF na coluna **CHAR** de *CTRL-J* são as iniciais de *Line Feed*, isto é, *CTRL-J* move o cursor uma linha para baixo e para o começo da linha.

No código 13, CR significa *Carriage Return* ou simplesmente *RETURN*; podemos usar *CTRL-M* no lugar de *RETURN*.

Por exemplo, digite o comando abaixo e, em vez de finalizar com *RETURN*, pressione *CTRL-M*.

```
FOR I = 1 TO 100: PRINT I,: NEXT I
```

Qual a resposta?

**R-16**

Faça:

```
PRINT CHR$(13)
```



**R-17**

**PRINT CHR$ (13)** produz a mesma saída que **PRINT : PRINT**, uma vez que **CHR$ (13)** pula uma linha na tela.

Digite agora, após limpar a tela:

```
PRINT "ABCDE"
```

A seguir pressione *CTRL-U*.

**R-18**

*CTRL-U* corresponde à tecla [→], isto é, a tecla que move o cursor uma casa para a direita.

Rode, a seguir, o programa P11.4.

```
10  REM  TESTE DE "CTRL-X"
20  HOME
30  INPUT "QUAL E' O SEU NOME? ";N$
40  PRINT : PRINT : PRINT "O SEU NOME E' ";N$
50  END
```

P11.4

Quando o computador perguntar seu nome digite:

**ZEZINH** *CTRL-X*

O que ocorreu?

**R-19**

Responda agora:

**JOSE ALBERTO** *RETURN*

O que ocorreu?

**R-2Ø**

*CTRL-X* cancela a entrada que você forneceu antes de pressionar *RETURN*; essa entrada pode ser uma linha de programa ou um **INPUT**.

## IV. MANIPULAÇÃO DE STRINGS

Antes de qualquer explicação a priori, digite o seguinte comando direto:

```
PRINT LEN("PELE'")
```

**R-21**

```
PRINT LEN("SANTOS FC")
```

**R-22**

```
T$ = "PONTE PRETA"
PRINT LEN(T$)
```



**R-23**

É de grande utilidade saber o tamanho (comprimento) de uma cadeia *string*. Usamos **LEN** desde para centralizar um título até para armazenarmos um texto completo em uma matriz. Faça o seguinte programa:

```
1Ø  REM  CENTRALIZADOR DE TITULOS
2Ø  HOME
3Ø  INPUT "DIGITE O TITULO: ";T$
4Ø P = (4Ø -  LEN (T$)) / 2 + 1
5Ø  HOME : INVERSE : HTAB P
6Ø  PRINT T$
7Ø  NORMAL : END
```

P11.5

Rode P11.5 e explique a instrução 4Ø. Lembre-se que a largura do vídeo é de 4Ø colunas.

**R-24**

Para podermos localizar, reduzir, separar e compor *strings*, as funções **LEFT$**, **RIGHT$** e **MID$** são extremamente úteis e poderosas.

Digite os seguintes comandos diretos:

```
A$ = "MATO GROSSO"
PRINT A$
```

**R-25**

```
PRINT LEFT$(A$,8)
```

**R-26**

```
PRINT LEFT$(A$,1)
```

**R-27**

```
PRINT LEFT$(A$,20)
```

**R-28**

```
PRINT RIGHT$(A$,1)
```

**R-29**

```
PRINT RIGHT$(A$,5)
```

**R-30**

```
PRINT RIGHT$(A$,15)
```



**R-31**

O que fazem as funções **LEFT$** e **RIGHT$** ?

**R-32**

As funções acima têm a forma **LEFT$** (*string, n*) e **RIGHT$** (*string, n*). Definem uma cadeia de caracteres com *n* caracteres de comprimento (onde *n* pode ser um número, uma variável numérica ou uma expressão algébrica) igual aos *n* primeiros caracteres no caso de **LEFT$** ou dos *n* últimos caracteres no caso de **RIGHT$**.

Digite o programa P11.6 e escreva a saída que você prevê:

```
10  REM  LEFT$
20  HOME
30 A$ = "MATO GROSSO"
40  FOR I = 1 TO 11
50  PRINT  LEFT$ (A$,I)
60  NEXT I
70  FOR J = 10 TO 1 STEP  - 1
80  PRINT  LEFT$ (A$,J)
90  NEXT J
100  END
```

P11.6

Saída prevista:

**R-33**

**RUN**ize o programa e justifique a sua saída.

**R-34**

No programa P11.6 tivemos que contar o número de caracteres em **A$** para determinarmos o valor final do contador. Entretanto, já vimos uma função que fornece diretamente o comprimento de um *string* qualquer. Essa função é definida pelo comando **LEN**.

Modifique o programa P11.6 para:

```
1Ø  REM  RIGHT$
2Ø  HOME
3Ø A$ = "MATO GROSSO"
4Ø  FOR I =  LEN (A$) TO 1 STEP  - 1
5Ø  PRINT  RIGHT$ (A$,I)
6Ø  NEXT I
7Ø  FOR J = 2 TO  LEN (A$)
8Ø  PRINT  RIGHT$ (A$,J)
9Ø  NEXT J
1ØØ  FOR Y = 1 TO 1ØØØ: NEXT Y
11Ø  FOR X = 1 TO 2Ø: PRINT  CHR$ (1Ø): NEXT X
12Ø  END
```

P11.7

Qual a nova saída?

**R-35**

Qual a função da linha 11Ø?

**R-36**

Explique as saídas fornecidas pelas instruções 5Ø e 8Ø.



**R-37**

Por que o **FOR** da instrução 7Ø inicia com 2?

**R-38**

Quais as modificações a serem feitas para escalonarmos um *string* qualquer?

**R-39**

A outra função importante para a manipulação de *strings* é a **MID$**. Digite:

```
A$ = "MATO GROSSO"
PRINT MID$(A$,3,5)
```

**R-4Ø**

```
B$ = MID$("CORINTHIANS",4,3)
```

**R-41**

```
PRINT B$
```

**R-42**

```
PRINT MID$(B$,2,1)
```

**R-43**

**MID$ (*sexpr, n1, n2*)** define uma nova cadeia de caracteres de comprimento *n2* a partir do caractere *n1*. A expressão *sexpr* significa expressão *string* (variável ou *string literal*). Assim, é válido o comando:

```
A$ = LEFT$(MID$("ABCDE",2,2),1)
```

Qual o *string* armazenado em A$?

**R-44**

Antes da sua resposta, faça, para sua comodidade:

```
B$ = MID$("ABCDE",2,2)
```

e em seguida

```
A$ = LEFT$(B$,1)
```

Sua resposta é:

**R-45**



Faça, a seguir,

```
X$ = "PERNAMBUCO"
PRINT MID$(X$,4,5)
```

**R-46**

```
PRINT MID$(X$,4)
```

**R-47**

```
PRINT MID$(X$,8)
```

**R-48**

Omitindo o argumento *n2*, o *string* resultante tem início no caractere *n1* do *string* fornecido, terminando no último caractere deste. Para se obter somente o enésimo (N) caractere de um *string*, fazemos

**MID$ (A$, N, 1)**

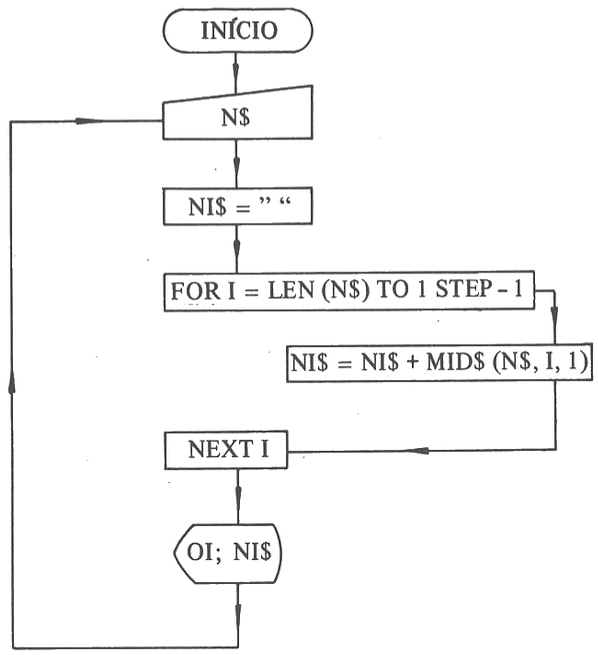
onde N é o caractere em questão.

Digite o programa P11.8 e antes de rodar o programa responda qual a sua saída.

```
10  HOME
20  PRINT : PRINT : PRINT : INPUT "DIGITE O SEU NOME: "
;N$
30 NI$ = ""
40  FOR I =  LEN (N$) TO 1 STEP  - 1
50 NI$ = NI$ +  MID$ (N$,I,1)
60  NEXT I
70  PRINT : PRINT : PRINT "OI, ";NI$;"!"
80  PRINT : PRINT : PRINT : GOTO 20
```

P11.8

Fluxograma




A saída do programa P11.8 é:

**R-49**

Qual a finalidade da linha 3Ø?

**R-5Ø**



O que faz o *loop* das linhas 4Ø - 6Ø - 4Ø?

**R-51**

Carregue o computador com o programa P11.9.

```
10  HOME
20  PRINT "ENTRE UM "; CHR$ (34);"STRING"; CHR$ (34);":
 ";
30  INPUT "";A$
40 N =  LEN (A$)
50  DIM A$(N)
60  FOR I = 1 TO N
70 A$(I) =  MID$ (A$,I,1)
80  NEXT I
90  PRINT : PRINT : PRINT A$
100  PRINT : PRINT : PRINT
110  FOR J = 1 TO N
120  PRINT A$(J);" ";
130  NEXT J
140  PRINT : PRINT : PRINT
150  END
```

P11.9

Por que usamos duas vezes **CHR$ (34)** na instrução 2Ø?

**R-52**

Você se lembra por que colocamos duas aspas (" ") juntas na instrução 3Ø?

**R-53**

O que faz o *loop* das linhas 6Ø - 8Ø - 6Ø?

**R-54**

A decomposição de *strings* num *ARRAY* de caracteres é uma técnica de programação muito útil, pois nos permite trabalhar melhor com cadeias de caracteres desde que tenhamos bastante memória disponível.

Como aplicação analise o programa P11.1Ø, que verifica se uma palavra é um palíndromo ou não.

Palíndromos são palavras que se lêem igualmente da esquerda para a direita e da direita para a esquerda.

```
5  REM  *** PALINDROMOS ***
1Ø  HOME
2Ø  INPUT "DIGITE UMA PALAVRA: ";P$
25  PRINT : PRINT : PRINT : PRINT
3Ø N =  LEN (P$)
4Ø  DIM P$(N),I$(N)
5Ø  FOR I = 1 TO N
6Ø P$(I) =  MID$ (P$,I,1)
7Ø  NEXT I
8Ø  FOR J = N TO 1 STEP  - 1
9Ø I$ = I$ + P$(J)
1ØØ  NEXT J
11Ø  PRINT "NORMAL: ";P$: PRINT
12Ø  PRINT : PRINT "INVERTIDO: ";I$: PRINT
13Ø  IF P$ = I$ THEN  PRINT "ESSA PALAVRA E' UM PALINDR
OMO": GOTO 15Ø
14Ø  PRINT P$;" NAO E' UM PALINDROMO"
15Ø  PRINT : PRINT : INPUT "CONTINUA (S/N)? ";R$: IF R$
 = "S" THEN  CLEAR : GOTO 1Ø
16Ø  END
```

P11.1Ø

Qual a finalidade da instrução 4Ø?

**R-55**



O que fazem e como trabalham os *loops* das instruções 5Ø, 7Ø, 5Ø e 8Ø, 1ØØ, 8Ø?

**R-56**

Qual a finalidade do comando **CLEAR** na instrução 15Ø?

**R-57**

Estruture um programa que centraliza um título e o escreva, no modo **FLASH**, de trás para frente (invertendo-se as posições das letras).

**R-58**

Analise o programa P11.11:

```
5  REM  LOCALIZACAO DE LETRAS NO ALFABETO
1Ø AL$ = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"
2Ø  HOME
3Ø  PRINT "DIGITE UMA LETRA DO ALFABETO: ";
4Ø  GET X$
5Ø  IF X$ < "A" THEN 14Ø
6Ø  IF X$ > "Z" THEN 14Ø
7Ø N = 1
8Ø  IF  MID$ (AL$,N,1) = X$ THEN 11Ø
9Ø N = N + 1
1ØØ  GOTO 8Ø
11Ø  HTAB 3: VTAB 1Ø
12Ø  PRINT X$;" E' A LETRA DE NUMERO ";N;" NO ALFABETO"
13Ø  FOR I = 1 TO 2ØØØ: NEXT I: GOTO 2Ø
14Ø  VTAB 2Ø: HTAB 1Ø: FLASH
15Ø  PRINT X$; CHR$ (7); CHR$ (7);: NORMAL
16Ø  PRINT " NAO E' LETRA DO ALFABETO"
17Ø  FOR I = 1 TO 2ØØØ: NEXT I: GOTO 2Ø
```

P11.11

Qual a função das instruções 5Ø e 6Ø?

**R-59**

O que faz a linha 8Ø?

**R-6Ø**

Como podemos sair (interromper) do **GET X**$ definido pela instrução 4Ø?

**R-61**

## V. NÚMEROS E CARACTERES

Vamos relembrar quais as diferenças entre variáveis numéricas e variáveis *string*. A resposta ao comando

```
PRINT 182
```

é cento e oitenta e dois, e a resposta ao comando

```
PRINT "182"
```

é um, oito, dois.

A saída do primeiro **PRINT** é um número e do segundo é uma cadeia de caracteres.

Quais as saídas de:

```
PRINT 182 + 182
```



**R-62**

```
PRINT "182" + "182"
```

**R-63**

## VI. MISTURANDO STRINGS E NÚMEROS

Digite, anotando suas respectivas saídas, os comandos abaixo:

```
A = 182
A$ = "182"
PRINT VAL(A$) + VAL(A$)
```

**R-64**

```
PRINT STR$(A) + STR$(A)
```

**R-65**

O que faz o comando **VAL**?

**R-66**

O que faz o comando **STR$** ?

**R-67**

Digite:

```
PRINT STR$(2/3)
```

**R-68**

```
PRINT STR$(1234567890123)
```

**R-69**

O que aconteceu em cada caso?

**R-7Ø**

O comando **STR$** (*expr algeb*) converte o valor da expressão algébrica ou numérica em um *string*.

O comando **VAL** (*sexpr*) faz o contrário de **STR$**. Transforma em valor numérico a *sexpr* (expressão *string*), parando quando encontra um caractere que não faz parte do número.

Tente prever as saídas dos comandos abaixo:

```
PRINT VAL("123")
```

**R-71**



```
PRINT VAL("123A")
```

**R-72**

```
PRINT VAL("A123B")
```

**R-73**

```
PRINT VAL("2.45E+23")
```

**R-74**

```
PRINT VAL("-2.45E+2A3")
```

**R-75**

Se o primeiro caractere da *sexpr* não for um caractere numérico ou um sinal (+ ou -), então **VAL** fornece valor zero, caso contrário cada caractere será examinado da esquerda para a direita até encontrar um caractere não válido. Os caracteres válidos são:

dígitos de Ø a 9
espaço
ponto decimal
letra E em caso de notação científica

Faça:

```
PRINT VAL("-2.45E-Ø3AB")
```

**R-76**

```
PRINT VAL("-2.45E-77AB")
```

**R-77**

Discuta as respostas anteriores.

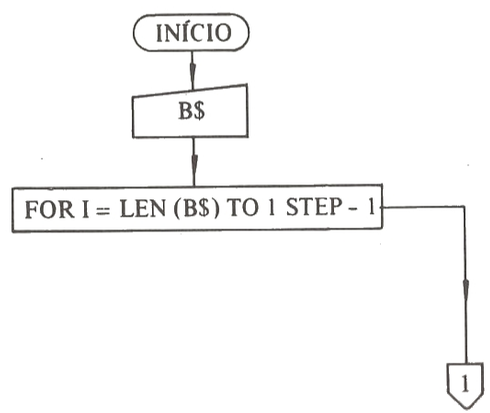
**R-78**

Utilizando a função **VAL** vamos escrever um programa que converte números binários em números decimais.

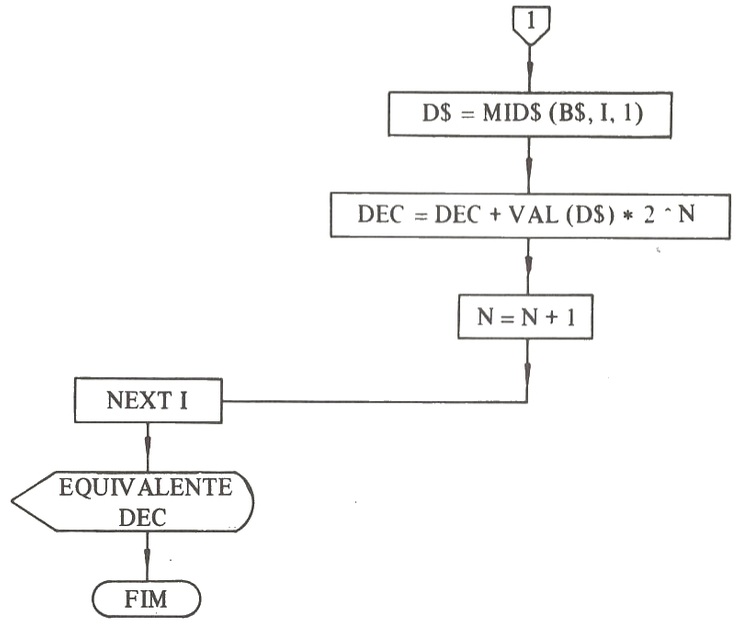
Lembre-se que na base 2 temos somente 1 e Ø como dígitos representativos. Assim:

$$11Ø1_2 = 1 . 2^3 + 1 . 2^2 + Ø . 2^1 + 1 . 2^0 = 8+4+Ø+1 = 13_{10}$$

Fluxograma






Codificação

```
1Ø N = Ø:DEC = Ø
2Ø  INPUT "QUAL O NUMERO BINARIO? ";B$
3Ø  FOR I =  LEN (B$) TO 1 STEP  - 1
4Ø D$ =  MID$ (B$,I,1)
5Ø DEC = DEC +  VAL (D$) * 2 ^ N
6Ø N = N + 1
7Ø  NEXT I
8Ø  PRINT "O EQUIVALENTE DECIMAL E' ";DEC
9Ø  END
```

P11.12

Por que foi necessário introduzir uma variável contadora extra (variável N) ?

**R-79**

O que faz a linha 5Ø?

**R-8Ø**

A função **STR$** pode ser usada para visualizar um número fixo de casas depois da vírgula. Analise o programa P11.13:

```
1Ø  REM  PROGRAMA PARA FIXAR O NUMERO DE CASAS DECIMAIS
2Ø  HOME
3Ø  PRINT : PRINT
4Ø  INPUT "QUANTAS CASAS DECIMAIS? ";FIX
5Ø  PRINT : PRINT
6Ø  INPUT "ENTRE UM NUMERO QUALQUER: ";C
7Ø C$ =  STR$ (C)
8Ø  IF C =  INT (C) THEN C$ = C$ + ".Ø"
9Ø N = 1
1ØØ  IF  MID$ (C$,N,1) = "." THEN 13Ø
11Ø N = N + 1
12Ø  GOTO 1ØØ
13Ø IN$ =  LEFT$ (C$,N):FRC$ =  RIGHT$ (C$, LEN (C$) -
N)
14Ø  IF FIX = Ø THEN FRC$ = "": GOTO 19Ø
15Ø  IF  LEN (FRC$) > FIX THEN FRC$ =  LEFT$ (FRC$,FIX)
16Ø  IF  LEN (FRC$) = FIX GOTO 19Ø
17Ø FRC$ = FRC$ + "Ø"
18Ø  GOTO 16Ø
19Ø  PRINT : PRINT : PRINT "O NUMERO EM FORMA FIXA E' "
;IN$ + FRC$
2ØØ  END
```

P11.13

Faça o fluxograma e explique como funciona o programa P11.13.

**R-81**



Qual a função da linha 7Ø?

**R-82**

Qual a função da linha 8Ø?

**R-83**

Como o número é decomposto em parte inteira e fracionária?

**R-84**

Explique **GOTO 19Ø** na instrução 14Ø.

**R-85**

O que faz o *loop* 16Ø - 17Ø - 18Ø?

**R-86**

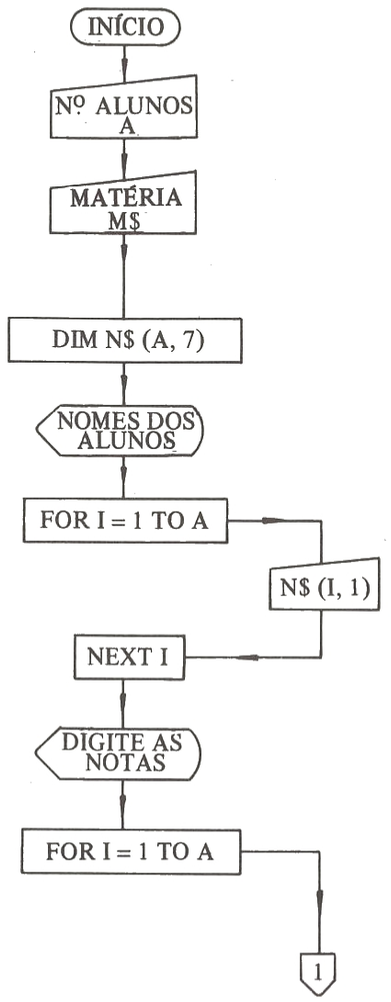
Agora que estamos familiarizados com as funções **VAL** e **STR$**, vamos estruturar um exercício complexo e aproveitar para fazermos uma revisão de matrizes.

Em muitos casos, pode-se definir um arquivo de alunos com nomes e notas, por exemplo, por meio de uma matriz alfanumérica. É claro que não poderemos operacionalizar diretamente a nota "6.5", pois ela deve ser convertida ao seu valor numérico através do comando **VAL**.

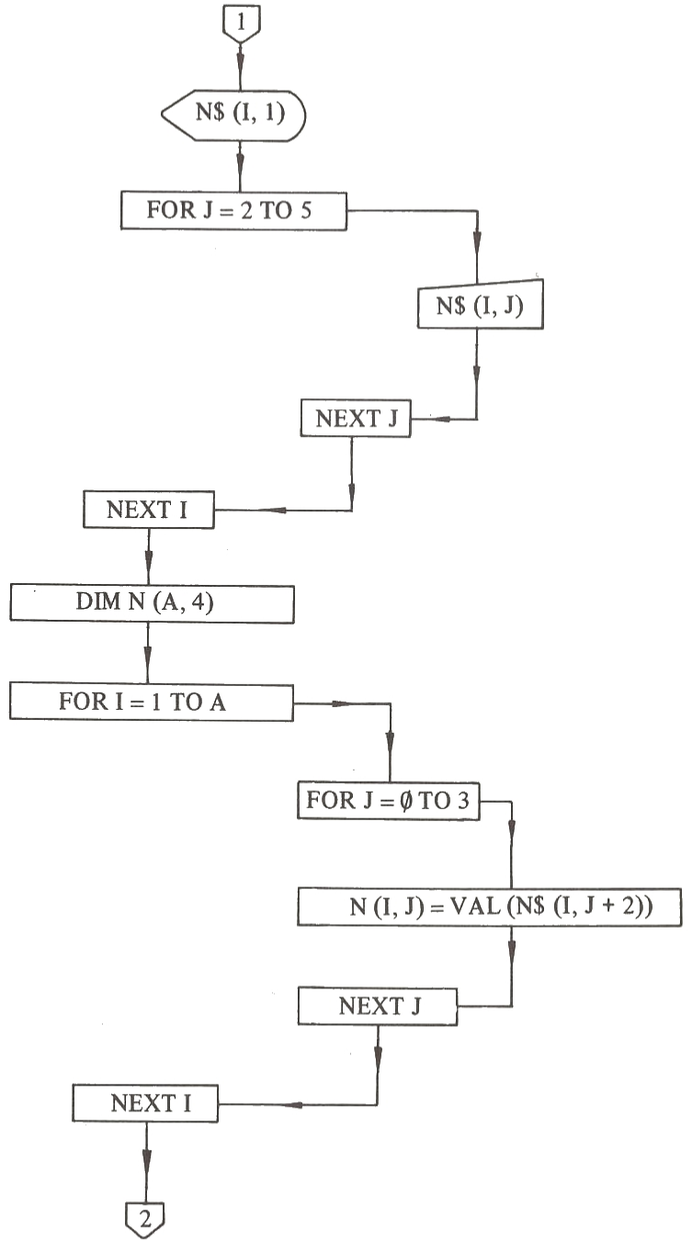
Vamos construir um programa que define uma matriz bidimensional para guardar os nomes dos alunos, suas notas do 1º ao 4º bimestre e a média final calculada pelo computador, onde a média final é dada por:

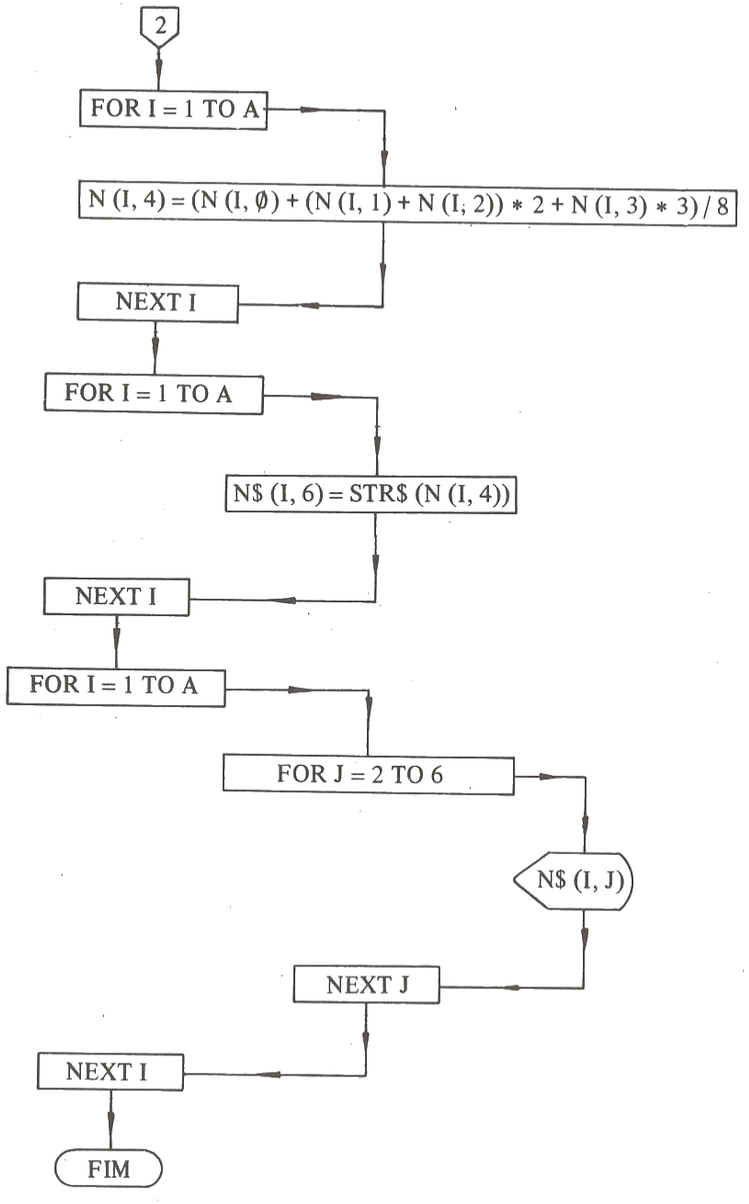
$$Mf = \frac{B_1 + 2 \times B_2 + 2 \times B_3 + 3 \times B_4}{8}$$

Fluxograma










Codificação

```
10  REM  ARQUIVO DE NOTAS
20  HOME
30  INPUT "QUANTOS ALUNOS? ";A
40  PRINT : PRINT
50  INPUT "MATERIA: ";M$
60  DIM N$(A,7)
70  PRINT "DIGITE OS NOMES DOS ALUNOS": PRINT
80  FOR I = 1 TO A
90  PRINT "ALUNO ";I;" ";
100  INPUT "";N$(I,1)
110  NEXT I
120  HOME
130  PRINT "DIGITE AS NOTAS": PRINT
140  FOR I = 1 TO A
150  PRINT N$(I,1)
160  FOR J = 2 TO 5
170  INPUT "";N$(I,J)
180  NEXT J
190  PRINT
200  NEXT I
210  REM TRANSFORMANDO
220  DIM N(A,4)
230  FOR I = 1 TO A
240  FOR J = Ø TO 3
250 N(I,J) =  VAL (N$(I,J + 2))
260  NEXT J,I
270  REM CALCULO DA MEDIA FINAL
280  FOR I = 1 TO A
290 N(I,4) = (N(I,Ø) + (N(I,1) + N(I,2)) * 2 + N(I,3) *
 3) / 8
300  NEXT I
310  REM TRANSFORMANDO
320  FOR I = 1 TO A
330 N$(I,6) =  STR$ (N(I,4))
340  NEXT
350  REM APRESENTACAO FINAL
360  HOME
370  HTAB 10: INVERSE : PRINT M$: NORMAL
380  PRINT : PRINT
390  FOR I = 1 TO A
400  PRINT N$(I,1);"  ";
410  FOR J = 2 TO 6
420  PRINT  SPC( 2);N$(I,J);
430  NEXT J
440  PRINT : PRINT
450  IF I / 10 =  INT (I / 10) THEN  GOSUB 1000
460  NEXT I
470  END
1000  REM SUBROTINA PARA PARAR A VISUALIZACAO
1010  VTAB 24: HTAB 1: INVERSE : PRINT "APERTEUMA TECLA
 PARA CONTINUAR";: GET A$
1020  HOME : NORMAL : RETURN
```

Explique os **DIM** das linhas 6Ø e 22Ø.

**R-87**

Como funciona o *loop* das linhas 14Ø - 2ØØ - 14Ø e seu *loop* interno 16Ø - 18Ø - 16Ø?

**R-88**

Na linha 25Ø explique a presença de **VAL** e dos índices J e J + 2.

**R-89**

Qual a finalidade do *loop* 23Ø - 26Ø - 23Ø?

**R-9Ø**

O que ocorre na linha 29Ø?

**R-91**

O que faz o *loop* 32Ø - 34Ø - 32Ø?

**R-92**



Explique a finalidade das linhas 4ØØ, 42Ø e 45Ø, bem como sua respectiva sub-rotina.

**R-93**

Analise o programa P11.15 abaixo:

```
1Ø  REM  CODIFICADOR DE MENSAGENS
2Ø  INPUT "SEU TEXTO: ";A$
3Ø  INPUT "FATOR DE CODIGO: ";A
4Ø  FOR I = 1 TO  LEN (A$)
6Ø B$(I) =  MID$ (A$,I,1)
7Ø  NEXT I
8Ø  FOR I = 1 TO  LEN (A$)
9Ø B$(I) =  CHR$ ( ASC (B$(I)) + A)
1ØØ  NEXT I
11Ø  FOR I = 1 TO  LEN (A$)
12Ø C$ = C$ + B$(I)
13Ø  NEXT I
14Ø  PRINT "O TEXTO CODIFICADO E': ";C$
15Ø  END
```

P11.15

Rode o programa e faça **A$ = "BASIC"** e **A = 5**. Antes, entretanto, tente prever a sua saída.

**R-94**

Por que é necessário fazer **DIM B$ (LEN (A$))**, na instrução 4Ø?

**R-95**

O que faz a instrução 6Ø?

**R-96**

O que faz a linha 9Ø?

**R-97**

O que faz o *loop* das linhas 11Ø - 13Ø - 11Ø?

**R-98**

## VII. COLETOR DE LINHAS

Você já deve ter notado os inconvenientes de se fazer um *input* de um texto através do comando **INPUT A$**, principalmente quando você está fazendo um programa para outro usuário. Dentre esses inconvenientes podemos citar:

1. não aceitação de vírgulas e texto a seguir;
2. não aceitação de aspas iniciais ou finais;
3. falta de controle sobre o número de caracteres entrados;
4. aceitação de todos os caracteres, muitos dos quais indesejáveis.

Por isso é muito interessante fazer uma rotina que "pegue" uma linha de texto por meio do comando **GET**. Como o **GET** só lê um caractere de cada vez, essa rotina deve incluir um *loop* controlado, de preferência não por **FOR ... NEXT**.



Digite o programa P11.16:

```
20000  REM  COLETOR DE LINHAS
20010  REM
20020  REM FAZ O 'INPUT' DE UM STRING
20030  REM DE 'LN' CARACTERES
20040  REM E A RETORNA EM L$
20050  REM VARIAVEIS USADAS:  A,L,LN,A$,L$
20060  REM
20070 L$ = "":L = 0
20080  GET A$:A =  ASC (A$)
20090  IF A = 13 THEN  RETURN
20180  IF A < 32 OR L >  = LNG THEN 20230
20190 L = L + 1
20200 L$ = L$ + A$
20210  PRINT A$;
20220  GOTO 20080
20230  PRINT  CHR$ (7);
20240  GOTO 20080
```

P11.16

Normalmente, P11.16 vem como uma sub-rotina de algum programa.

Que variável deve ser fornecida pelo programa principal a esta rotina? O que ela representa?

**R-99**

O que acontece quando digitamos um caractere válido como A ou B?

**R-1ØØ**

O que faz a instrução 2ØØ8Ø?

**R-1Ø1**

Quando L$ já chegou ao seu tamanho máximo e tentamos digitar uma outra informação, o que acontece?

**R-1Ø2**

E quando pressionamos *RETURN*?

**R-1Ø3**

Se digitarmos um caractere erradamente, como podemos corrigi-lo?

**R-1Ø4**

Que acontece quando digitamos um caractere de controle, cujos códigos ASCII são menores que 32?

**R-1Ø5**

Para podermos definir, nesse *line collector*, o *back space*, temos que definir uma outra sub-rotina. A sub-rotina a seguir pode ser adicionada à sub-rotina COLETOR DE LINHAS, para implementar a função:

BACKSPACE "←":

```
20100  IF A = 8 AND L > Ø THEN  GOSUB 20250: GOTO 20080
20260 L = L - 1
20270  IF L = Ø THEN L$ = "": GOTO 20290
20280 L$ =  LEFT$ (L$,L)
20290  PRINT  CHR$ (8);" "; CHR$ (8);
20300  RETURN
```

P11.16a



Explique o funcionamento da sub-rotina *back space.*

**R-1Ø6**

Faça o fluxograma completo do *LINE COLLECTOR.*

**R-1Ø7**

A título de ilustração, analise as sub-rotinas de *line collector* abaixo:

```
100 LN = 50
110  PRINT : GOSUB 1000
120  PRINT : PRINT : PRINT L$
130  END
1000  REM  COLETOR DE LINHAS
1010 L$ = "":L = Ø
1020  GET A$:A =  ASC (A$)
1030  IF A = 13 THEN  RETURN
1040  IF A = 8 AND L > Ø THEN  GOSUB 1160: GOTO 1020
1050  REM  DEFINIR TECLAS CTRL-Q, CTRL-W, CTRL-E
1060  IF A = 17 THEN A$ =  CHR$ (91): GOTO 1100
1070  IF A = 23 THEN A$ =  CHR$ (95): GOTO 1100
1080  IF A = 5 THEN A$ =  CHR$ (92): GOTO 1100
1090  IF A < 32 OR L > LN THEN 1140
1100 L = L + 1
1110 L$ = L$ + A$
1120  PRINT A$;
1130  GOTO 1020
1140  PRINT  CHR$ (7);
1150  GOTO 1020
1160  REM  BACKSPACE
1170 L = L - 1
1180  IF L = Ø THEN L$ = "": GOTO 1200
1190 L$ =  LEFT$ (L$,L)
1200  PRINT  CHR$ (8);" "; CHR$ (8);
1210  RETURN
```

P11.17

```
100  HOME
110  PRINT "DIGITE SEU TEXTO:": PRINT
120 LN = 50
130  GOSUB 1000
140  PRINT : PRINT : PRINT L$: END
999  REM  LINE COLLECTOR
1000 L$ = "":L = 0
1010  GET A$:A =  ASC (A$)
1020  IF A = 13 THEN  RETURN
1030  IF A = 8 AND L > 0 THEN  GOSUB 1110: GOTO 1010
1040  IF A < 32 OR L >  = LNG THEN 1090
1050 L = L + 1
1060 L$ = L$ + A$
1070  PRINT A$;
1080  GOTO 1010
1090  PRINT  CHR$ (7);
1100  GOTO 1010
1110  REM  BACKSPACE
1120 L = L - 1
1130  IF L = 0 THEN L$ = "": GOTO 1150
1140 L$ =  LEFT$ (L$,L)
1150  PRINT  CHR$ (8);" "; CHR$ (8);
1160  RETURN
```

P11.18

## VIII. EXERCÍCIOS PROPOSTOS

11.1 Faça um programa para aceitar uma palavra (via **INPUT**) e contar quantas vogais foram digitadas.

11.2 Retome o exercício anterior e modifique-o para também indicar as posições das vogais.

11.3 Baseado no programa anterior, estruture um jogo de forca que forneça o total de erros e acertos (10 tentativas no máximo). As palavras para o jogador adivinhar devem estar em **DATA**, e, no lugar das letras escondidas, o computador deverá mostrar o caractere *under line*, **CHR$ (95).**



11.4 Faça um programa que peça uma palavra ou sentença com um número ímpar de caracteres (até 39) e produza uma saída em forma de losango, formado pelos caracteres do *string*.

Exemplo:

```
"STRING":   APPLESOFT


                E
               LES
              PLESO
             PPLESOF
            APPLESOFT
             PPLESOF
              PLESO
               LES
                E
```

Obs.: A palavra **STRING** deve aparecer entre aspas (caractere 34) no vídeo.

11.5 Faça um programa que peça uma lista de palavras cujo número não esteja determinado, e as alinhe pela direita. Exemplo:

```
                POR
             DENTRO
                 DO
              APPLE
```

Obs.: Para que o computador saiba que a lista de entrada terminou, devemos pressionar tão-somente *RETURN*.

11.6 Os símbolos do sistema hexadecimal são:

0 1 2 3 4 5 6 7 8 9 A B C D E F

Faça um programa que converta números na base dez para números na base hexadecimal. Sugestão: use também os códigos ASCII das letras A até F.

11.7 Reveja o programa P11.14. Modifique-o para que possam ser arquivadas as notas de várias matérias do aluno. Sugestão: use uma matriz tridimensional.

11.8 Reescreva a rotina COLETOR DE LINHAS, implementando os comandos *CRTL-X* e *CRTL-C*.

11.9 Reveja o programa CODIFICADOR DE MENSAGEM. Estruture um programa para decodificar um texto já codificado. Deve pedir o código, o texto codificado e apresentar o texto original.

11.10 Estruture o fluxograma e faça a codificação de um programa que tabule o ponto decimal. Exemplo:

```
  3.0148
 27.31
141.0
  0.00005
```

11.11 Usando os comandos e funções até agora vistos, estruture um programa para simular uma máquina de escrever.

11.12 Faça um fluxograma e a respectiva codificação do programa que, dado um título qualquer, faça a separação letra por letra (um espaço em branco entre elas) e centralize o título espaçado.

11.13 Construa um programa que, após solicitar um *string* qualquer, solicite um caractere e o localize no *string* determinado a sua posição, com ou sem repetições, avisando quantas vezes o caractere for repetido.

11.14 Faça um programa para sublinhar um título, pulando espaços, hifens, vírgulas e ponto-e-vírgulas.

11.15 Faça o fluxograma do programa P11.3.

## IX. RAXAKUKA

Diga quais as conclusões falsas e quais as verdadeiras:

11.1 Se as estrelas brilharem hoje à noite, amanhã o tempo estará quente. Com efeito, hoje à noite as estrelas estão brilhando, logo:

a) amanhã o tempo não estará quente.
b) as estrelas brilharão amanhã à noite.
c) o tempo estará quente amanhã.

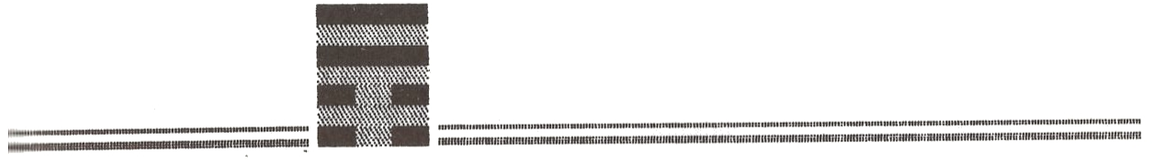


11.2 Todos "zublet" possuem 3 olhos. Este "keptik" possui 3 olhos. Logo:

a) este "keptik" é o mesmo que um "zublet".
b) este "keptik" apresenta "zubletiania" ocular.

11.3 Nenhum homem é bom, mas alguns não são maus. Logo:

a) todos os homens não são maus.
b) nenhum homem é mau.
c) todos os homens não são bons.



# Funções definindo suas próprias funções

*"É insuficiente convencer a inteligência, é necessário persuadir o coração.*
*Portanto, o espírito geométrico é impotente e se deve recorrer ao espírito de finura."*

*Blaise Pascal*
*(1623-1662)*

## I. INTRODUÇÃO

Sem dúvida, as funções matemáticas são de grande valia e extrema utilidade para cálculos mais elaborados e dirigidos.

O computador, por meio de seu Applesoft, apresenta toda uma gama de funções que fornecem um determinado valor para cada valor estipulado.

Todas essas funções têm o formato:

*FUNÇÃO (argumento)*

O argumento de uma função matemática pode ser um número, uma variável numérica ou uma expressão algébrica, incluindo número e/ou variáveis.

## II. FUNÇÃO SQR

A função **SQR** devolve a raiz quadrada positiva do argumento.
Digite:

```
B = 4
A = SQR(B)
PRINT A
```

**R-1**

A função **SQR** é mais rápida do que ^ . 5. Como você pode prová-lo?
Digite:

```
PRINT SQR(-4)
```

**R-2**



Qual o significado da mensagem anterior?

**R-3**

## III. FUNÇÃO SGN

Digite os comandos abaixo e analise suas respostas:

```
PRINT SGN(10)
```

**R-4**

```
PRINT SGN(-100)
```

**R-5**

```
PRINT SGN(0)
```

**R-6**

O que faz a função **SGN**?

**R-7**

A função **SGN (*arg*)** imprime -1 se *arg* $<$ Ø, Ø se *arg* = Ø e +1 se *arg* $>$ Ø.

## IV. FUNÇÃO ABS

A função **ABS** fornece o valor absoluto (módulo) do argumento.
Digite:

```
A = 3*4 : B = 2 - 1Ø
PRINT ABS(A), ABS(B)
```

**R-8**

## V. FUNÇÃO INT

A função **INT** fornece o maior inteiro, menor ou igual ao argumento.
Tendo em vista essa definição, preveja os resultados de:

```
PRINT INT(6.45)
```

**R-9**



```
PRINT INT(-5.5)
```

**R-1Ø**

Deve-se lembrar que -5 é maior que -5,5 e portanto -6 é o maior inteiro menor que -5,5.

É interessante notar que para se determinar o número de casas decimais, sem arredondamento, usamos:

$$\mathbf{X = INT\,(X * 1Ø\ \hat{}\ D)/1Ø\ \hat{}\ D} \qquad 12.1$$

no qual D é o número de casas decimais.

Digite:

```
D = 3 : P = 1Ø^D
X = 4.Ø111
Y = 4.Ø113
Z = 4.Ø118

PRINT X, INT(X*P)/P
```

**R-11**

```
PRINT Y, INT(Y*P)/P
```

**R-12**

```
PRINT Z, INT(Z*P)/P
```

**R-13**

```
A = -X
B = -Y
C = -Z

PRINT A, INT(A*P)/P
```

**R-14**

```
PRINT B, INT(B*P)/P
```

**R-15**

```
PRINT C, INT(C*P)/P
```

**R-16**

A solução apresentada pela relação 12.1 não prevê o arredondamento.
Para se obter o número de casas decimais com arredondamento usamos a expressão:

$$X = INT(X * 10 \hat{} D + .5)/10 \hat{} D \qquad 12.2$$

onde D é o número de casas decimais.

Considerando as variáveis definidas anteriormente, digite:



```
PRINT X, INT(X*P + .5)/P
```

**R-17**

```
PRINT Y, INT(Y*P + .5)/P
```

**R-18**

```
PRINT Z, INT(Z*P + .5)/P
```

**R-19**

```
PRINT A, INT(A*P + .5)/P
```

**R-20**

```
PRINT B, INT(B*P + .5)/P
```

**R-21**

```
PRINT C, INT(C*P + .5)/P
```

**R-22**

Como ocorre o arredondamento por essa solução?
Digite:

```
K = Ø.ØØØØ
PRINT K, INT(K*P + .5)/P
```

**R-23**

Foram impressas as 3 casas decimais na resposta R-23?

**R-24**

Mais adiante, mostraremos uma rotina para limitar o número de dígitos após o ponto decimal.

## VI. FUNÇÃO EXP

A função **EXP** eleva o número *e* (base dos logaritmos naturais) à potência indicada em seu argumento.
Como você deve proceder para descobrir o valor de *e*?

**R-25**



## VII. FUNÇÃO LOG

A função **LOG** retorna o logaritmo natural do argumento.
Faça:

```
PRINT LOG(2)
```

**R-26**

```
PRINT LOG (EXP(1))
```

**R-27**

Qual a razão de tal resultado ?

**R-28**

Para obtermos mudança de base em logaritmos usamos, da matemática, a expressão:

$$\log_a b = \frac{\log_c b}{\log_c a}$$

Determine, pelo computador, o logaritmo de 2 na base 1Ø.

**R-29**

## VIII. FUNÇÕES TRIGONOMÉTRICAS

Para calcularmos os valores das funções seno, cosseno e tangente, o Applesoft apresenta as funções **SIN, COS** e **TAN**. A unidade de ângulo previamente determinada é o radiano. Digite:

```
PI = 3.14159265
PRINT SIN(PI)
```

**R-3Ø**

```
PRINT COS(PI/4)
```

**R-31**

```
PRINT TAN(PI/6)
```

**R-32**

```
PRINT TAN(PI/2)
```

**R-33**

Como podemos transformar radianos em graus e em grados?

**R-34**



Para aqueles que ainda não estudaram medidas de ângulos, ou já esqueceram, lembramos:

1 radiano = 57.2957795 graus
360 graus = 400 grados

Digite o programa P12.1:

```
5  REM FUNCOES TRIGONOMETRICAS
10 PI = 3.14159265
20  HOME
30  PRINT : PRINT
40 A$ = "UNIDADE DE ENTRADA"
50  GOSUB 1000
60  PRINT : PRINT
70  PRINT " 1 - RADIANOS"
80  PRINT " 2 - GRAUS"
90  PRINT " 3 - GRADOS"
100  PRINT : PRINT
110  GOSUB 1000
130  INPUT "";A
140  ON A GOSUB 500,600,700
150  INPUT "ANGULO ";ANG
155 ARC = ANG
160 ANG = ANG * FACT
170  HOME : PRINT : PRINT
180 A$ = "OPERACOES A EFETUAR"
190  GOSUB 1000
200  PRINT : PRINT
210  PRINT " 1 - SENO"
220  PRINT " 2 - COSSENO"
230  PRINT " 3 - TANGENTE"
240  PRINT : PRINT
250 A$ = "ESCOLHA UMA"
260  GOSUB 1000
270  INPUT "";A
280  ON A GOSUB 800,850,900
300  PRINT U$;ARC;") = ";FUNC
400  END
500 FACT = 1: RETURN
600 FACT = PI / 180: RETURN
700 FACT = PI / 200: RETURN
800 FUNC =  SIN (ANG)
810 U$ = "SIN("
```

```
820  RETURN
850 FUNC =  COS (ANG)
860 U$ = "COS("
870  RETURN
900 FUNC =  TAN (ANG)
910 U$ = "TAN("
920  RETURN
1000  INVERSE
1010  PRINT A$
1020  NORMAL
1030  RETURN
```

P12.1

A função **ATN** fornece o arco-tangente do argumento. O ângulo fornecido, em radianos, está compreendido sempre entre $-\pi/2$ e $+\pi/2$.

Como curiosidade rode o programa P12.2:

```
10  HOME
20  FOR Y = 0 TO 62.8 STEP 6.28 / 20
30 X = 10 * (1 +  SIN (Y))
40  HOME
50  VTAB 12
60  PRINT "POR DENTRO DO"; SPC( X);"APPLE"
70  NEXT
```

P12.2

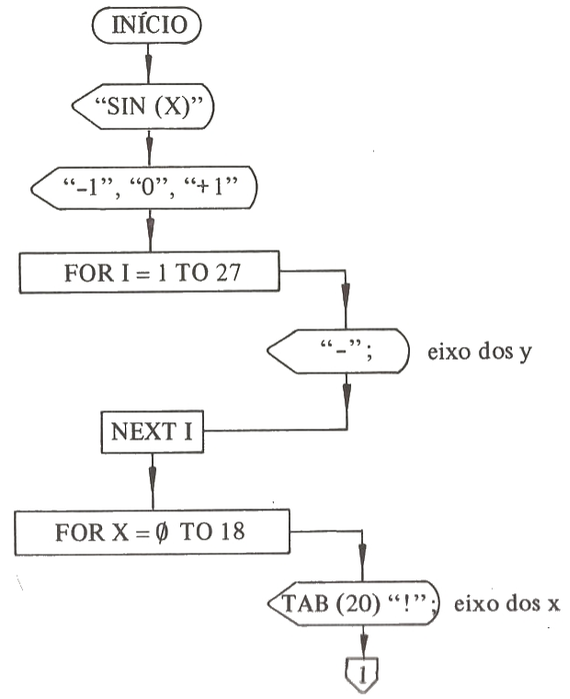
Apague as linhas 40 e 50 do programa P12.2 e rode o novo programa assim constituído.

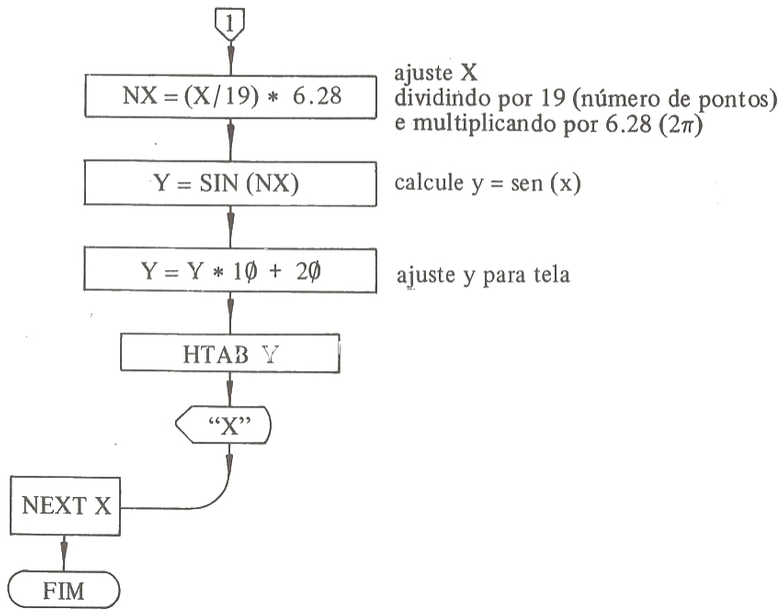
Como exemplo vamos fazer um programa que produza a seguinte saída:



```
SIN(X)
  -1          0          +1
  ------------------------------
                X
                !   X
                !       X
                !           X
                !             X
                !             X
                !             X
                !           X
                !       X
                !   X
               X!X
             X  !
           X    !
         X      !
        X       !
        X       !
        X       !
         X      !
           X    !
             X  !
```

Fluxograma






Codificação

```
1Ø  HOME
2Ø  PRINT  TAB( 4);"SIN(X)"
3Ø  PRINT  TAB( 7);"-1"; TAB( 2Ø);"Ø"; TAB( 32);"+1"
4Ø  PRINT  TAB( 7);
5Ø  FOR I = 1 TO 27
6Ø  PRINT "-";
7Ø  NEXT I
8Ø  PRINT : FOR X = Ø TO 18
9Ø  PRINT  TAB( 2Ø);"!";
1ØØ NX = (X / 19) * 6.28
11Ø Y =  SIN (NX)
12Ø Y = Y * 1Ø + 2Ø
13Ø  HTAB Y: PRINT "X"
14Ø  NEXT X
15Ø  END
```

P12.3

Rode o programa P12.3 e verifique a sua saída.

Qual a função da instrução 4Ø?



**R-35**

O que faz o *loop* 5Ø, 6Ø, 5Ø?

**R-36**

E o *loop* 8Ø, 14Ø, 8Ø?

**R-37**

Como foram ajustados os eixos (instruções 1ØØ e 12Ø)?

**R-38**

## IX. A FUNÇÃO RND

A função **RND** é a responsável pela geração de números aleatórios ou RANDÔMICOS.

Digitando **AL = RND (1)**, o computador, por meio de **RND (1)**, coloca, na variável **AL**, um número decimal com nove algarismos significativos, entre Ø e 1, isto é, $Ø \leqslant AL < 1$.

Na sintaxe da função **RND** é obrigatória a presença do número 1. Se quisermos gerar um número aleatório entre Ø e 5, **NÃO VALE O COMANDO RND (5)**. Para tanto, devemos lançar mão do programa que segue:

```
10  REM  ALEATORIOS INTEIROS ENTRE Ø E 5
20  HOME
30 A =  INT ( RND (1) * 6)
40  PRINT A;" ";
50  GOTO 30
```

P12.4

Rode o programa anterior e verifique sua saída.

**R-39**

Modifique o programa P12.4, para gerar aleatórios com, até, duas casas decimais.

**R-4Ø**

Faça um programa, baseado no programa P11.6, do capítulo anterior, para formular aleatoriamente as multiplicações. Os números gerados devem ser compreendidos entre Ø e 1Ø.

## X. DEFININDO SUAS PRÓPRIAS FUNÇÕES

O comando **DEF FN** nos permite definir nossas próprias funções. Essa definição obedece a algumas restrições:

1. podemos definir apenas funções matemáticas;
2. as definições só podem ter uma linha de programa de comprimento.

O comando **DEF FN** tem a sintaxe:

**DEF FN** *nome* (*variável fantasma*) = *expressão algébrica*

O comando **FN** tem a sintaxe:

**FN** *nome* (*expressão algébrica*)



Como ilustração rode o programa P12.5.

```
10  DEF  FN A(W) = 2 * W + W
20  PRINT  FN A(23)
110  DEF  FN B(X) = 4 + 3
140 G =  FN B(23)
150  PRINT G
210  DEF  FN A(Y) =  FN B(2) + Y
220  PRINT  FN A(G)
230  END
```

P12.5

Quando é executado o comando **FN**, o computador procura a definição da função com o nome dado (que já deve ter sido definida), substitui o valor da expressão algébrica e executa as operações definidas por **DEF FN**.

As regras para a nomenclatura são as mesmas para a nomenclatura das variáveis reais.

Exemplos:

```
10  DEF  FN ABC(I) =  COS (I)
50  DEF  FN ABX(I) =  TAN (I)
```

O computador define, na instrução 1Ø, a função AB como cosseno e depois redefine a mesma função AB como tangente; apenas a última é conservada.

Quando a função não foi definida, o computador envia uma mensagem específica.

Digite e rode o programa P12.6 (não se esqueça do **NEW**!).

```
10  INPUT A
20 B =  FN P(A)
30  PRINT B
```

P12.6

Digite **NEW** e coloque o programa P12.7 no computador.

```
10  HOME
20  DEF  FN QUADRADO(X) = X * X
30  FOR X = 1 TO 10
40  PRINT X, FN QUADRADO(X)
50  NEXT X
60  END
```

P12.7

Rode o programa anterior e em seguida digite:

```
PRINT FN QUADRADO(13)
```

Qual a resposta? Por quê?

**R-41**

```
NEW
PRINT FN QUADRADO(13)
```

Qual a resposta? Por quê?

**R-42**

Tente a seguir:

```
DEF FN F(X) = 180 - X
```

Qual a resposta? Qual seu significado?

**R-43**

Observe que a variável usada na definição de uma função não tem relação qualquer com variáveis usadas no resto do programa, mesmo que sejam variáveis de mesmo nome.

## XI. FUNÇÕES DERIVADAS

Você deve ter notado a falta de certas funções, comuns em calculadoras, tais



como arco-seno, arco-cosseno, secante etc. Essas funções não são disponíveis em BASIC. Mas utilizando **DEF FN** e a tabela de funções derivadas, podemos facilmente implementá-las em um programa.

### XI.1 – Função Secante

Sabemos que sec x = 1/cos x. Logo:

```
DEF FN SEC(X) = 1/COS(X)
```

### XI.2 – Função Cossecante

Por analogia:

```
DEF FN CSC(X) = 1/SIN(X)
```

### XI.3 – Função Cotangente

Seguindo a mesma linha:

```
DEF FN COT(X) = 1/TAN(X)
```

### XI.4 – Função Arco-seno

Lembrando que:

$$\text{sen}^2 x + \cos^2 x = 1$$

$$\text{tg } x = \frac{\text{sen } x}{\cos x}$$

temos:

$$\text{tg } x = \frac{\text{sen } x}{\sqrt{1 - \text{sen}^2 x}}$$

ou ainda:

$$\text{sen } x = \text{tg } x \cdot \sqrt{1 - \text{sen}^2 x}$$

Desse modo definimos arco-seno como:

```
DEF FN ASN(X) = ATN(X/SQR(1-X*X))
```

## XI.5 – Função Arco-cosseno

Sendo $\text{sen } x = \cos(\pi/2 - x)$

temos:

$$\text{arc cos } x = \text{arc sen } (-x) + \frac{\pi}{2}$$

$$\text{arc cos } x = -\text{arc sen } (x) + \frac{\pi}{2}$$

Logo:

```
DEF FN ACS(X) = -ATN(X/SQR(1-X*X)) + 1.571
```

# XII. EXERCÍCIOS PROPOSTOS

12.1 Faça um programa que, dado um número qualquer, devolva:
a) seu valor truncado (sem casas decimais);
b) seu valor truncado com aproximação;
c) sua parte fracionária e com número de casas decimais determinadas sem aproximação;



d) sua parte fracionária e com número de casas decimais determinadas com aproximação.

12.2 Faça um programa que, dado um número qualquer, devolva-o com quatro dígitos depois da vírgula, obedecendo o critério de aproximação e colocação de zeros.

12.3 Estruture um programa que simule um lançamento de dados. Devem ser visualizados os valores independentes.

Exemplo:

**D1 = 5** **D2 = 3**

12.4 Estruture um programa que, dado um inteiro N, lance dois dados N vezes, conte o número de vezes que cada valor (de 2 a 12) foi obtido e, depois, faça uma tabela ou um gráfico de asteriscos a pedido do usuário.

Exemplo de **RUN**:

```
QUAL O VALOR DE N?
--> 10

TABELA

VALOR          FREQUENCIA

  2
  3                1
  4                4
  5                2
  6
  7                1
  8                1
  9                1
 10
 11
 12

GRAFICO (S/N)? ■
```

```
                    GRAFICO

VALOR

  2
  3   *
  4   ****
  5   **
  6
  7   *
  8   *
  9   *
 1Ø
 11
 12
```

12.5 Faça um programa que calcule as mantissas dos logaritmos decimais de 1 a 1ØØ e imprima na tela com seis dígitos. Exemplo:

```
X    LOG(X)          X    LOG(X)

1    ØØØØØØ          51   7Ø757Ø
2    3Ø1Ø3Ø          52   716ØØ3
:                    :
:                    :
```

12.6 Sabendo-se que SENO HIPERBÓLICO e COSSENO HIPERBÓLICO são definidos:

$$\text{SINH } X = \frac{e^X - e^{-X}}{2}$$

$$\text{COSH } X = \frac{e^X + e^{-X}}{2}$$

escreva um programa que faça uma tabela com duas casas decimais e com aproximação dos valores de X, SINH X e COSH X de 1 a 1Ø com incremento de Ø,1.

12.7 Deduza os comandos **DEF FN** para definir as funções ARSINH e ARCOSH.

12.8 Estruture um programa ou uma rotina para fazer a formatação de números de dígitos após a vírgula decimal já com arredondamento.

12.9 Faça um programa, usando **TAB** ou **VTAB** e **HTAB**, para "graficar" as funções SENO, COSSENO e TANGENTE.



12.10 Faça um programa para visualizar uma parábola definida pela função $y = ax^2 + bx + c$. Na tela, deverão aparecer as informações sobre as raízes e discriminante.

12.11 Estruture um programa e o respectivo fluxograma, em que, dados dois números de 5 dígitos, um padrão e o outro formado pela seqüência das respostas de um aluno, compare cada dígito dos dois números e diga quantas foram as respostas certas do aluno.

Exemplo:

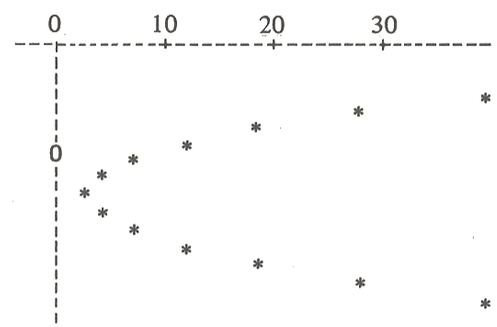
```
gabarito:  5 8 3 7 9
resposta:  5 6 3 4 9
           ---------
           C X C X C   -→  3 acertos
```

OBS.: 1) O número (respostas do aluno) deve ser digitado todo de uma vez.

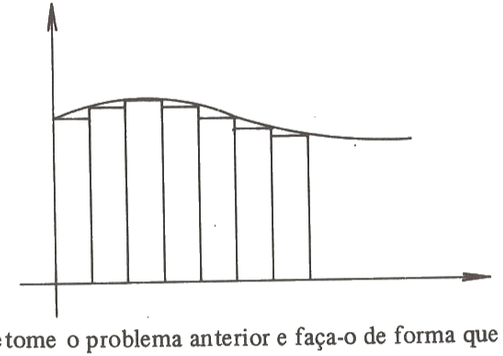
2) Você poderá dividir sucessivamente por dez, usar a função **INT** e comparar dígito a dígito, definidos em duas matrizes lineares; ou

3) Usar as funções **LEFT$**, **RIGHT$** ou **MID$**, definir matrizes lineares e comparar os elementos correspondentes.

12.12 Retome o problema 12.10 e modifique-o para apresentar a saída.




12.13 Uma das técnicas para se determinar a área sob uma curva é dividi-la em pequenos retângulos. Veja a figura seguinte. Defina uma função que permite o cálculo da área sob a curva y = a log (x), onde *a* é uma constante definida pelo usuário. O programa deve solicitar a margem de precisão do cálculo.



12.14 Retome o problema anterior e faça-o de forma que liste, na tela de vídeo, as aproximações efetuadas.

## XIII. RAXAKUKA

12.1 Três pessoas, num bar, fizeram uma despesa que importou em Cr$ 9,00 para cada uma, totalizando Cr$ 27,00. Todavia, cada uma deu ao garçom Cr$ 10,00. Por falta de troco, este devolveu Cr$ 5,00. Destes, tiraram-se Cr$ 3,00, que lhe deram como gorjeta. Então, como sobraram Cr$ 2,00?

12.2 Dez viajantes hospedaram-se num hotel e pediram um quarto para cada um. O hoteleiro, que só dispunha de nove quartos, fez o seguinte:

No quarto nº 1 colocou 2 viajantes; no nº 2 o 3º viajante; no nº 3 o 4º; no nº 4 o 5º; no nº 5 o 6º; no nº 6 o 7º; no nº 7 o 8º; no nº 8 o 9º e no nº 9 foi buscar um dos que ficaram no quarto nº 1, resolvendo, assim, a situação. Como foi possível?

12.3 Colocados em fila indiana (coluna por um), temos dois índios e três brancos. Os índios falam sempre a verdade, os brancos não. Não se pode vê-los. Pode-se chamar dois deles e, em outro local, fazer duas perguntas, uma a cada um. Após isto, pode-se dizer a ordem em que estão colocados na fila. Como?



*"Do mesmo modo que a arte nasceu de um desejo de beleza, a ciência foi criada para atender as necessidades do conhecimento humano."*

*Jan Łukasiewicz*
*(1878-1956)*

# I. INTRODUÇÃO

Neste capítulo veremos alguns recursos e comandos que nos permitirão ilustrar, por meio de *figuras gráficas*, alguns programas que assim o exigirem. Além da página de texto, que normalmente possui 24 linhas por 4Ø colunas, o nosso computador possui mais páginas de trabalho: duas páginas para gráficos de baixa resolução e duas páginas para alta resolução.

Podemos imaginar, como primeira idéia, a página de texto como sendo uma folha comum de caderno, a página de baixa resolução como sendo uma folha quadriculada e a de alta resolução como sendo uma folha milimetrada. O que acontece na realidade não é bem assim, e normalmente a página dois dos gráficos de baixa resolução não é utilizada.

# II. GRÁFICOS DE BAIXA RESOLUÇÃO

O Apple apresenta duas páginas para gráficos de baixa resolução que são duas áreas distintas reservadas na sua memória. Ambas possuem 48 linhas por 4Ø colunas. Veja a figura 13.1.

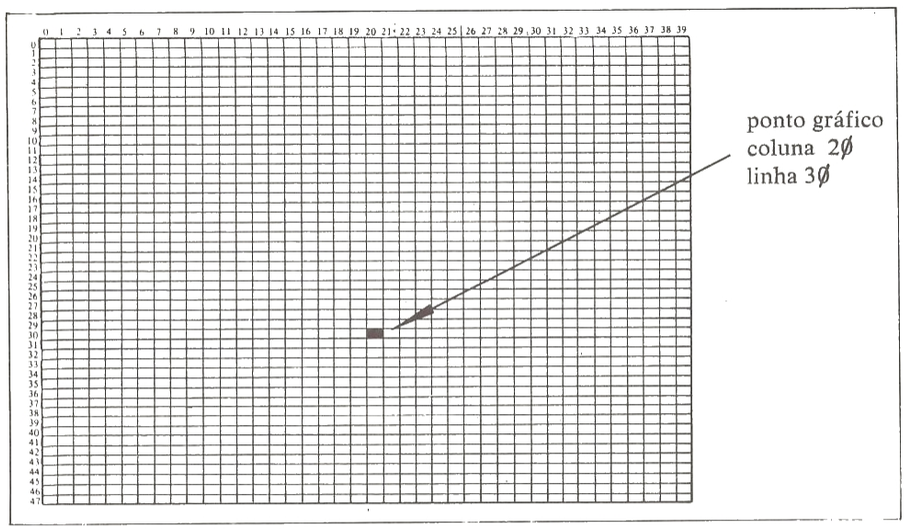



fig. 13.1



Cada página de gráfico tem um total de 192Ø retângulos que denominaremos *pontos gráficos*, e as cores disponíveis para pintá-los estão mostradas na tabela 1.

Tabela 1

| COR | NÚMERO | COR | NÚMERO |
|---|---|---|---|
| preto | Ø | marrom | 8 |
| magenta | 1 | laranja | 9 |
| azul-escuro | 2 | cinza # 2 | 1Ø |
| púrpura | 3 | rosa | 11 |
| verde-escuro | 4 | verde-claro | 12 |
| cinza # 1 | 5 | amarelo | 13 |
| azul-médio | 6 | verde-azulado | 14 |
| azul-claro | 7 | branco | 15 |

Perceba que cada cor tem um número associado, e o programador pode escolher uma determinada pelo número correspondente.

## II.1 – Entrando no Modo Gráfico

A página de gráficos de baixa resolução é a mesma página de texto só que em outro modo. Para entrar no modo gráfico, digite:

**GR**

Esse comando coloca a tela no modo gráfico e a limpa, isto é, pinta de preto todos os pontos (retângulos). Entretanto, a tela gráfica que você está vendo não tem todas as 48 linhas de gráfico mas apenas 4Ø, pois a parte inferior da tela é dedicada a 4 linhas de texto onde você pode digitar instruções ou escrever mensagens. Para conseguir todas as 48 linhas que você tem direito, use o comando:

**POKE - 163Ø2,Ø**

Mas agora tudo o que você escrever sairá na tela em modo gráfico (lembre-se que a página de texto e de gráficos é a mesma). Por isso, esse comando deve ser usado principalmente em programas que não necessitem escrever nada. Para voltar ao gráfico de 4Ø linhas e mais as 4 linhas de texto use o comando:

**POKE - 163Ø1,Ø**

OBS.: O comando **POKE** será visto no próximo capítulo.

Para voltar ao modo de texto, com todas as suas 24 linhas, faça:

**TEXT**

Quando voltamos do modo gráfico para o modo de texto, a tela aparece cheia de caracteres (lembre-se que é a mesma página do gráfico).

## II.2 – Plotando os Retângulos

É muito simples desenhar um ponto na tela de gráficos: em primeiro lugar, entre no modo gráfico, digitando

**GR**

e em seguida escolhendo a cor, usando o comando **COLOR**:

**COLOR = 7**

e, para colocar um ponto azul-claro na posição indicada na figura 13.1, lançamos mão do comando PLOT, digitando:

**PLOT 2Ø, 3Ø**

No comando **PLOT** devemos definir sempre a coluna e em seguida, separada por vírgula, a linha.

Também podemos usar variáveis com os comandos **PLOT** e **COLOR**. Assim, para colocar um ponto azul no canto superior direito podemos fazer:

```
*13.3
C = 2
X = 39
Y = Ø
COLOR = C
PLOT X,Y
```



Geralmente se associa X com a coluna e Y com a linha, como fazemos nos gráficos cartesianos.

## II.3 – Programando com Gráficos

Como primeiro exemplo, vamos mostrar um programa simples para plotar todas as 16 cores:

```
5  HOME
1Ø  GR
2Ø  FOR C = Ø TO 15
3Ø  COLOR= C
4Ø  PLOT 2Ø,3Ø
5Ø  PRINT "COR = ";C
6Ø  FOR T = 1 TO 15ØØ: NEXT T
7Ø  NEXT C
8Ø  GOTO 5
```

P13.1

O que você observou?

**R-1**

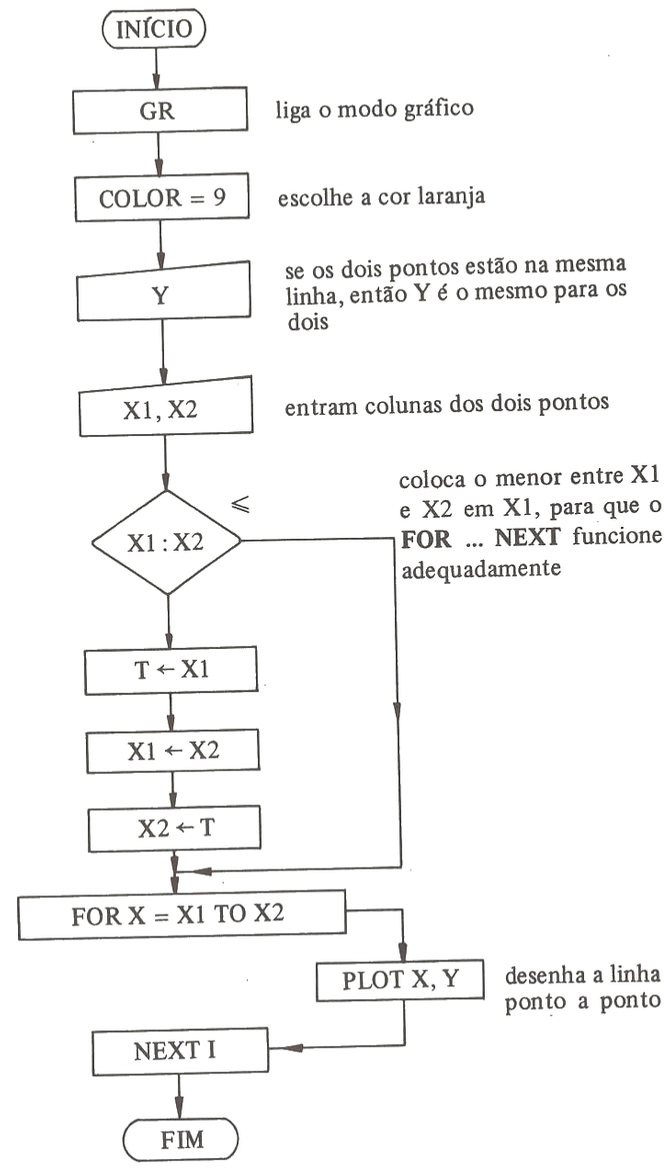
Como podemos concatenar texto e gráfico?

**R-2**

Mude o programa P13.1 para que o computador peça uma cor ao usuário.

Vamos agora analisar um programa que desenha uma linha horizontal entre dois pontos, que obviamente possuem as mesmas ordenadas.

Fluxogramação






Codificação

```
10  REM  PROGRAMA QUE LIGA OS PONTOS (X1,Y) E (X2,Y)
20  GR : REM ENTRA NO MODO GRAFICO
30  COLOR= 9: REM LARANJA
40  INPUT "ENTRE Y (COMUM): ";Y
50  INPUT "ENTRE X1 E X2:";X1,X2
60  IF X1 <  = X2 THEN 80
70 T = X1:X1 = X2:X2 = T
80  FOR X = X1 TO X2
90  PLOT X,Y
100  NEXT X
110  END
```

P13.2

Rode o programa para ligar os pontos 1Ø, 1Ø e 3Ø, 1Ø. Qual a resposta?

**R-3**

Rode o programa P13.2 novamente e digite 3Ø, -5; 15, 1ØØ e -5, 2ØØ. Qual a resposta?

**R-4**

Como exercício faça um programa que desenhe linhas verticais e não aceite valores não permitidos.

## II.4 – Os Comandos VLIN e HLIN

Os comandos **VLIN** e **HLIN** desenham as linhas horizontal (P4.2) e vertical discutidas anteriormente, isto é, ligam 2 pontos entre si, através de uma linha horizontal ou vertical.

**HLIN X1, X2 AT Y**: liga por uma linha horizontal os pontos X1, Y e X2, Y.

**VLIN Y1, Y2 AT X**: liga por uma linha vertical os pontos X, Y1 e X, Y2.

OBS.: É válido tanto **HLIN 1Ø, 3Ø AT 2Ø** como **HLIN 3Ø, 1Ø AT 2Ø**. Ambos produzem o mesmo efeito. Tente.

## II.5 – O Comando SCRN

Esse comando tem a sintaxe **SCRN(*x*, *y*)** e na verdade ele é uma função que retorna o código da cor do ponto *(x, y)*. Só tem sentido usar esse comando na tela de texto de baixa resolução, sendo que x deve ter um valor entre Ø e 39 e y entre Ø e 47.

Esse comando pode ser útil em jogos, quando é importante saber a cor de um determinado ponto da tela.

## II.6 – Alguns Desenhos

Faça o programa P13. 3 e verifique a sua saída:

```
1Ø  REM  DESENHANDO UM "X"
2Ø  GR
3Ø  FOR Y = Ø TO 39
4Ø  COLOR=  RND (1) * 16
5Ø  PLOT Y,Y: PLOT 39 - Y,Y
6Ø  NEXT Y
7Ø  GOTO 3Ø
```

P13.3



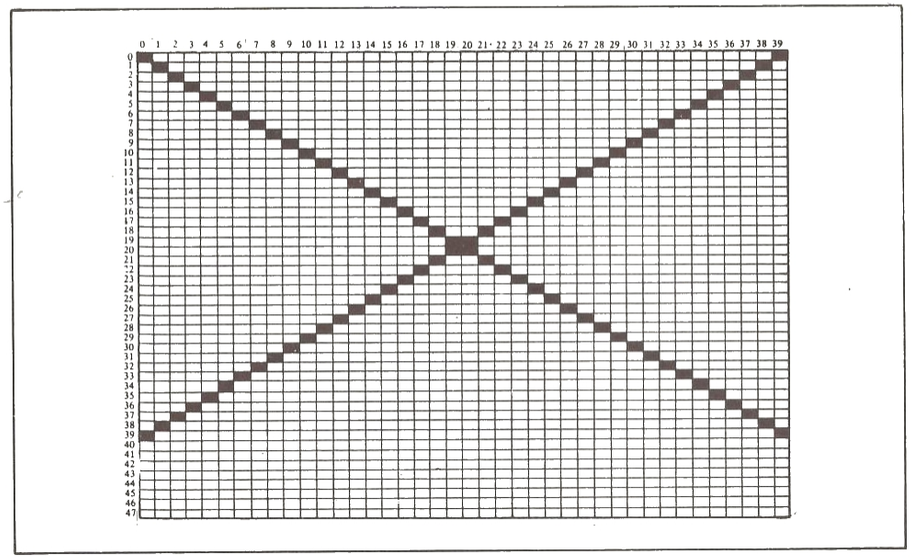


fig. 13.2

O que você obteve na tela deve coincidir com a figura 13.2, com exceção das cores!

Digite uma nova linha 5Ø:

```
5Ø  PLOT X,Y: PLOT 39 - Y,Y: PLOT Ø,Y: PLOT Y,Ø:
    PLOT 39,Y: PLOT Y,39
```

e tente prever o resultado do programa.

## II.7 – Desenhando Gráficos

Para podermos fazer gráficos temos que ajustar a tela do computador ao plano matemático normal, pois, enquanto na tela o ponto (Ø,Ø) é o canto superior direito, no plano matemático, de um modo geral, é o canto inferior esquerdo.

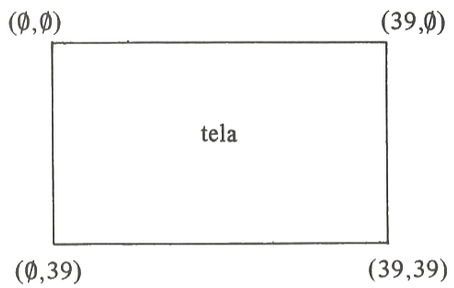



fig. 13.3

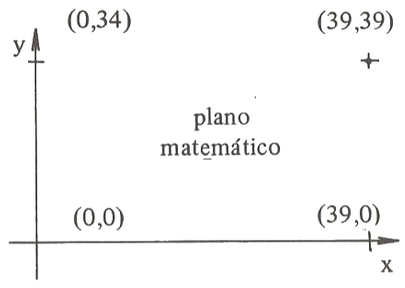



fig. 13.4

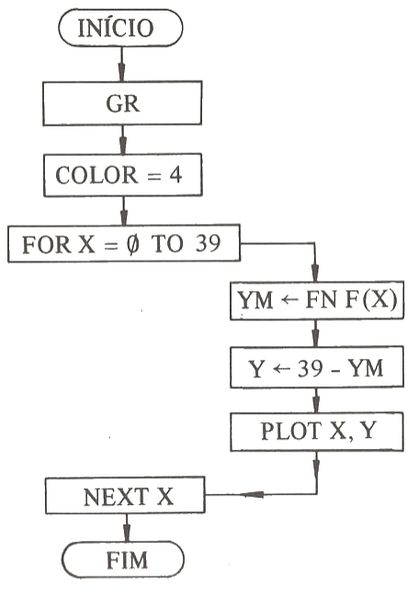
Você deve ter percebido que apenas o Y tem que ser ajustado. Assim, temos a seguinte conversão:

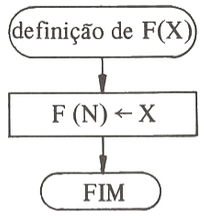
$$Y_{tela} = 39 - Y_{matemático}$$

Veja também que não foi usada a página inteira (as 8 últimas linhas gráficas serão as 4 últimas linhas de texto).

Vamos fazer um programa para graficar a função Y = X.

Fluxograma






Codificação

```
10  REM  GRAFICO DE FUNCOES
20  REM DEFINICAO DA FUNCAO Y=X
30  DEF  FN F(X) = X
40  GR
50  COLOR= 4: REM VERDE
60  FOR X = Ø TO 39
70 YM =  FN F(X)
80 Y = 39 - YM
90  PLOT X,Y
100  NEXT X
110  END
```

P13.4

Rode o programa e veja o resultado.

**R-5**

Perceba que para *graficar* outra função basta mudar a linha **3Ø**. Por exemplo, para *graficar* a função Y = 2x + 3 faça:

```
3Ø  DEF FN F(X) = 2*X + 3
```

Rode o programa e veja o que acontecé.

**R-6**

Veja que às vezes o valor calculado para Y fica fora do valor permitido para as coordenadas do comando **PLOT** (de Ø a 39 para o X e de Ø a 47 para o Y).

No caso acima, quando X = 2Ø, YM = 43 e Y = -4, o que causa uma mensagem de erro, podemos evitar a interrupção com o comando:

```
85  IF Y < Ø THEN Y = Ø
```

Rode o programa com essa linha e tente explicar o que ela faz.

**R-7**

## II.8 – Gráficos de Barra

Gráficos de barra são muito usados no mundo dos negócios e têm a forma de barras de altura proporcional ao valor que se quer representar.

Exemplo:

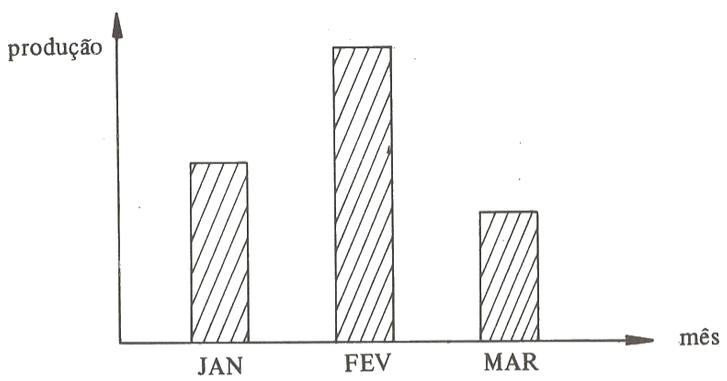
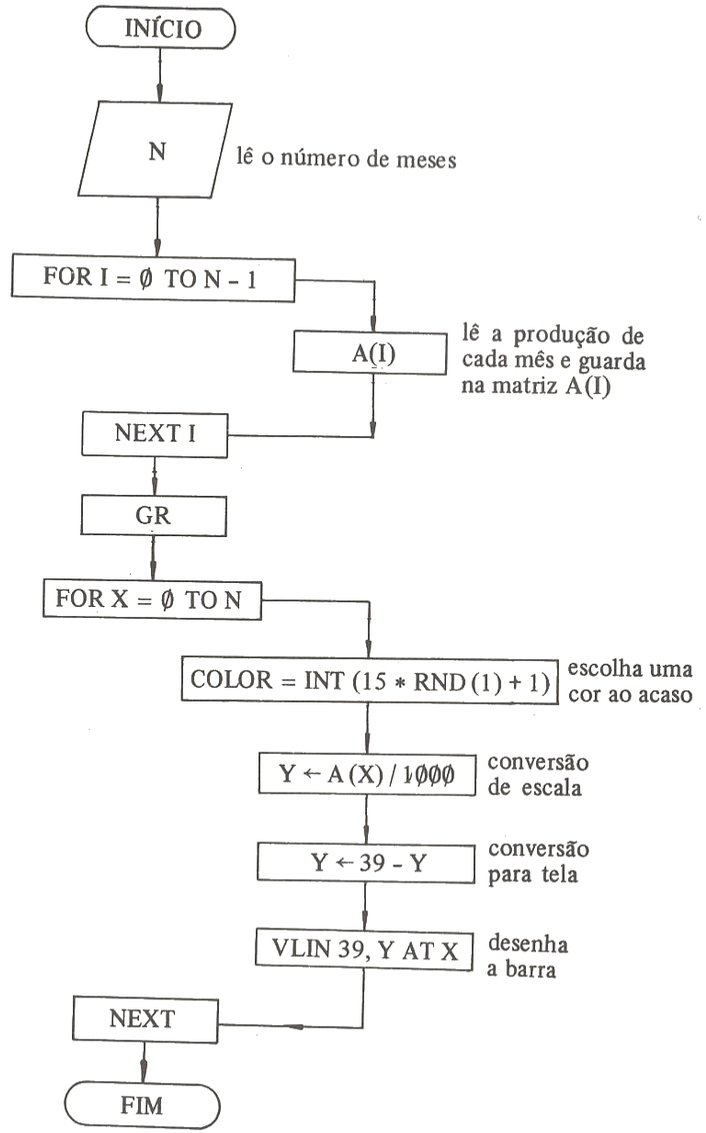



fig. 13.5

Vamos utilizar o computador para fazer um gráfico de barras representando a produção de uma empresa, durante N meses consecutivos.



Fluxograma



Codificação

```
10  REM GRAFICOS DE BARRA
20  READ N: REM NUMERO DE MESES
30  FOR I = 0 TO N - 1
40  READ A(I): REM PRODUCAO DO MES
50  NEXT I
60  GR
70  FOR X = 0 TO N - 1
80  COLOR=  INT (15 *  RND (1) + 1)
90 Y = A(X) / 1000
100 Y = 39 - Y
110  VLIN 39,Y AT X
120  NEXT X
130  DATA 10
140  DATA 5000,10000,20000,15000,35000,7000,18000,23000
,3000,33000
```

P13.5

Rode o programa e veja o resultado.

**R-8**

Perceba que a linha 30 escolhe uma cor qualquer, exceto a 0 (preto).

A linha 90 efetua uma conversão de escala, isto é, cada ponto da tela ao longo do eixo Y representa 1000 unidades de produção. O fator 1000 foi escolhido de acordo com os valores no **DATA**.

Se você acha que os gráficos ficaram muito juntos, mude a linha 110 para

```
110  VLIN 39,Y AT 3*X
```

e veja o resultado. Perceba que você está efetuando uma mudança de escala, agora no eixo X.

Seria interessante escrever o número do mês que a barra representa embaixo de cada barra. Podemos fazer isto com a linha:

```
125  FOR I = 1 TO N: PRINT TAB(I*2 - 1);I: NEXT I
```

Rode o programa P13.5 com essa modificação e verifique a sua resposta.



Como último exemplo para gráfico em baixa resolução, vamos analisar o programa seguinte, que faz uma "bolinha" se mover horizontalmente na tela, indo e vindo entre seus limites. Tente explicar como funciona o programa:

```
10  REM  ESCOLHE A COR DA BOLA
20  INPUT "COR DA BOLA? ";BOLA
30  GR
40  REM POSICAO INICIAL
50 X1 = 0
60  REM MOVE A BOLA PARA A DIREITA
70  FOR X = 0 TO 39 STEP 1
80  COLOR= 0
90  PLOT X1,20
100  COLOR= BOLA
110  PLOT X,20
120 X1 = X
130  NEXT X
140  REM MOVE A BOLA PARA A ESQUERDA
150  FOR X = 39 TO 0 STEP  - 1
160  COLOR= 0
170  PLOT X1,20
180  COLOR= BOLA
190  PLOT X,20
200 X1 = X
210  NEXT X
220  GOTO 50
```

P13.6

Como é que esse programa faz a bola se mexer? Como ele apaga as bolas que já foram desenhadas?

**R-9**

## III. BRINCANDO COM SOM

Cada vez que se executa um **PEEK (-16336)** o alto-falante do Apple produz um *click*. Podemos usar isso para gerar um barulhinho cada vez que a bola do programa P13.6 toca na parede.

A maneira mais fácil de se executar um **PEEK (-16336)** é através da instrução:

**SOM** (ou qualquer outra variável) = **PEEK (- 16336)**

Assim sendo, adicione as seguintes linhas ao programa P13.6 e em seguida rode-o:

```
135  FOR I = 1 TO 10: S = PEEK(-16336): NEXT I
215  FOR I = 1 TO 10: S = PEEK(-16336): NEXT I
```

OBS.: O comando **PEEK** será analisado no próximo capítulo.

## IV. GRÁFICOS DE ALTA RESOLUÇÃO

Além desse modo gráfico, de 40 colunas por 48 linhas, o Apple apresenta o modo de alta resolução ou de resolução mais fina, com 280 colunas e 192 linhas, num total de 53760 pontos, denominados *pixels*. Com isso, aumentamos a resolução em 7 vezes na horizontal e 4 na vertical.

Para entrarmos nos gráficos de alta resolução, usamos o comando:

**HGR**

Esse comando também deixa 4 linhas de texto na parte inferior da tela. O comando **POKE -16302,0** coloca todas as 192 linhas de gráfico à disposição, e você pode escrever à vontade na página de texto que isso não afetará os gráficos (os gráficos de alta resolução não têm nada a ver com a página de texto). O comando **POKE -16301,0** retorna as 4 linhas de texto no fim da tela de gráfico.

Escolhemos a cor a ser usada nos gráficos de alta resolução, através do comando

**HCOLOR** = nº da cor

Os códigos das cores são mostrados na tabela 2.



Tabela 2

| COR | NÚMERO | COR | NÚMERO |
|---|---|---|---|
| Preto # 1 | 0 | Preto # 2 | 4 |
| Verde | 1 | Laranja | 5 |
| Violeta | 2 | Azul | 6 |
| Branco # 1 | 3 | Branco # 2 | 7 |

Como você percebeu, há bem menos cores a escolher, mas em compensação podemos fazer desenhos muito mais finos.

Para colocar um ponto da cor escolhida num lugar da tela, usamos o comando

**HPLOT X, Y**

onde X é a coluna e Y a linha.

A figura 13.6 mostra os 4 pontos que delimitam a tela dos gráficos de alta resolução.

A tela tem a distribuição

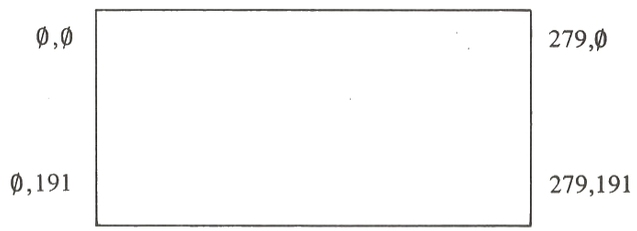



fig. 13.6

Não há comandos nos gráficos de alta resolução equivalentes a **HLIN** e **VLIN**, mas podemos ligar dois pontos quaisquer por uma linha através do comando

**HPLOT X1, Y1 TO X2, Y2**

Nesse comando não é necessário que a linha a ser desenhada seja horizontal ou vertical (ou seja, X1 pode ser diferente de X2 e Y1 pode ser diferente de Y2).

Faça o programa a seguir:

```
10  HGR
20  POKE  - 16302,0
30 C =  RND (1) * 8
40 X =  RND (1) * 280
50 Y =  RND (1) * 192
60  HCOLOR= C
70  HPLOT X,Y
80  GOTO 30
```

P13.7

Rode o programa P13.7. O que ele faz?

**R-10**

Altere P13.7, introduzindo:

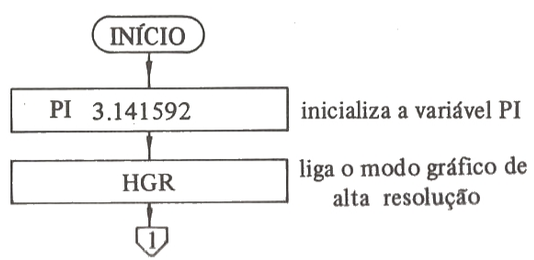
```
30  HCOLOR= 5: HPLOT 0,0

70  HPLOT TO X,Y
```

Rode novamente P13.7. Qual foi a mudança observada?

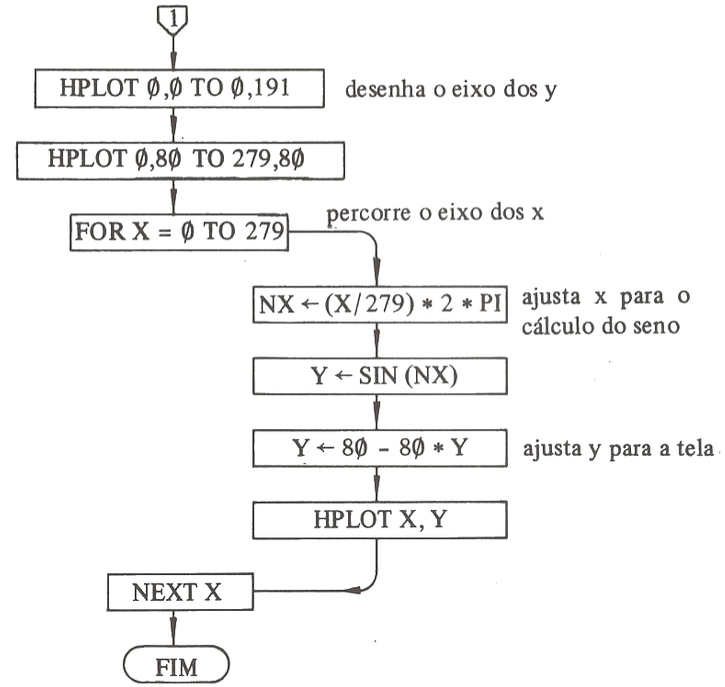
**R-11**

Como um outro exemplo de aplicação vamos fazer o gráfico de sen (x) de Ø até $2\pi$. Lembre-se que o seno de um ângulo está sempre entre -1 e 1.

Fluxograma






Codificação

```
10  HOME
20 PI = 3.14159265
30  HGR : HCOLOR= 3
40  HPLOT Ø,Ø TO Ø,191
50  HPLOT Ø,8Ø TO 279,8Ø
60  FOR X = Ø TO 279
70 NX = (X / 280) * 2 * PI
80 Y =  SIN (NX)
90 Y = 8Ø - 8Ø * Y
100  HPLOT X,Y
110  NEXT X
120  END
```

P13.8

Um efeito visual melhor pode ser conseguido adicionando-se a linha

```
55  HPLOT Ø,8Ø
```

e mudando a linha 1ØØ para

```
1ØØ  HPLOT TO X,Y
```

Essa instrução desenha uma linha contínua desde o último ponto "plotado" até o ponto X, Y.

Explique então a finalidade da linha 55.

**R-12**

## IV.1 – O Comando HGR2

Esse comando *liga* a 2ª página de gráficos de alta resolução e só é acessível em computadores de 24 K ou mais de memória, sem o DOS, e 36 K de memória para o uso simultâneo de gráficos e DOS.

A 2ª página é toda de gráficos, sem as 4 linhas de texto, e é perfeitamente possível fazer um desenho na página 2 com outro desenho guardado na página 1. Esse fato nos leva a fazer **animação** com o Apple, isto é, podemos desenhar uma figura em **HGR1** e apresentá-la; redesenhá-la ligeiramente modificada em **HGR2** e apresentá-la; redesenhá-la em **HGR1** e apresentá-la; e assim sucessivamente.

## IV.2 – POKEs Controlando a Tela

É possível entrar no modo gráfico sem apagar o conteúdo anterior. Este fato é possível através do comando **POKE**.

1) texto . . . . . **POKE -163Ø3,Ø**
   ou gráficos, alta ou baixa resolução . . . . . **POKE -163Ø4,Ø**



2) página 1 . . . . . **POKE - 163ØØ,Ø**
   página 2 . . . . . **POKE - 16299,Ø**

3) texto e baixa resolução . . . . . **POKE - 16298,Ø**
   alta resolução . . . . . **POKE - 16297,Ø**

4) página inteira de gráficos . . . . . **POKE - 163Ø2,Ø**
   gráficos e texto juntos . . . . . **POKE - 163Ø1,Ø**

Exemplo: se quisermos ligar a página 2 de gráficos de alta resolução, sem as 4 linhas de texto, digitamos a instrução:

```
POKE -16304,Ø: POKE -16299,Ø: POKE -16297,Ø:
POKE -16302,Ø
```

Observações:

1. A página 2 de texto é ininteligível, portanto não há sentido usar página 2 de gráficos com as 4 linhas de texto.

2. Quando ligamos uma página de gráficos através de **POKEs**, o que antes estava desenhado na tela *não é apagado* (o que pode ser muito útil em alguns casos). Para apagar a página de alta resolução que está sendo usada faça **CALL - 6245Ø**. Para apagar a página 1 de baixa resolução use **CALL - 1998**.

Para finalizar, analise os programas P13.9 e P13.1Ø.

```
1Ø  HGR : HCOLOR= 3: HOME : VTAB 21
2Ø  REM EIXOS
3Ø  HPLOT 1ØØ,Ø TO 1ØØ,15Ø
4Ø  FOR I = Ø TO 15
5Ø  HPLOT 1Ø1,1Ø * I
6Ø  NEXT I
7Ø  HPLOT Ø,1ØØ TO 2ØØ,1ØØ
8Ø  FOR I = Ø TO 2Ø
9Ø  HPLOT 1Ø * I,99
1ØØ  NEXT I
11Ø  REM  PLOTAR PONTOS
12Ø  FOR I =  - 7 TO 1Ø STEP .1
13Ø X = I
14Ø Y = .2 * X ^ 3 - 2 * X ^ 2 - 7 * X + 5Ø
15Ø  HPLOT 1ØØ + 1Ø * X,1ØØ - Y
16Ø  NEXT I
17Ø  REM  TEXTO
18Ø  PRINT "GRAFICO DA EQUACAO"
19Ø  PRINT "Y = .2X^3 - 2X^2 - 7X + 5Ø"
2ØØ  PRINT "CADA DIVISAO VALE DEZ UNIDADES"
21Ø  GOTO 21Ø
```

P13.9

Tente, como exercício, adaptar o programa P13.9 para estudar e *graficar* a função

$$y = e^{-(x^3 - 3x)}$$

```
1Ø  HGR : HCOLOR= 3: HOME : VTAB 21
2Ø  REM EIXOS
3Ø  HPLOT 1Ø,9 TO 1Ø,159 TO 2ØØ,159
4Ø  FOR I = Ø TO 15
5Ø  HPLOT 11,9 + 1Ø * I
6Ø  NEXT I
7Ø  FOR I = 1 TO 6
8Ø  HPLOT 28 * I,158
9Ø  NEXT I
1ØØ  REM  PLOTAR
11Ø  READ SV
12Ø  FOR I = 1 TO 5
13Ø  READ S
14Ø  HPLOT 28 * I,159 - SV TO 28 * (I + 1),159 - S
15Ø SV = S
16Ø  NEXT I
17Ø  DATA 48,69,79,34,56,12Ø
18Ø  PRINT "   78  79  8Ø  81  82  83  VENDAS"
19Ø  PRINT : PRINT "CADA DIVISAO VALE DEZ MIL UNIDADES"
2ØØ  GOTO 2ØØ
```

P13.1Ø

## V. EXERCÍCIOS PROPOSTOS

13.1 Modifique o problema da bola que rebate nos limites laterais da tela para que a bola ande um número aleatório de pontos para frente ou para trás, para baixo ou para cima.

13.2 Faça um programa para graficar a função cosseno, com um número de ciclos a ser escolhido pelo usuário. Note que a função cosseno é limitada entre -1 e 1. Use gráficos de alta resolução.

13.3 Usando as páginas 1 e 2 dos gráficos de alta resolução e o comando **POKE**, estruture um programa que "desenhe" um boneco movimentando-se. Exemplo:



13.4 Usando a técnica dos gráficos de alta resolução, grafique as funções $y = x^3 - 3x + 3$ (veja programa P13.9).

13.5 Construa um gráfico de linha, usando alta resolução, dos dados abaixo, que fornecem o tempo médio de vida de homens e mulheres.

| ANO | HOMEM | MULHER |
|---|---|---|
| 1920 | 53.5 | 54.5 |
| 1930 | 58.0 | 61.5 |
| 1940 | 61.0 | 65.0 |
| 1950 | 65.5 | 71.0 |
| 1960 | 66.5 | 73.5 |
| 1970 | 67.0 | 75.0 |

sob a seguinte condição:

a) O computador deve perguntar o sexo.
b) No vídeo, com diferentes cores, ambos os gráficos deverão ser mostrados.

## VI. RAXAKUKA

13.1 Se de um número de 3 algarismos que eu pensei, subtrairmos 7, então ele se tornará divisível por 7; se subtrairmos 8, ele se tornará divisível por 8; e, se subtrairmos 9, ele se tornará divisível por 9.
Qual é esse número?

13.2 Nas igualdades abaixo, os algarismos foram substituídos por letras.
Descubra, através de raciocínio lógico, quais os números que estão ocultos. A cada letra corresponde um só algarismo.

DO + RE = MI
FA + SI = LA
RE + SI + LA = SOL

13.3 "As 4 operações"

1 2 3 = 1
1 2 3 4 = 1
1 2 3 4 5 = 1
1 2 3 4 5 6 = 1
1 2 3 4 5 6 7 = 1
1 2 3 4 5 6 7 8 = 1
1 2 3 4 5 6 7 8 9 = 1

Você é capaz de tornar estas igualdades válidas usando somente os sinais: (+) (-) (÷) e (·)?

Observações:

a) Não é permitida a mudança da localização dos algarismos.
b) Se for necessário, dois algarismos consecutivos poderão ser considerados como um número de dois algarismos.
c) Você pode usar parênteses, colchetes e chaves.
d) Quando não usar os sinais da observação (c), as operações deverão ser realizadas, consecutivamente, da esquerda para a direita.
e) Tente duas soluções: uma usando somente $\mp$ e outra com os 4 sinais.

# PEEK, POKE, CALL

*"Não há máquina capaz de mentir, mas não há máquina capaz de dizer toda a verdade."*

*G. K. Chesterton*
*(1874-1936)*

## I. INTRODUÇÃO

A memória do Apple está dividida em 65536 ($2^{16}$) unidades, chamadas *bytes*. Cada *byte* pode armazenar um número de Ø a 255 ($2^8$ possibilidades). Todas as informações que o computador guarda estão codificadas numa série de números, que podem ser lidos e modificados pelo microprocessador. Esses números podem ser examinados e mudados (quase todos) diretamente pelo programador, através dos comandos **PEEK** e **POKE**.

## II. OS COMANDOS PEEK E POKE

O **PEEK** permite ao programador ler o conteúdo de qualquer *byte* da memória do computador. Na verdade, esse comando é uma função e, como toda função, tem a forma

**PEEK(*adr*)**

onde *adr* é o endereço da memória, um número entre Ø e 65535, indicando qual o *byte* a ser lido.

Digite:

```
PRINT PEEK(16383)
```

**R-1**

O computador responde com um número entre Ø e 255, provavelmente 179 se você estiver usando disco.



Tabela 1: OS CARACTERES ASCII

| | Ø | 16 | 32 | 48 | 64 | 8Ø | 96 | 112 |
|---|---|---|---|---|---|---|---|---|
| Ø | nul | dle | | Ø | @ | P | | p |
| 1 | soh | dc1 | ! | 1 | A | Q | a | q |
| 2 | stx | dc2 | ” | 2 | B | R | b | r |
| 3 | etx | dc3 | # | 3 | C | S | c | s |
| 4 | eot | dc4 | $ | 4 | D | T | d | t |
| 5 | enq | nak | % | 5 | E | U | e | u |
| 6 | ack | syn | & | 6 | F | V | f | v |
| 7 | bel | etb | ’ | 7 | G | W | g | w |
| 8 | bs | can | ( | 8 | H | X | h | x |
| 9 | ht | em | ) | 9 | I | Y | i | y |
| 1Ø | lf | sub | * | : | J | Z | j | z |
| 11 | vt | esc | + | ; | K | [ | k | { |
| 12 | ff | fs | , | < | L | \ | l | \| |
| 13 | cr | gs | – | = | M | ] | m | } |
| 14 | so | rs | . | > | N | ^ | n | ~ |
| 15 | si | us | / | ? | O | – | o | rub |

Analise a tabela 1.

Com base na tabela 1, o que significa esse número? Se o número for maior que 127, subtraia 128 dele.

**R-2**

Tente:

```
PRINT CHR$(PEEK(16383))
```

**R-3**

Digite:

```
PRINT PEEK(2053)
```

**R-4**

Por que esse número não pode ser maior que 255?

**R-5**

Qual o caractere, de acordo com a tabela 1, que aparecerá como resposta ao comando

```
PRINT CHR$(PEEK(2053))
```

**R-6**

**Digite agora:**

```
POKE 16383,75
```

**R-7**

O que fez esse comando, na sua opinião?



**R-8**

Como provar que o conteúdo do *byte* 16383 foi alterado?

**R-9**

Qual a resposta para

```
PRINT CHR$(PEEK(16383))
```

**R-10**

A sintaxe do comando **POKE** é

**POKE** *adr*,*x*

sendo *adr* o endereço da memória (como no **PEEK**) e *x* um número entre Ø e 255. O número *x* será colocado na posição *adr*.

## III. A MEMÓRIA DA TELA

Cada caractere representado na tela corresponde a um *byte* na memória do computador, organizado como mostra a tabela 2. Cada vez que o programa manda alguma informação à tela, essa informação é codificada conforme a tabela 3 e gravada, um caractere por vez, na memória de texto, na posição correta.

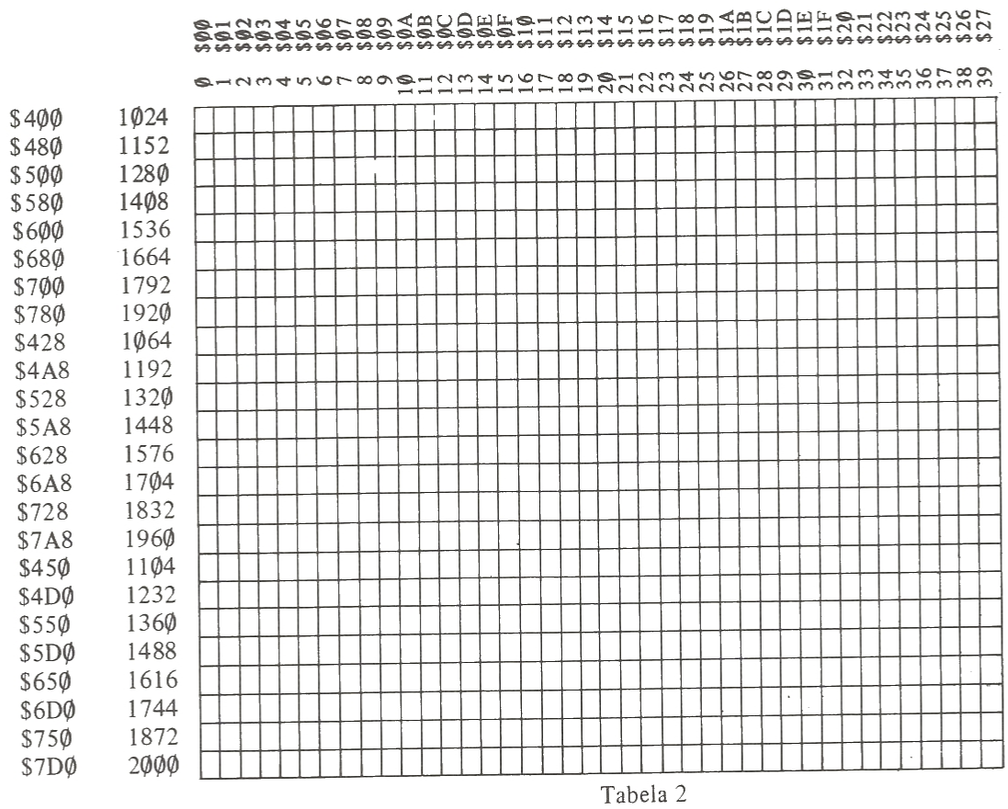



Tabela 2

Tabela 3

| | INVERSE | | | | FLASH | | | | CONTROL | | NORMAL | | | | Minúsculas | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Decimal | Ø | 16 | 32 | 48 | 64 | 8Ø | 96 | 112 | 128 | 144 | 16Ø | 176 | 192 | 2Ø8 | 224 | 24Ø |
| Hex | $ØØ | $1Ø | $2Ø | $3Ø | $4Ø | $5Ø | $6Ø | $7Ø | $8Ø | $9Ø | $AØ | $BØ | $CØ | $DØ | $1Ø | $1Ø |
| Ø$Ø | @ | P | | Ø | @ | P | | Ø | @ | P | | Ø | @ | P | | Ø |
| 1$1 | A | Q | ! | 1 | A | Q | ! | 1 | A | Q | ! | 1 | A | Q | ! | 1 |
| 2$2 | B | R | ” | 2 | B | R | ” | 2 | B | R | ” | 2 | B | R | ” | 2 |
| 3$3 | C | S | # | 3 | C | S | # | 3 | C | S | # | 3 | C | S | # | 3 |
| 4$4 | D | T | $ | 4 | D | T | $ | 4 | D | T | $ | 4 | D | T | $ | 4 |
| 5$5 | E | U | % | 5 | E | U | % | 5 | E | U | % | 5 | E | U | % | 5 |
| 6$6 | F | V | & | 6 | F | V | & | 6 | F | V | & | 6 | F | V | & | 6 |
| 7$7 | G | W | ‘ | 7 | G | W | ‘ | 7 | G | W | ‘ | 7 | G | W | ‘ | 7 |
| 8$8 | H | X | ( | 8 | H | X | ( | 8 | H | X | ( | 8 | H | X | ( | 8 |
| 9$9 | I | Y | ) | 9 | I | Y | ) | 9 | I | Y | ) | 9 | I | Y | ) | 9 |
| 1Ø$A | J | Z | * | : | J | Z | * | : | J | Z | * | : | J | Z | * | : |
| 11$B | K | [ | + | ; | K | [ | + | ; | K | [ | + | ; | K | [ | + | ; |
| 12$C | L | \ | , | < | L | \ | , | < | L | \ | , | < | L | \ | , | < |
| 13$D | M | ] | – | = | M | ] | – | = | M | ] | – | = | M | ] | – | = |
| 14$E | N | ^ | . | > | N | ^ | . | > | N | ^ | . | > | N | ^ | . | > |
| 15$F | O | – | / | ? | O | – | / | ? | O | – | / | ? | O | – | / | ? |

Escreva:

```
HOME
POKE 1024,65
```

O que apareceu na tela?

**R-11**

Como você explica isso?

**R-12**

Digite:

```
POKE 1380,20
```

Qual a resposta?

**R-13**

Experimente **POKE**ar algumas posições da tela, conferindo os valores com a tabela 3.

Como verificação final digite o programa abaixo:

```
10  HOME
20  FOR I = 1024 TO 2039
30  POKE I,J
40 J = J + 1
50  IF J = 255 THEN J = 0
60  NEXT I
70  GOTO 70
```

P14.1



O que faz esse programa?

**R-14**

## IV. REGULANDO O TAMANHO DA TELA

Através do comando **POKE**, podemos definir margens na tela de texto, nos seus quatro lados. As posições de memória 32, 33, 34 e 35 controlam, respectivamente, margem esquerda, largura de linha, margem superior e margem inferior da tela. Veja a tabela 4.

Tabela 4

| POSIÇÕES ESPECIAIS DA JANELA DE TEXTO | | |
|---|---|---|
| Função | Posição | Mínimo/Normal/Máximo |
| M. Esq. | 32 | Ø / Ø / 39 |
| Largura | 33 | Ø / 4Ø / 4Ø |
| M. Sup. | 34 | Ø / Ø / 24 |
| M. Inf. | 35 | Ø / 24 / 24 |

Digite:

```
POKE 32,15: POKE 33,10: HOME
```

O que aconteceu?

**R-15**

Os valores máximos e mínimos para as quatro posições estão na tabela 4. Devemos ter cuidado para não ultrapassar esses limites. Isso pode gerar resultados inesperados e, às vezes, desastrosos.

Defina agora uma tela com os valores 1Ø, 1Ø, 1Ø e 2Ø, nessa ordem:

```
FOR Y = 1 TO 23: FOR X = 1 TO 40: PRINT "X";:
   NEXT Y,X: POKE 32,1Ø: POKE 33,1Ø: POKE 34,1Ø:
   POKE 35,2Ø: HOME
```

**R-16**

Veja que tudo o que estava na tela, fora das margens, se conserva inalterado, mesmo quando o texto dentro da janela se move. Pressione *RETURN* junto com *REPT* para confirmar isso. A seguir, pressione *I,* junto com *REPT*. Qual a resposta?

**R-17**

Como podemos fazer para escrever fora dessa "janela"? Uma solução seria temporariamente redefini-la para uma nova posição. Outro método é simplesmente posicionar o cursor, com **VTAB** e **HTAB**, ou **HOME**. Este último limpa apenas o texto da janela e posiciona o cursor dentro dela. O **VTAB** ignora completamente a janela de texto: escrevendo acima da janela tudo funciona normalmente, enquanto escrevendo abaixo da janela tudo aparece na mesma linha. O **HTAB** posiciona o cursor em relação à margem esquerda da janela, e pode mover o cursor para fora da janela apenas o suficiente para imprimir um único caractere.

Digite:

```
VTAB 5: PRINT "ASD"
```

**R-18**

```
VTAB 22: PRINT "ASD"
```



**R-19**

```
VTAB 3: HTAB 3Ø: PRINT "+++"
```

**R-2Ø**

Pressione *RESET*. O que ocorreu?

**R-21**

Digite o programa abaixo:

```
1Ø  POKE 32,Ø: POKE 33,4Ø: POKE 34,Ø: POKE 35,24
2Ø  HOME
3Ø  PRINT  TAB( 7);"LISTA DE STRINGS ALEATORIOS"
4Ø  PRINT "=======================================": R
EM  4Ø ='S
5Ø  POKE 34,4
6Ø  HOME
7Ø A$ = ""
8Ø  FOR I = 1 TO  RND (1) * 4 + 3
9Ø A$ = A$ +  CHR$ ( RND (1) * 26 + 65)
1ØØ  NEXT I
11Ø  FOR I = 1 TO 3ØØ: NEXT I
12Ø  PRINT A$
13Ø  GOTO 7Ø
```

P14.2

Qual a função da linha 1Ø?

**R-22**

Quantos caracteres são gerados, no máximo, em cada **FOR ... NEXT**?

**R-23**

Que faz a instrução 9Ø?

**R-24**

Por que é repetido o **HOME** na linha 6Ø?

**R-25**

## V. O COMANDO CALL

Internamente, o computador usa uma linguagem binária, diferente do BASIC. Dentro do computador há diversos programas, gravados permanentemente em memórias especiais, ROM, nessa linguagem de máquina. Eles fazem tarefas simples, mas muitas delas não podem ser efetuadas diretamente em BASIC. Para isso, existe o comando **CALL**, que transfere o controle do computador, do BASIC, para um programa binário, no endereço especificado.

**CALL***adr*

faz a execução "pular" para o endereço *adr,* de maneira semelhante a um desvio incondicional para uma sub-rotina, em BASIC (**GOSUB**). Terminada a sub-rotina, a execução volta ao programa BASIC.



Seguem alguns exemplos de sub-rotinas úteis:

**CALL -936** (ou 64600):

Limpa a janela de texto e leva o cursor à sua posição superior esquerda. É igual a *ESC @* ou **HOME**.

**CALL -958** (ou 64578):

Limpa a janela de texto, desde a posição do cursor até a margem inferior. No modo direto, use *ESC F.*

**CALL -868** (ou 64668):

Limpa a linha de texto, do cursor até a margem direita. É o mesmo que *ESC E* no modo direto.

**CALL -912** (ou 64624):

O texto dentro da janela é "levantado" uma linha, e a última linha é limpa.

O apêndice 4 apresenta uma lista mais completa de sub-rotinas (acessadas por **CALL**) com as quais você pode experimentar.

## VI. PEEK E O TECLADO

O teclado manda sete *bits* de informação que, juntos, formam um caractere. Estes sete *bits,* junto com outro sinal que indica quando uma tecla é apertada, estão disponíveis à maior parte dos programas como o conteúdo de uma posição de memória. Programas podem ler o estado atual do teclado, lendo o conteúdo desta posição.

Quando você pressiona uma tecla no teclado, o valor desta posição fica sendo 128 ou mais, e o valor assumido é o código numérico para o caractere que foi digitado.

A tabela 5 mostra os caracteres ASCII e seus respectivos códigos numéricos. A posição guardará este valor até que você pressione outra tecla ou até que seu programa "diga" a essa posição de memória para "esquecer" o caractere que está segurando.

Tabela 5

AS TECLAS E SEUS RESPECTIVOS CÓDIGOS

| Tecla | Única | *CTRL* | *SHIFT* | Ambas |
|---|---|---|---|---|
| *RETURN* | 141 | 141 | 141 | 141 |
| G | 199 | 135 | 199 | 135 |
| H | 200 | 136 | 200 | 136 |
| I | 201 | 137 | 201 | 137 |
| J | 202 | 138 | 202 | 138 |
| K | 203 | 139 | 203 | 139 |
| L | 204 | 140 | 204 | 140 |
| M | 205 | 141 | 221 | 157 |
| N | 206 | 142 | 222 | 158 |
| O | 207 | 143 | 207 | 143 |
| P | 208 | 144 | 192 | 128 |
| Q | 209 | 145 | 209 | 145 |
| R | 210 | 146 | 210 | 146 |
| S | 211 | 147 | 211 | 147 |
| T | 212 | 148 | 212 | 148 |
| U | 213 | 149 | 213 | 149 |
| V | 214 | 150 | 214 | 150 |
| W | 215 | 151 | 215 | 151 |
| X | 216 | 152 | 216 | 152 |
| Y | 217 | 153 | 217 | 153 |
| Z | 218 | 154 | 218 | 154 |
| → | 136 | 136 | 136 | 136 |
| ← | 149 | 149 | 149 | 149 |
| *ESC* | 155 | 155 | 155 | 155 |

| Tecla | Única | *CTRL* | *SHIFT* | Ambas |
|---|---|---|---|---|
| *espaço* | 160 | 160 | 160 | 160 |
| Ø | 176 | 176 | 176 | 176 |
| 1 ! | 177 | 177 | 161 | 161 |
| 2 " | 178 | 178 | 162 | 162 |
| 3 # | 179 | 179 | 163 | 163 |
| 4 $ | 180 | 180 | 164 | 164 |
| 5 % | 181 | 181 | 165 | 165 |
| 6 & | 182 | 182 | 166 | 166 |
| 7 ' | 183 | 183 | 167 | 167 |
| 8 ( | 184 | 184 | 168 | 168 |
| 9 ) | 185 | 185 | 169 | 169 |
| : * | 186 | 186 | 170 | 170 |
| ; + | 187 | 187 | 171 | 171 |
| , < | 172 | 172 | 188 | 188 |
| - = | 173 | 173 | 189 | 189 |
| . > | 174 | 174 | 190 | 190 |
| / ? | 175 | 175 | 191 | 191 |
| A | 193 | 129 | 193 | 129 |
| B | 194 | 130 | 194 | 130 |
| C | 195 | 131 | 195 | 131 |
| D | 196 | 132 | 196 | 132 |
| E | 197 | 133 | 197 | 133 |
| F | 198 | 134 | 198 | 134 |



Uma vez que seu programa aceitou e entendeu um aperto de tecla, este deve dizer à posição de memória do teclado para "largar" o caractere que está sendo segurado e se preparar para receber outro. Seu programa pode fazer isto referenciando outra posição de memória.

Quando você referencia esta posição, o valor contido na primeira posição cairá para um valor inferior a 128. Este valor continuará baixo até que você aperte outra tecla. Esta ação é chamada de "zerar o *strobe* de teclado". Seu programa pode ler ou escrever para esta posição especial; os dados escritos ou lidos são irrelevantes. É a mera REFERÊNCIA a esta posição que zera o *strobe*.

Uma vez que você limpou o *strobe* do teclado, você ainda pode recuperar o código para a tecla que foi pressionada por último, somando 128 ao valor na posição de memória do teclado.

Estas são as posições especiais de memória usadas pelo teclado.

Tabela 6

POSIÇÕES ESPECIAIS DO TECLADO

| Posição: | Descrição: |
|---|---|
| 49152 ou -16384 | Dados do Teclado |
| 49168 ou -16368 | Zerar *Strobe* |

Como um programa "sabe" se uma tecla foi pressionada ou não?

**R-26**

Veja o seguinte programa:

```
10  HOME : PRINT "ESTOU ESPERANDO POR UMA TECLA"
20  PRINT "   (SEM O CURSOR)"
30 P =  PEEK ( - 16384)
40  IF P < 128 THEN 30
50  POKE  - 16368,0
60  PRINT : PRINT "VOCE PRESSIONOU A TECLA DE CODIGO ";
P
70  PRINT "QUE NO VIDEO APARECE COMO "; CHR$ (P)
```

P14.3

Analise e explique o funcionamento do programa.

**R-27**

## VII. EXERCÍCIOS PROPOSTOS

14.1 Faça um programa para listar todos os *bytes* da memória.

14.2 Os cinco caracteres que compõem a palavra reservada **PRINT** estão armazenados em posições sucessivas da memória. Os quatro primeiros caracteres estão representados pelos seus *ASCII-code* e o último caractere pelo seu *ASCII-code* mais 127. Descubra a primeira memória.

14.3 Faça um programa para listar somente as palavras reservadas do Applesoft.

14.4 Uma vez carregado o **DOS**, verifique se os endereços das memórias dos exercícios anteriores foram alterados.

14.5 Faça uma rotina para trocar a página de texto com a página de gráfico de baixa ou alta resolução, a critério do usuário.

14.6 Desenhe na página de gráficos de alta resolução um *grid* de 24 linhas por 4Ø colunas, ocupando proporcionalmente a tela inteira e use a rotina acima.

14.7 Usando as idéias dos exercícios 14.5 e 14.6, estruture um jogo de batalha naval.

14.8 Sabendo que a página 1 de alta resolução começa no *byte* 8.192 e termina em 16.383 e a página 2 vai de 16.384 até 24.575, faça uma sub-rotina para trocar as duas páginas de gráfico.

14.9 Faça um programa que pesquise os *bytes* da ROM iniciando em 53.248. Usando a página gráfica de alta resolução, estruture esse programa para construir histograma da distribuição dos valores dos *bytes*, evidenciando assim o *byte* que ocorre mais.



## VIII. RAXAKUKA

14.1 Um jovem mora perto da estação do metrô de Manhattan.

Ele tem duas namoradas, uma no Brooklyn e outra em Bronx.

Para visitar a garota do Brooklyn, ele toma o metrô na plataforma do lado do centro comercial; para visitar a garota do Bronx, ele toma o metrô na plataforma do lado residencial. Como gosta igualmente das duas namoradas, ele simplesmente toma o primeiro trem que chega. Assim ele deixa o destino escolher se deve ir ao Bronx ou ao Brooklyn.

Todo sábado, à tarde, ele chega à estação.

Tanto o trem de Bronx como o de Brooklyn irrompem na gare ferroviária de dez em dez minutos.

Entretanto, alguma razão desconhecida faz com que nosso amigo passe mais tempo com a garota do Brooklyn! Na verdade, em cada 10 visitas, 9 são feitas a esta garota, com quem afinal ele vai acabar se casando.

Você sabe dizer porque o destino favoreceu tanto a garota do Brooklyn?

14.2 Murunga Muru era um potentado da Costa do Marfim para quem os sobas de dez tribos vizinhas pagavam tributo: cem moedas de ouro, de dez gramas cada uma, mensalmente vinham enriquecer os cofres do rei.

Um belo dia, o vizir M'bongo Inuru contou ao rei que corria um boato segundo o qual um dos "sobas" estava enganando seu senhor, pagando-lhe tributo em moedas de apenas 9 gramas cada uma.

Ocorre que Murunga Muru tinha recebido naqueles dias, dos Estados Unidos, uma moderna balança americana, sem mostrador, que registrava o peso num "ticket" após receber, no orifício adequado, uma fichinha especial.

Murunga Muru não teve dúvidas. Ao receber os dez sacos de moedas de ouro dos sobas escravizados, correu à sala da balança pretendendo pesar os sacos um a um e descobrir assim o malandro.

Qual não foi entretanto a sua surpresa ao descobrir que somente lhe restava uma fichinha, importada dos Estados Unidos, para acionar a delicada balança.

Entretanto, horas depois, Molobongo Anembu, um dos sobas, era preso pelos guardas do rei, acusado de fraudar o tributo devido ao soberano. Como fez Murunga Muru para descobrir em Molobongo Anembu, com uma só pesagem, o soba desonesto?

14.3 Mr. Thomas O'Neill, importante industrial, costumava voltar de trem, do centro de Londres para sua mansão suburbana.

Todos os dias, britânica que era, a composição ferroviária chegava exatamente às 18 horas na estação de Brianchurch.

Exatamente quando Mr. Thomas colocava seu pé direito na gare, Mr. Keith Storm, seu chofer e mordomo, encostava o Rolls-Royce cinza-claro diante da estação:
– "Good evening, Sir!"

Mr. Thomas subia e o carro seguia, placidamente, rumo à casa, enquanto o milionário lia a edição do "The Time".

Certa vez, em completo desacordo com as tradições inglesas, o trem chegou a Brianchurch uma hora mais cedo, ou seja, às 17 horas.

Embora profundamente contrariado, Mr. Thomas colocou fleumaticamente o "The Time" sob o braço e tomou, a pé, o rumo de casa. A certa altura encontrou-se com Keith que vinha buscá-lo.

"Good afternoon, Sir!"

Subiu no Rolls-Royce e nesse dia chegou em casa 20 minutos mais cedo.

Pergunta-se: Por quanto tempo Mr. Thomas O'Neill andou a pé?

# Apêndices

# 1. Resumo dos comandos e funções do Apple

**CONVENÇÕES**

| | |
|---|---|
| [ ] | Colchetes, que não devem aparecer no comando dado ao computador, indicam itens opcionais. |
| . . . | Reticências, que não devem aparecer no comando dado ao computador, indicam que o termo antecedente pode ser repetido mais de uma vez. |
| outros sinais de pontuação | Devem ser digitados como aparecem: vírgula, ponto-e-vírgula, aspas e parênteses. |
| MAIÚSCULAS | Devem ser digitadas exatamente como aparecem. |
| texto em itálico | São os termos genéricos, cujos valores devem ser especificados pelo programador. As definições destes termos são dadas abaixo. |
| const | Constante numérica ou alfanumérica. |
| d | Número de um *drive*, 1 ou 2. |
| ender | Expressão numérica cujo valor se refere a um endereço de memória válido, entre -65535 e 65535, onde -65535 é igual a 1, -65534 é igual a 2, e assim por diante. |
| expr | Expressão numérica, alfanumérica (ou *string*), relacional, Booleana, constante, variável, ou uma combinação destes. |
| expralfa | Constante, variável ou expressão alfanumérica; qualquer operação que tenha como saída um *string*. |
| exprnum | Constante, variável ou expressão numérica; qualquer operação que tenha como saída um número. |

| | |
|---|---|
| lin | Linha de programa. |
| mvar | Nome de uma variável subscrita ou matriz |
| nome | Nome de um arquivo de disco. |
| rvar | Nome de uma variável real (numérica). |
| s | Número de *slot* para entrada/saída, um inteiro entre Ø e 7. |
| sub | Índice ou conjunto de índices separados por vírgulas, que definem um elemento de uma variável subscrita ou matriz. |
| v | Número de volume que identifica um disco, um inteiro entre Ø e 255. |
| var | Variável numérica, inteira ou alfanumérica (*string*), subscrita ou não. |
| x | Expressão numérica que define uma coluna nos gráficos de baixa resolução, um inteiro entre Ø e 39. |
| xa | Expressão numérica que define uma coluna nos gráficos de alta resolução, um inteiro entre Ø e 279. |
| y | Expressão numérica que define uma linha nos gráficos de baixa resolução, um inteiro entre Ø e 47. |
| ya | Expressão numérica que define uma linha nos gráficos de alta resolução, um inteiro entre Ø e 191. |

## COMANDOS

**BLOAD** *nome* [, A*ender*] [, D*d*] [, S*s*] [, V*v*] (DOS)
Carrega, do disco no *drive d, slot s* e de *volume v*, um arquivo binário, colocando-o na posição de memória de onde foi gravado, ou na posição *ender*.

**BRUN** *nome* [, A*ender*] [, D*s*] [, S*s*] [, V*v*] (DOS)
Carrega um programa binário, como no comando **BRUN**, e executa um **JMP** ao endereço inicial do programa carregado.

**CALL** *ender*
Transfere a execução para a rotina em linguagem de máquina no endereço especificado.

**CATALOG** [, D*d*] [, S*s*] (DOS)
Mostra na tela uma lista de todos os arquivos no disco do *drive d, slot s*.

**CLEAR**
Coloca o valor zero em todas as variáveis numéricas e o *string* de valor nulo (tamanho zero) em todas as variáveis alfanuméricas, inclusive nas variáveis subscritas.

**COLOR** = *exprnum*
Inicializa a cor, especificada pela expressão numérica, a ser usada nas próximas operações com gráficos de baixa resolução.



**CONT**
Reinicia a execução após uma interrupção do programa por **STOP, END, ctrl-C** ou **RESET.**

**DATA** *const* [, *const*] . . .
Cria uma lista de constantes, que pode ser lida pelo *Applesoft* através do comando **READ.**

**DEF FN** *nome* (*var*) = *exprnum*
Permite a definição, dentro de um programa, de funções numéricas especiais.

**DEL** *lin1, lin2*
Apaga da memória as linhas de programação entre e inclusive *lin1* e *lin2.*

**DELETE** *nome* [, D*d*] [, S*s*] [, V*v*] (DOS)
Apaga, do disco no *drive d, slot s* e de *volume v,* o arquivo de nome dado.

**DIM** *var* (*sub*) [, *var* (*sub*)] . . .
Permite dimensionar uma matriz ou variável subscrita.

**DRAW** *exprnum* [**AT** *xa, ya*]
Coloca na tela de alta resolução a figura, predefinida, especificada por *exprnum,* a partir da última posição **PLOTada**, ou a partir da coordenada dada por xa e ya.

**END**
Interrompe a execução de um programa.

**FLASH**
Inicia a impressão em modo **FLASH,** fazendo com que os próximos caracteres mandados ao vídeo via **PRINT** saiam piscando.

**FOR** *rvar* = *exprnum1* **TO** *exprnum2* [**STEP** *exprnum3*]
Inicia um *loop* que incrementa de *exprnum3* (ou de 1) o conteúdo da variável real *rvar,* variando de *exprnum1* até *exprnum2.* Usado junto com **NEXT.**

**FP** [, D*d*] [, S*s*] [, V*v*] (DOS)
Coloca o computador na linguagem *Applesoft,* apagando programa e dados.

**GET** *var*
Pega do teclado um único caractere, armazenando-o na variável *var,* sem imprimi-lo na tela.

**GOSUB** *lin*
Transfere a execução do programa para a sub-rotina iniciada na linha especificada, voltando ao comando que segue o **GOSUB** quando é encontrado um **RETURN.**

**GOTO** *lin*
Transfere a execução do programa para a *lin*ha indicada.

**GR**
Coloca no vídeo a tela de baixa resolução, página um, definindo uma matriz 4Ø por 4Ø para gráficos e deixando quatro linhas na parte inferior da tela para texto.

**HCOLOR** = *exprnum*
Prepara o computador para **PLOTar** na cor indicada por *exprnum*.

**HGR**
Coloca no vídeo a tela de alta resolução, página um, definindo uma matriz 28Ø por 16Ø para gráficos e deixando quatro linhas na parte inferior da tela para texto.

**HGR2**
Coloca no vídeo a tela de alta resolução, página dois, definindo uma matriz 28Ø por 192 para gráficos.

**HIMEM:** *exprnum*
Define a mais alta posição de memória disponível para um programa BASIC e suas variáveis.

**HLIN** *x1, x2* **AT** *y*
Desenha uma linha horizontal de baixa resolução, da posição (*x1, y*) até a posição (*x2, y*).

**HOME**
Limpa a tela de texto e posiciona o cursor na posição superior esquerda do vídeo.

**HPLOT** *xa, ya*
**HPLOT** [*xa1, ya1*] **TO** *xa2, ya2*
**HPLOT** *xa1, ya1* **TO** *xa2, ya2* [**TO** *xa3, ya3*]...
Coloca um ponto (primeiro formato), traça uma linha entre dois pontos (segundo formato) ou traça um conjunto de linhas definidas por mais de dois pontos (terceiro formato) na tela de alta resolução.

**HTAB** *exprnum*
Posiciona o cursor na coluna especificada por *exprnum*, entre 1 e 4Ø.

**IF** *exprnum* **THEN** *comando* [:*comando*]...
O comando ou lista de comandos seguindo o **THEN** será executado se e somente se a expressão numérica for diferente de zero. Se *exprnum* for uma comparação entre dois elementos, isso significa que essa comparação deve ser verdadeira para que o **THEN** seja executado.

**IN#** *s*
Define o conector periférico de onde virão as próximas entradas. É considerado comando de *DOS* apenas quando este estiver na memória.

**INIT** *nome* [,**D***d*] [,**S***s*] [,**V***v*] (DOS)
Inicializa o disco do *drive d, slot s*, dando-lhe um *volume v*.

**INPUT** ["*texto*";] *var* [,*var*]...
Pega, do teclado ou do periférico de entrada ativado, uma cadeia de caracteres terminada por **RETURN**, analisando-a e atribuindo os valores por ela especificados às variáveis dadas. Antes de mostrar o cursor, o texto entre aspas ou um ponto de interrogação é colocado na tela ou transmitido ao periférico de saída ativado.

**INT** (DOS)
Coloca o *Apple* na linguagem *Integer BASIC*, apagando programa e dados.



**INVERSE**
Inicia a impressão em modo **INVERSE**, fazendo com que os próximos caracteres mandados ao vídeo via **PRINT** saiam invertidos, isto é, pontos pretos sobre um fundo branco.

[**LET**] *var* = *expr*
Coloca em *var* o resultado de *expr*.

**LIST** [*lin 1* - *lin 2*]
**LIST** *lin*
**LIST** *lin*-
**LIST** -*lin*
Mostra na tela todo ou parte do programa na memória, de *lin 1* a *lin 2* (primeiro formato), apenas *lin* (segundo formato), a partir de *lin* (terceiro formato), ou até *lin* (quarto formato). Em qualquer um dos casos, o hífen pode ser substituído por uma vírgula.

**LOAD**
Carrega um programa em *BASIC* do *cassete*, substituindo qualquer programa que possa estar na memória.

**LOAD** *nome* [,**D***d*] [,**S***s*] [,**V***v*] (DOS)
Carrega o programa em *BASIC* de nome dado do disco no *drive d, slot s* e de *volume v*, substituindo qualquer programa que possa estar na memória.

**LOCK** *nome* [,**D***d*] [,**S***s*] [,**V***v*] (DOS)
Protege o arquivo de nome dado, do disco no *drive d, slot s* e *de volume v*, de ser apagado ou alterado acidentalmente.

**LOMEM:** *exprnum*
Define a mais baixa posição de memória disponível para um programa *BASIC* e suas variáveis.

**NEW**
Retira da memória qualquer programa *BASIC* e suas variáveis.

**NEXT** [*rvar* [,*rvar*]...]
Define o final de um *loop* iniciado por um comando **FOR**, incrementando e testando a variável de controle *rvar* e voltando ao comando que segue o **FOR**, no caso de não ter sido completado o *loop*.

**NORMAL**
Desliga os modos de impressão na tela **FLASH** e **INVERSE**, voltando ao modo de impressão normal.

**NOTRACE**
Desliga o modo **TRACE**.

**ONERR GOTO** *lin*
Define uma rotina, iniciando na linha *lin*, que será executada sempre que houver qualquer erro na execução do programa.

**ON** *exprnum* **GOSUB** *lin* [,*lin*]...
Permite a chamada condicional de uma entre várias sub-rotinas. Se *exprnum* for 1, a primeira sub-rotina é executada; se for 2, a segunda; se for 3, a terceira, e assim por diante. Se *exprnum* for Ø ou maior que o número de linhas na lista, o comando que segue o **ON... GOSUB** é executado.

**ON** *exprnum* **GOTO** *lin* [, *lin*] . . .
Permite o desvio condicional para uma entre várias linhas do programa. A execução passa para a primeira linha da lista se *exprnum* for 1, para a segunda se for 2, e assim por diante. Se *exprnum* for ∅ ou maior que o número de linhas na lista, o comando que segue o **ON ... GOTO** é executado.

**PLOT** *x, y*
Coloca, no ponto definido por *x* e *y* da tela de gráficos de baixa resolução, um bloco da última cor definida por **COLOR =** .

**POKE** *ender, exprnum*
Coloca, na posição de memória definida por *ender*, o valor de *exprnum* (entre ∅ e 255).

**POP**
Faz com que o *Applesoft* "esqueça" o último **GOSUB** executado, permitindo ao programa sair de uma sub-rotina via **GOTO.**

**PR#** *s*
Define o conector periférico para onde irão as próximas saídas. É considerado comando de *DOS* apenas quando este estiver na memória.

**PRINT** [[*expr*] [;] [,] . . . ] . . .
Manda caracteres à tela ou ao periférico ativado. Um **PRINT** isolado manda o cursor para a próxima linha; um **PRINT** seguido de uma ou mais expressões imprimirá o valor dessas expressões, juntas ou organizadas em colunas, se o separador entre elas for o ponto-e-vírgula ou a vírgula, respectivamente. O ponto de interrogação pode ser utilizado como abreviação do comando **PRINT.**

**READ** *var* [, *var*] . . .
Coloca valores lidos de uma lista de **DATA,** seqüencialmente, nas variáveis especificadas no **READ.** O comando **READ,** quando utilizado no modo direto com o *DOS* ativo na memória, será interpretado como um comando *DOS,* que devolverá uma mensagem de erro.

**RECALL** *mvar*
Carrega uma matriz do *cassete*. A matriz deve ter sido dimensionada com as mesmas dimensões da original e não pode ser uma matriz alfanumérica.

**REM** *comentário*
Tudo o que segue o comando **REM** numa linha de programa será tomado como comentário e ignorado pelo *BASIC.*

**RENAME** *nome1, nome2* [, D*d*] [, S*s*] [, V*v*] (DOS)
Substitui por *nome2* o nome do arquivo *nome1,* armazenado no disco do *drive d, slot s* e de *volume v.*

**RESTORE**
Reinicializa o **DATA,** fazendo com que o próximo **READ** pegue o primeiro item da lista.

**RESUME**
Reinicia a execução na linha em que houve um erro, apenas se este foi interceptado por **ONERR GOTO.**

**RETURN**
Pula para a instrução que segue o último **GOSUB** executado.



**ROT =** *exprnum*
Define a orientação das figuras de alta resolução a serem colocadas na tela por **DRAW** ou **XDRAW,** com *exprnum* variando de ∅ (zero graus) até 64 (36∅ graus).

**RUN** [*lin*]
Executa o programa que se encontra na memória, a partir de sua primeira linha ou de *lin*, se especificada. **RUN** apaga todas as variáveis.

**RUN** *nome* [, D*d*] [, S*s*] [, V*v*] (DOS)
Carrega e executa o programa *BASIC nome* do disco do *drive d, slot s* e de *volume v.*

**SAVE**
Armazena o programa *BASIC* em *cassete.*

**SAVE** *nome* [, D*d*] [, S*s*] [, V*v*] (DOS)
Armazena o programa *BASIC* no disco do *drive d, slot s* e de *volume v,* com o *nome* dado, substituindo qualquer programa não protegido por **LOCK** que tenha o mesmo nome.

**SCALE =** *exprnum*
Define o tamanho das figuras a serem colocadas na tela por **DRAW** ou **XDRAW,** com *exprnum* variando de ∅ a 255. Na escala 1, as figuras aparecerão como definidas; na escala 2, aparecerão com o dobro do tamanho original, e assim por diante, até 255 (255 vezes) e ∅ (256 vezes o tamanho original).

**SHLOAD**
Carrega uma tabela de figuras predefinidas, a serem usadas por **DRAW** e **XDRAW,** do *cassete.*

**SPEED =** *exprnum*
Define a velocidade com que os caracteres serão impressos na tela, ou transmitidos ao periférico ativo, com *exprnum* variando de ∅ (velocidade mais lenta) até 255 (velocidade mais rápida).

**STOP**
Interrompe a execução de um programa, avisando em que linha ocorreu essa interrupção.

**STORE** *mvar*
Armazena em *cassete* o conteúdo da matriz numérica *mvar,* a ser lida, em outra ocasião, por **RECALL.**

**TEXT**
Coloca no vídeo a tela normal de texto, levando o cursor à última linha.

**TRACE**
Inicia o modo **TRACE,** em que são impressos na tela os números de linha de cada comando executado.

**UNLOCK** *nome* [, D*d*] [, S*s*] [, V*v*] (DOS)
Elimina a proteção conferida pelo comando **LOCK** ao arquivo *nome* armazenado no disco do *drive d, slot s* e de *volume v.*

**VERIFY** *nome* [, D*d*] [, S*s*] [, V*v*] (DOS)
Verifica o arquivo *nome,* no disco do *drive d, slot s* e de *volume v,* assegurando ao operador que este foi e continua armazenado corretamente. A versão *3.3* do *DOS* executa um **VERIFY** automaticamente, depois de cada **SAVE** ou **BSAVE.**

VLIN *y1*, *y2* AT *x*
Desenha uma linha vertical de baixa resolução, da posição (*x*, *y1*) até a posição (*x*, *y2*).

VTAB *exprnum*
Posiciona o cursor na linha especificada por *exprnum*, entre 1 e 24.

WAIT *ender*, *exprnum1* [, *exprnum2*]
Interrompe temporariamente a execução de um programa em *Applesoft*, até que seja diferente de zero o resultado da operação (*ender* XOR *exprnum2*) AND *exprnum1*. Se *exprnum2* não for especificada, o *BASIC* usa o valor zero em seu lugar, podendo a expressão acima ser simplificada para *ender* AND *exprnum1*. Enquanto o resultado dessa operação for zero, o computador permanece num *loop* contínuo, que só pode ser interrompido por RESET. Os valores de *exprnum1* e *exprnum2* devem estar entre ∅ e 255, enquanto *ender* deve ser uma expressão que resulte num endereço de memória permitido.

XDRAW *exprnum* [AT *xa*, *ya*]
Coloca na tela de alta resolução a figura, predefinida, especificada por *exprnum*, a partir da última posição PLOTada, ou a partir da coordenada dada por *xa* e *ya*, sendo que cada é ponto colocado na tela com a cor complementar da que lá estava. As cores complementares são ∅ e 3, 1 e 2, 4 e 7, e 5 e 6. XDRAW é utilizado, no lugar do DRAW, quando se deseja desenhar e depois apagar a mesma figura; o primeiro XDRAW a coloca na tela, enquanto o segundo a apaga.

## FUNÇÕES

ABS (*exprnum*)
Retorna o valor absoluto, ou módulo, de *exprnum*.

ASC (*expralfa*)
Retorna o código *ASCII* do primeiro caractere de *expralfa*.

ATN (*exprnum*)
Retorna o arco-tangente de *exprnum*, em radianos, entre –PI/2 e PI/2.

CHR$ (*exprnum*)
Retorna o caractere alfanumérico cujo código *ASCII* é igual a *exprnum*.

COS (*exprnum*)
Retorna o cosseno do ângulo *exprnum*, expresso em radianos.

EXP (*exprnum*)
Retorna e elevado a *exprnum*, com e = 2.71828183.

FN *nome* (*exprnum*)
Retorna o valor da função nome, previamente definida por DEF FN, aplicada a *exprnum*.

FRE (*exprnum*)
Retorna o número de *bytes* disponíveis a um programa *BASIC*, depois de compactada a área utilizada pelas variáveis alfanuméricas. O valor de *exprnum* não influi no resultado. Se o resultado for menor que ∅, some 65536 a ele.



INT (*exprnum*)
Retorna o maior inteiro menor ou igual ao valor de *exprnum*.

LEFT$ (*expralfa*, *exprnum*)
Retorna o *string* composto pelos *exprnum* primeiros caracteres de *expralfa*, sendo retornado o *string* inteiro no caso de *exprnum* exceder o número de caracteres de *expralfa*.

LEN (*expralfa*)
Retorna o tamanho, ou número de caracteres, de *expralfa*.

LOG (*exprnum*)
Retorna o logaritmo natural de *exprnum*.

MID$ (*expralfa*, *exprnum1* [, *exprnum2*])
Retorna os primeiros *exprnum2* caracteres de *expralfa*, começando do caractere de ordem *exprnum 1*. Se *exprnum2* não é especificado, a função retorna todos os caracteres de *expralfa*, a partir de *exprnum1*.

PDL (*exprnum*)
Retorna um número, entre ∅ e 255, baseado na posição do controlador de jogos (*paddle*) de número *exprnum* (*∅ a 3*).

PEEK (*ender*)
Retorna o conteúdo numérico decimal da posição de memória definida pela expressão numérica *ender*.

POS (*exprnum*)
Retorna a posição horizontal, entre ∅ e 39, do cursor. O valor de *exprnum* não influi no resultado.

RIGHT$ (*expralfa*, *exprnum*)
Retorna o *string* composto pelos *exprnum* últimos caracteres de *expralfa*, sendo retornado o *string* inteiro no caso de *exprnum* exceder o número de caracteres de *expralfa*.

RND (*exprnum*)
Retorna um número aleatório maior ou igual a ∅ e menor que 1. Se *exprnum* é positivo, a função retorna um número diferente cada vez que é usada, a não ser que tenha sido inicializada uma seqüência repetitível, conseguida quando *exprnum* é negativo; qualquer valor negativo sempre inicia a mesma seqüência de números, que pode ser continuada usando um valor positivo como argumento. Quando *exprnum* é zero, o último número gerado pela função será retornado.

SCRN (*x*, *y*)
Retorna um número, entre ∅ e 15, correspondendo à cor do ponto especificado por *x* e *y*.

SGN (*exprnum*)
Retorna 1 se *exprnum* for positivo, ∅ se for zero e –1 se for negativo.

SIN (*exprnum*)
Retorna o seno do ângulo *exprnum*, expresso em radianos.

SPC (*exprnum*)
Imprime *exprnum* espaços; é usado com PRINT.

**SQR (*exprnum*)**
Retorna a raiz quadrada positiva de *exprnum*.

**STR$ (*exprnum*)**
Retorna um *string* composto pelos dígitos e sinais que fazem parte do valor de *exprnum*.

**TAB (*exprnum*)**
Move o cursor para a coluna definida por *exprnum*, passando para outra linha se necessário, se esta estiver à direita da posição atual do cursor. A posição do cursor não é alterada se *exprnum* não define uma posição à sua direita. O valor zero posiciona o cursor na coluna 256. **TAB** deve ser usado com **PRINT.**

**TAN (*exprnum*)**
Retorna a tangente do ângulo *exprnum*, expresso em radianos.

**USR (*exprnum*)**
Retorna o valor presente no acumulador, depois de executada a rotina em linguagem de máquina iniciada na posição 1Ø ($ØA). As posições de memória 1Ø a 12 ($ØA a $ØC) devem conter uma instrução **JMP** para o endereço inicial da sub-rotina.

**VAL (*expralfa*)**
Converte *expralfa* num valor numérico, retornando o número representado pelo *string expralfa*. Se o primeiro caractere de *expralfa* não for numérico, o valor zero é retornado. O *string expralfa* é lido e convertido apenas até chegar ao primeiro caractere não numérico.

# 2. Glossário

**65Ø2:** O microprocessador usado no Apple e em vários outros computadores.

**Applesoft:** Uma versão estendida da linguagem de programação BASIC, usada com o computador Apple ][ e capaz de processar números na forma de ponto flutuante. Compare com **Integer BASIC.**

**arquivo:** Uma coleção de informações gravadas como uma unidade, com um nome, num meio de armazenamento periférico, como uma unidade de disco.

**arquivo binário:** Um arquivo contendo as informações "cruas", não expressas em forma de texto. Compare com **arquivo-texto.**

**arquivo, nome do:** O nome sob o qual um arquivo é armazenado.

**arquivo-texto:** Um arquivo contendo informações expressas em forma de texto. Compare com **arquivo binário.**

**assembler** (montador): Um tradutor de linguagem que converte um programa escrito na linguagem *assembler* em um programa equivalente em linguagem de máquina.

**base de dados:** Uma coleção de informações organizadas numa forma que pode ser processada pelo sistema de computador.

**base de dados, sistema de manipulação de** (DBMS): Um sistema de *software* que organiza, armazena, lê e modifica informações numa base de dados.

BASIC: *Beginner's All-purpose Symbolic Instruction Code*; uma linguagem de programação de alto nível feita para ser fácil de aprender e usar. No Apple ][ há duas versões do BASIC: Applesoft e Integer BASIC.

**binário:** A representação de números em termos de potências de dois, usando os dígitos Ø e 1. Normalmente usado em computadores, já que os valores Ø e 1 podem ser facilmente representados em forma física numa variedade de modos: presença ou ausência de corrente, tensão positiva ou negativa, ou um ponto preto ou branco numa tela de vídeo.

**bit:** Um dígito binário (∅ ou 1); a unidade de informação mais simples possível, consistindo em uma escolha simples entre duas alternativas, como sim ou não, ligado ou desligado, positivo ou negativo, alguma coisa ou nada.

**boot:** A ação de inicializar o computador carregando um programa na memória a partir de um meio de armazenamento externo, como uma unidade de disco.

**bootstrap:** ver **boot.**

**buffer:** Uma área da memória do computador reservada para uma finalidade específica, como, por exemplo, para armazenar informações gráficas a serem mostradas numa tela, ou caracteres de texto sendo lidos de algum periférico. Normalmente usado como uma área de armazenamento intermediário para a transferência de informações entre aparelhos operando em velocidades diferentes, tal como o processador do computador e uma impressora ou disco. A informação pode ser gravada no *buffer* por um dos aparelhos, e lido pelo outro a uma velocidade diferente.

**bug:** Um erro num programa que faz com que este não funcione como era esperado.

**byte:** Uma unidade de informação consistindo em um número fixo de *bits;* no Apple ][, um *byte* é composto por oito *bits* e pode armazenar um número entre ∅ e 255.

**CAI:** ver **instrução assistida por computador.**

**caractere:** Uma letra, dígito, sinal de pontuação ou outro símbolo gráfico usado na apresentação de informações numa forma compreensível por seres humanos.

**carregar:** Transferir informações de algum meio de armazenamento periférico para a memória principal do computador para o seu uso.

**catálogo:** Uma lista de todos os arquivos armazenados num disco.

**chip:** O pequeno pedaço de material semicondutor (como silício) sobre o qual um circuito integrado é fabricado. A palavra "chip" se refere ao próprio pedaço de silício, mas é muito usado também como sendo o circuito integrado e o seu invólucro. Ver **circuito integrado.**

**CI:** ver **circuito integrado.**

**circuito impresso, placa de:** Um componente do *hardware* do computador ou de outro aparelho eletrônico, consistindo em uma placa retangular de algum material, como fibra de vidro, à qual circuitos integrados e outros componentes são ligados.

**circuito integrado:** Um componente eletrônico consistindo em muitos elementos de um circuito fabricados sobre um único pedaço de material semicondutor, como o silício. Ver **chip.**

**codificação:** As instruções que compõem um programa.

**código:** Um número ou símbolo usado para representar alguma informação numa forma compacta ou facilmente processável.

**cold start** (partida inicial): O processo de inicialização de um computador assim que ele é ligado (ou como se ele tivesse sido ligado de novo), carregando o sistema operacional na memória. Compare com **warm start.**



**comando:** Uma comunicação do usuário para o sistema (normalmente através do teclado), fazendo com que este execute algo imediatamente.

**compilador:** Um tradutor de linguagem que converte um programa escrito numa linguagem de alto nível em seu equivalente numa linguagem de baixo nível (como linguagem de máquina) para a sua execução posterior. Compare com **interpretador.**

**computador:** Um aparelho eletrônico capaz de executar operações com números a uma alta velocidade e com grande precisão.

**conector:** Um elemento físico, como uma tomada ou um soquete, usado na conecção de um componente de *hardware* com outro.

**controlador de disco:** Um cartão periférico que controla a operação, no Apple, de até dois *drives* de disco.

**CPU:** ver **UCP.**

**cursor:** Uma marca ou símbolo mostrado na tela que indica o lugar em que a próxima ação do usuário terá efeito, ou onde o próximo caractere digitado no teclado aparecerá.

**dados:** Informações, especialmente aquelas usadas ou manipuladas por um programa.

**demodular:** Recuperar as informações sendo transmitidas através de um sinal modulado; por exemplo, um receptor de rádio convencional demodula o sinal recebido pela antena para convertê-lo no som emitido pelo alto-falante.

**dígito:** (1) Um dos caracteres ∅ a 9, usados para representar números na forma decimal. (2) Um dos caracteres usados para representar números em qualquer outra forma, tal como ∅ e 1 em binário ou ∅ a 9 e A a F em hexadecimal.

**diretório:** ver **catálogo.**

**disco:** Um meio de armazenamento de informações consistindo em uma superfície magnética, circular e lisa, na qual informações podem ser registradas na forma de minúsculos pontos magnetizados, de maneira semelhante às fitas de áudio.

**disco de inicialização** (*startup disk*): Um disco contendo *software* gravado na forma correta para ser carregado na memória do computador e preparar o sistema para operar. Também chamado "disco de boot"; ver **boot.**

**disco flexível:** Um disco feito de plástico flexível; também chamado *floppy disk.*

**diskette:** Um termo usado para os discos flexíveis, de 5 1/4 polegadas de diâmetro, usados no Apple e na maioria dos computadores pessoais atuais.

**display:** (1) Informações apresentadas visualmente, especialmente numa tela de vídeo. (2) Um aparelho que apresenta informações visualmente, como um receptor de televisão ou um monitor de vídeo.

**DOS:** ver **Sistema Operacional de Disco.**

**drive** (de disco): Um aparelho periférico que grava e lê informações na superfície de um disco magnético.

**editar:** Alterar ou modificar; por exemplo, inserir, retirar, substituir ou mover texto num documento.

**endereço** (*address*): Um número usado para identificar alguma coisa, como uma posição na memória do computador.

**entrada:** Informação transferida para o computador a partir de uma fonte externa, como um teclado, *drive* ou *modem*.

**entrada/saída:** ver **I/O.**

**erro, mensagem de:** Uma mensagem apresentada para avisar ao usuário de algum erro ou problema na execução de um programa.

**escape, seqüência de:** No Apple ][, uma seqüência de teclas, começando pela tecla ESC, usadas para posicionamento do cursor na tela.

**executar:** Realizar uma ação ou seqüência de ações, tal como aquelas descritas por um programa.

**firmware:** Aqueles componentes de um sistema de computador consistindo em programas gravados permanentemente em ROM. Tais programas são instalados no computador na fábrica; eles podem ser executados a qualquer hora, mas não podem ser modificados ou apagados da memória principal. Compare com **hardware** e **software.**

**floppy disk:** ver **disco flexível.**

**fonte de alimentação:** O componente do *hardware* do computador que transforma os 11Ø ou 22Ø V de uma tomada comum nas formas exigidas pelos outros componentes do *hardware*.

**FORTRAN:** Uma linguagem de programação de alto nível muito usada, orientada para aplicações que requerem uso extensivo de cálculos numéricos, como em matemática, engenharia e ciências.

**gráficos:** Informação apresentada pelo computador na forma de desenhos ou imagens.

**gráficos de alta resolução:** No Apple ][, o *display* de gráficos com uma resolução de 28Ø x 192 pontos, em seis cores diferentes.

**gráficos de baixa resolução:** No Apple ][, o *display* de gráficos com uma resolução de 48 x 4Ø blocos em 16 cores diferentes.

**gravação:** A transferência de informações do computador para um destino externo ao computador (como uma unidade de disco, *modem* ou impressora) ou do processador para um elemento externo ao processador (como a memória principal).

**hardware:** Aqueles componentes de um sistema de computador consistindo em elementos físicos (eletrônicos ou mecânicos). Compare com **software** e **firmware.**



**hexadecimal:** A representação de números em termos de potências de 16, usando os 16 dígitos Ø a 9 e A a F. Números no sistema hexadecimal são mais fáceis para os seres humanos de ler e entender que os números no sistema binário, mas podem ser convertidos diretamente à forma binária. Cada dígito hexadecimal corresponde a uma seqüência de quatro dígitos binários (*bits*).

**IC:** ver **circuito integrado.**

**impressora de impacto:** Uma impressora que forma os caracteres batendo mecanicamente numa fita entintada contra o papel.

**impressora de matriz de pontos** (*dot-matrix*): Uma impressora que representa seus caracteres como um conjunto de pontos numa matriz de tamanho fixo.

**impressora "margarida"** (*daisy-wheel*): Uma impressora de impacto que produz os caracteres batendo numa roda com letras e símbolos em relevo contra uma fita entintada, formando no papel um caractere completo por batida.

**impressora paralela:** Uma impressora que aceita informações do computador por meio de um *interface* paralelo.

**impressora para processamento de texto:** Uma impressora que produz resultados comparáveis em qualidade àqueles produzidos por uma máquina de escrever elétrica.

**impressora serial:** Uma impressora que aceita informações do computador por meio de um *interface* serial.

**impressora térmica:** Uma impressora que imprime através da aplicação de pequenos pontos de calor num papel sensível ao calor.

**inicializar:** (1) Colocar num estado inicial, em preparação para alguma computação. (2) Preparar um disco virgem para receber informações, dividindo a sua superfície em *tracks* e setores; também, formatar.

**instrução:** Uma unidade de um programa, correspondendo a uma ação ou conjunto de ações que o computador deverá fazer.

**instrução assistida por computador** (CAI): O uso de um computador para ensinar alguma matéria a um usuário humano, de forma interativa.

**Integer BASIC:** Uma versão da linguagem de programação BASIC usada no Apple ][, mais velha que o Applesoft e capaz de processar números apenas na forma de inteiros. Compare com **Applesoft.**

**interativo:** Operando por meio de um diálogo entre o computador e o seu usuário humano.

**interface:** Os elementos, regras ou convenções pelos quais um componente de um sistema se comunica com outro.

**interface** (com o usuário): As regras e convenções que regem a comunicação do sistema de computador com o usuário.

**interface paralelo:** Um *interface* em que vários *bits* de informação (tipicamente um *byte* de oito *bits*) são transmitidos simultaneamente através de fios ou canais separados. Compare com **interface serial.**

**interface serial:** Um *interface* em que informações são transmitidas seqüencialmente, *bit* por *bit,* através de um único fio ou canal. Compare com **interface paralelo.**

**interpretador:** Um tradutor de linguagem que lê um programa escrito numa linguagem de programação e imediatamente executa as ações que o programa descreve. Compare com **compilador.**

**I/O:** *Input/Output* (entrada/saída); a transferência de informações de e para o computador; ver **entrada** e **saída.**

**janela de texto:** Uma área do *display* do Apple, dentro da qual pode ser apresentado texto.

**leitura:** A transferência de informações à memória do computador, a partir de uma fonte externa ao computador (como uma unidade de disco ou *modem*), ou ao processador, a partir de uma fonte externa a este (como a memória ou o teclado).

**linguagem de alto nível:** Uma linguagem de programação em que programas são expressos numa forma relativamente fácil para a sua compreensão por seres humanos. Uma única instrução de uma linguagem deste tipo tipicamente corresponde a várias instruções de linguagem de máquina – às vezes dezenas ou até centenas destas. Entre as linguagens de alto nível estão BASIC, FORTRAN, Pascal, COBOL, Pilot e Logo.

**linguagem de assembler** (linguagem de montagem): Uma linguagem de programação de baixo nível em que instruções de máquina individuais são escritas numa forma simbólica, compreensível pelo programador humano com maior facilidade que a própria linguagem de máquina.

**linguagem de baixo nível:** Uma linguagem de programação em que programas são expressos numa forma relativamente próxima àquela que pode ser executada diretamente pelo processador do computador. Entre as linguagens de baixo nível estão a linguagem de máquina e a linguagem de *assembler* (montagem) para o 65Ø2.

**linguagem de máquina:** A forma em que instruções ao computador são armazenadas na memória para execução direta pelo processador do computador. Cada modelo de processador (como o 65Ø2 no Apple ][) tem a sua própria forma de linguagem de máquina que ele executa diretamente.

**linguagem de programação:** Um conjunto de regras ou convenções para escrever programas.

**Logo:** Uma linguagem de programação desenvolvida recentemente e desenhada para ensinar programação a crianças de qualquer idade, fazendo uso extensivo dos recursos gráficos do computador.

**memória, posição de:** Uma unidade da memória principal que é identificada por um endereço e pode armazenar um único item de informação de um tamanho fixo: no Apple ][, cada posição guarda um *byte,* de oito *bits,* de informação.

**memória principal:** O componente de memória de um sistema de computador que está contido dentro do próprio computador e cujo conteúdo é diretamente acessível pelo processador.



**menu:** Uma lista de alternativas apresentada por um programa na tela do computador, entre as quais o usuário pode escolher uma ou (às vezes) mais.

**microcomputador:** Um computador cujo processador é um microprocessador.

**microprocessador:** Um processador contido num único CI, como é o caso do 65Ø2, no Apple ][.

**microssegundo:** Um milionésimo de segundo.

**milissegundo:** Um milésimo de segundo.

**modem:** Modulador/demodulador; um aparelho periférico que permite ao computador transmitir e receber informações através de fios telefônicos.

**modulador de RF:** Um aparelho que converte os sinais de vídeo produzidos pelo computador na forma que pode ser aceita por um receptor de televisão.

**modular:** Modificar ou alterar um sinal a fim de transmitir informação; por exemplo, as emissoras de rádio transmitem som modulando a amplitude (em AM) ou a freqüência (em FM) de um sinal portador.

**monitor de vídeo:** Um aparelho de *display* capaz de receber sinais de vídeo por conecção direta, que não pode receber sinais de televisão transmitidos normalmente, mas que pode ser ligado diretamente ao Apple. Compare com **receptor de televisão.**

**nanossegundo:** Um bilionésimo de segundo (ou, no uso britânico, um milésimo-milionésimo de segundo).

**Pascal:** Uma linguagem de programação de alto nível desenvolvida a fim de ensinar programação como uma disciplina de resolução de problemas.

**periférico:** Um elemento externo ao computador, fisicamente (como um aparelho periférico – monitor de vídeo, *drives,* impressora etc. – ligado ao computador por fios) ou num sentido lógico (como um cartão periférico, um cartão de circuito impresso removível, inserido num conector no computador).

**PILOT:** *Programmed Inquiry, Learning or Teaching;* uma linguagem de programação de alto nível feita para auxiliar professores a criar aulas CAI que incluem gráficos coloridos, efeitos sonoros, texto da aula e verificação de respostas do aluno.

**processador:** O componente de *hardware* de um computador que faz a própria computação, executando diretamente as instruções representadas em linguagem de máquina e armazenadas na memória principal.

**processador de texto** (ou de palavras): Um programa aplicativo para criar e modificar textos.

**programa:** Um conjunto de instruções descrevendo ações a serem executadas pelo computador para fazer alguma tarefa, conforme as regras e convenções da linguagem de programação usada.

**programa aplicativo:** Um programa que usa os recursos do computador para um propósito específico, tal como processamento de texto, manipulação de dados, gráficos ou telecomunicações. Compare com **programa de sistema.**

**programação:** O ato de escrever programas.

**programa de sistema:** Um programa que torna acessíveis os recursos de um computador, para fins gerais, como num sistema operacional ou um tradutor de linguagem. Compare com **programa aplicativo.**

**programador:** Aquele que escreve programas.

**proteger:** (1) Impedir que qualquer informação possa ser gravada num disco, cobrindo a pequena ranhura existente na sua lateral, evitando assim a perda ou alteração dos dados gravados. (2) Evitar a cópia de informações gravadas num meio de armazenamento, como um disco contendo *software* vendido como um produto comercial.

**RAM:** *Random-Access Memory*; memória em que o conteúdo de posições individuais pode ser lido ou alterado numa ordem qualquer, arbitrária, aleatória. Informações gravadas em RAM são perdidas ao se desligar o computador.

**receptor de televisão:** Um aparelho de *display* capaz de receber sinais de vídeo (como da televisão comercial) por meio de uma antena. Pode ser utilizado, junto com um modulador de RF, como um aparelho de *display* para o Apple. Compare com **monitor de vídeo.**

**rede** (de computadores): Um conjunto de computadores interligados, controlados individualmente, junto com o *hardware* e *software* usados na sua conecção.

**rodar:** Executar (um programa).

**ROM:** Memória de acesso aleatório (como a RAM), cujo conteúdo pode ser lido mas não modificado; usado para armazenar *firmware*. As informações são gravadas no ROM na fábrica e se mantêm mesmo com o computador desligado.

**rotina:** Uma parte de um programa que executa uma tarefa subordinada à tarefa principal do programa.

**saída:** Informações transferidas do computador para algum elemento externo, como uma tela de vídeo, um *drive* de disco, uma impressora ou um *modem*.

**setor:** Uma parte da superfície de um disco que consiste em uma fração fixa de um *track*. No DOS atual do Apple há 16 setores por *track*. Ver **track.**

**sistema:** Uma coleção coordenada de partes interativas e inter-relacionadas organizadas para executar alguma função ou atingir algum objetivo ou finalidade.

**sistema de computador:** Um computador e o *hardware*, *firmware* e *software* associados a ele.

**sistema operacional:** Um sistema de *software* que organiza os recursos do computador e torna-os acessíveis ao usuário ou a programas aplicativos sendo executados no computador.

**Sistema Operacional de Disco** (DOS): Um sistema de *software* que permite ao computador comunicar-se e controlar um ou mais *drives*.



**software:** Aqueles componentes de um sistema de computador consistindo em programas que determinam ou controlam o comportamento do computador. Compare com **hardware** e **firmware.**

**software aplicativo:** O componente de um sistema de computador que consiste em programas aplicativos.

**software de sistema:** O componente de um sistema de computador consistindo em programas de sistema.

**tela:** ver **display** (2).

**telecomunicações:** A transmissão de informações por longas distâncias, como através de fios telefônicos.

**texto:** Informação apresentada na forma de caracteres legíveis por seres humanos.

**track:** Uma faixa circular num disco magnético, onde as informações podem ser armazenadas. No *drive do Apple*, e com o DOS 3.3, há 35 *tracks*. Ver **setor.**

**tradutor de linguagem:** Um programa de sistema que lê um programa e o executa diretamente, ou então o converte para outra linguagem para a sua execução posterior.

**trancar** (*lock*): Proteger um arquivo de disco contra alterações e evitar que este seja apagado acidentalmente.

**UCP:** Unidade Central de Processamento; ver **processador.**

**Unidade Central de Processamento:** ver **processador.**

**usuário:** A pessoa que opera ou controla o sistema.

**vídeo inverso:** A apresentação de texto na tela do computador na forma de pontos pretos sobre um fundo branco (ou de alguma outra cor), ao invés de pontos brancos sobre um fundo preto, como é mais comum.

**warm start:** O processo de reinicialização do Apple, já estando ligado, sem que o sistema operacional seja recarregado e geralmente sem perder as informações na memória. Compare com **cold start.**

# 3. DOS 3.3/3.2 - operações com disco

## I. INTRODUÇÃO

Este apêndice serve de introdução e referência aos comandos de disco relacionados ao armazenamento e recuperação de programas, utilizando o *Sistema Operacional de Disco – DOS.*

Este sistema consiste numa série de programas de controle, unidos ao *BASIC*, cada vez que o computador é ligado com um disco no *drive.* Sua função é justamente controlar todas as operações com o disco, permitindo que o programador não se preocupe com detalhes como acionamento do motor, posicionamento da cabeça de leitura/gravação etc., e sim, apenas com a tarefa principal a ser executada.

## II. INICIALIZANDO DISCOS

Tendo sido o computador ligado com um disco no *drive* (conforme o parágrafo IV do capítulo 1 e o parágrafo IV do capítulo 3), podemos prosseguir com a inicialização de um disco:

1. Digite um programa introdutório qualquer, que será colocado no disco durante a inicialização e executado cada vez que o computador é ligado com esse disco. Se você quiser um exemplo, veja a página 72.

2. Coloque no *drive* o disco a ser inicializadq. Todas as informações que possam estar nesse disco serão apagadas durante o processo. Não se esqueça de fechar a porta da unidade de disco.

3. Digite:

**INIT HELLO**

O motor do *drive* entrará em funcionamento por cerca de um minuto. Assim que parar, o disco estará inicializado e pronto para ser usado. O programa que estava na memória do computador foi armazenado no disco sob nome *HELLO.* Se quiser, você pode usar qualquer outro nome válido (veja adiante) na inicialização.

4. O disco que acabou de ser inicializado pode ser usado para ligar o computador. Ele não serve, porém, para computadores que tenham memória menor que aquela na qual foi inicializado,



e não utiliza eficientemente a memória de computadores de maior capacidade. Para corrigir esse problema, deve existir no disco que veio com o seu computador um programa denominado *"MASTER CRIATE".* Digite:

**BRUN MASTER CRIATE**

e responda com *HELLO* à pergunta do computador. Seguindo as instruções no vídeo, você terá ao final do processo um disco com o *DOS* realocável, que pode ser usado em qualquer sistema com mais de **16 kbytes** de memória.

## III. ARMAZENANDO E LENDO PROGRAMAS

Um programa em *BASIC* que está na memória pode ser guardado permanentemente num disco. O comando **SAVE** do *DOS* permite ao operador fazê-lo. Cada programa armazenado num disco é identificado por um nome.

Se você tiver um programa qualquer na memória do computador, digite o seguinte comando para gravá-lo com o nome *TESTE:*

**SAVE TESTE**

O disco no *drive* começa a girar e após alguns segundos, de acordo com o tamanho do programa, o cursor voltará à tela, indicando o final de gravação. Para termos certeza de que o programa foi gravado, digite o comando *DOS:*

**CATALOG**

Se você estiver usando um disco recém-inicializado, a tela deverá mostrar algo semelhante a:

**DISK VOLUME 254**
**A 002 HELLO**
**A 003 TESTE**

A letra **A,** no início de cada, indica que o arquivo gravado é um programa em *Applesoft.* O número que a segue mostra o tamanho de cada programa, em blocos de **256 bytes,** denominados setores.

Mais tarde, o programa *TESTE* pode ser executado através do comando:

**RUN TESTE**

Este comando carrega diretamente do disco, para a memória do computador, o programa *TESTE* e inicia a sua execução. Se por algum motivo o programa cujo nome é dado no comando não existir no disco, a mensagem:

**FILE NOT FOUND**

aparecerá no vídeo. Lembre-se que o nome dado no **RUN** deve corresponder exatamente àquele com que o programa foi gravado.

Muitas vezes, porém, não desejamos rodar o programa e sim apenas carregá-lo do disco. O programa *TESTE* seria lido do disco para a memória principal pelo comando:

**LOAD TESTE**

O programa pode ser **LIST**ado e/ou modificado e, se necessário, gravado com um novo nome, ou sobre o antigo, usando o mesmo nome.

## IV. O NOME DO ARQUIVO

Como vimos, cada programa no disco deve ter, associado a ele, um nome que o identifique. Este nome pode ser escolhido pelo operador, desde que obedecendo aos seguintes critérios:

1. O nome deve começar por uma letra; na verdade o primeiro caractere do nome deve ter um código **ASCII** maior ou igual a 65 (a letra A). São válidos portanto os caracteres ^ e ].

2. O nome pode conter qualquer caractere exceto a vírgula ",".

3. O tamanho de um nome não deve exceder 30 caracteres; além do 30º, os caracteres serão ignorados.

## V. COMANDOS DE MANUTENÇÃO

Além do comando **CATALOG**, que produz uma lista de todos os arquivos armazenados num disco, o *DOS* tem mais alguns recursos para facilitar a manutenção de arquivos.

### V.1 – Trocando Nomes

O programa *TESTE* pode ter seu nome mudado para, por exemplo, *NOME NOVO*. A maneira mais simples e eficiente de fazê-lo é através do comando **RENAME**. Digite:

**RENAME TESTE, NOME NOVO**

O arquivo não é modificado, com exceção do seu nome, que é trocado pelo nome que vem depois da vírgula. Devemos ter o cuidado de não usar um nome que já existe no disco, pois nenhuma comparação interna é feita.

### V.2 – Apagando Arquivos

Um programa qualquer que já não seja de nenhuma utilidade pode ser retirado do disco com o comando **DELETE**. Um arquivo de nome *FOLHA* seria apagado digitando

**DELETE FOLHA**

O arquivo deletado não pode ser mais recuperado por qualquer comando de *DOS*.

### V.3 – Protegendo Contra Acidentes

Não são raros os acidentes com arquivos armazenados em disco e, na maioria das vezes, estes resultam na perda completa de um programa gravado. É bem provável que o acidente mais comum seja gravar um programa com o nome de outro já existente no disco, destruindo-o.

Estes desastres podem ser evitados colocando nos arquivos mais importantes uma proteção individual, que é feita pelo comando **LOCK**. Digite:

**LOCK TESTE**



O arquivo *TESTE*, se este existir, receberá um indicador de proteção, visível no CATALOGo como um "*". Um arquivo protegido desta forma não poderá ser alterado por qualquer comando *DOS*, a não ser que ele seja desprotegido, com o comando **UNLOCK**. A linha

**UNLOCK TESTE**

leva o arquivo *TESTE* ao seu estado desprotegido.

### V.4 – Verificando Arquivos

Ocasionalmente pode acontecer de um programa ser armazenado incorretamente no disco ou, com o tempo, ser estragado, geralmente por uma falha na superfície do disco ou simplesmente por falha humana. O comando **VERIFY** permite ao operador verificar o estado de um arquivo de qualquer tipo.

**VERIFY FOLHA**

retornará o cursor sem nenhuma mensagem se *FOLHA* estiver gravado no formato correto. No caso de haver algum erro, aparecerá a mensagem:

**I/O ERROR**

no vídeo.

## VI. MENSAGENS DE ERRO

As mensagens de erro de maior importância estão aqui relacionadas. As demais podem ser encontradas no *apêndice 6 ou 8*, junto com uma breve explicação de seu significado.

**FILE NOT FOUND:** A mensagem aparece quando um arquivo especificado por um comando *DOS* não existe no disco.

**SYNTAX ERROR:** Indica que há algum erro de sintaxe de um comando *DOS*, como um nome de arquivo ilegal ou um parâmetro inexistente ou com valor fora dos limites. Note a ausência da interrogação.

**FILE LOCKED:** Foi feita uma tentativa de alteração de um arquivo protegido por **LOCK**, através de comandos como **SAVE**, **RENAME** ou **DELETE**.

**I/O ERROR:** Pode ocorrer por vários motivos, sempre que o *DOS* não for capaz de ler ou gravar informações no disco.

**DISK FULL:** A capacidade do disco foi excedida, podendo ser em número de setores (normalmente 496) ou em arquivos (tipicamente 105).

**WRITE PROTECTED:** Nenhuma informação pode ser gravada num disco cuja abertura lateral esteja coberta.

## VII. USANDO MAIS DE UM DRIVE

A maior parte dos comandos *DOS* (todos aqueles vistos neste apêndice) aceita dois parâmetros, opcionais, que servem para controlar qual o *drive* que será acionado. O primeiro deles é o *parâmetro D*, que vem depois do nome do arquivo (menos no **CATALOGo**, que não tem nome), separado deste por uma vírgula, e pode ser *D1* ou *D2*. Assim:

**SAVE TESTE, D2**

armazenará o programa, com o nome *TESTE*, no *drive 2*. Todas as operações seguintes serão feitas sobre este disco. O *parâmetro D1* em outro comando *DOS* retornaria ao *drive 1*.

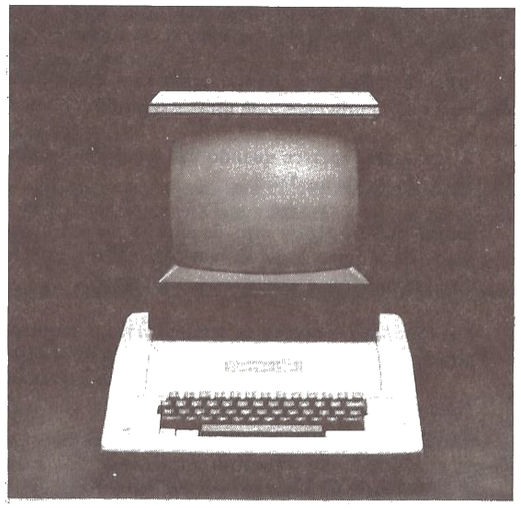
O segundo *parâmetro, S,* serve apenas para aqueles que têm três ou mais unidades de disco. Ele indica ao *DOS* o *slot,* ou conector periférico, onde está ligado o *drive* que desejamos usar. Ele, como o *parâmetro D,* também se mantém efetivado até ser modificado por outro comando que especifique um novo *slot*. Considerando uma quarta unidade de disco ligada no segundo conector do cartão no *slot 5*, o comando:

**CATALOG, S5, D2**

mostrará a lista de arquivos no disco desse *drive*. Digitando, a seguir,

**CATALOG, D1**

produzirá na tela o **CATALOGo** do disco no *slot 5* e no *drive 1*.



$\mu$ 6502 – Cortesia de Appletrônica – Computadores e Sistemas

# 4. PEEKs, POKEs e CALLs

**PÁGINA ZERO**

| | |
|---|---|
| 32 | Margem esquerda da janela de texto (Ø-39) |
| 33 | Largura da janela de texto (1-4Ø) |
| 34 | Margem superior da janela de texto (Ø-22) |
| 35 | Margem inferior da janela de texto (1-24) |
| 36 | Posição horizontal do cursor (Ø-39) |
| 37 | Posição vertical do cursor (Ø-23) |
| 43 | *Slot de boot* * 16 (depois do *boot*) |
| 44 | Ponto final do último HLIN, VLIN ou PLOT |
| 48 | Valor do COLOR * 17 |
| 5Ø | Formato de saída de texto |
| 51 | Caráter da linguagem |
| 78 - 79 | Campo de número aleatório |
| 1Ø3 - 1Ø4 | Início do programa |
| 1Ø5 - 1Ø6 | Início das variáveis |
| 1Ø7 - 1Ø8 | Início das matrizes |
| 1Ø9 - 11Ø | Fim das variáveis numéricas |
| 111 - 112 | Início dos *strings* |
| 115 - 116 | HIMEM: |
| 117 - 118 | Linha sendo executada |
| 119 - 12Ø | Linha em que a execução parou |
| 121 - 122 | Endereço da linha em execução |
| 123 - 124 | Linha dos DATA sendo lidos |
| 125 - 126 | Endereço dos DATA sendo lidos |
| 127 - 128 | Endereço do INPUT ou DATA |
| 129 - 13Ø | Nome da última variável usada |
| 131 - 132 | Endereço da última variável usada |
| 175 - 176 | Endereço do fim do programa |
| 2Ø2 - 2Ø3 | Endereço do início do programa |
| 2Ø4 - 2Ø5 | Fim do espaço das variáveis |
| 214 | Flag de RUN (RUN automático se > 127) |
| 216 | Flag de ONERR |
| 218 - 219 | Linha em que houve o erro |

| | |
|---|---|
| 222 | Código de erro do ONERR |
| 224 - 225 | Coordenada X do último HPLOT |
| 226 | Coordenada Y do último HPLOT |
| 228 | Valor do HCOLOR |
| 230 | Página de alta resolução a ser acessada (32 = página 1, 64 = página 2) |
| 231 | Valor do SCALE |
| 232 - 233 | Início do *shape table* |
| 234 | Contador de colisões de alta resolução |
| 241 | 256 – valor do SPEED |
| 243 | Máscara do FLASH |
| 249 | Valor do ROT |

**DOS** (Obs.: endereços de DOS > 40000 se referem a sistemas de 48K; subtraia 16384 para 32K.)

| | |
|---|---|
| 976 - 978 | Vetor de reentrada no DOS |
| 1010 - 1012 | Vetor de RESET |
| 1013 - 1015 | Vetor de & |
| 1016 - 1018 | Vetor de ctrl-Y |
| 42350 | Início da rotina de CATALOG |
| 43140 - 43271 | Tabela de comandos |
| 43378 - 43582 | Tabela de mensagens de erro |
| 43607 | Valor de MAXFILES |
| 43616 - 43617 | Tamanho do último BLOAD |
| 43624 | Número do *drive* |
| 43626 | Número do *slot* |
| 43634 - 43635 | Endereço do último BLOAD |
| 43697 | *Default* do MAXFILES |
| 43698 | Caractere de comando do DOS |
| 43702 | Flag do BASIC ativo |
| 44033 | Track do catálogo |
| 44567 | No. de caracteres - 1 dos nomes no catálogo |
| 44611 | No. de dígitos - 1 dos números do setor e volume |
| 45991 - 49998 | Tabela dos códigos de tipo de arquivo |
| 45999 - 46010 | Cabeçalho do catálogo |
| 46017 | Volume do disco |
| 46064 | Número de setores por *track* |

**SUB-ROTINAS**

| | |
|---|---|
| 62420 (-3116) | HGR2 |
| 62430 (-3106) | HGR |
| 62450 (-3086) | Limpar a tela de alta resolução |
| 62454 (-3082) | Pintar a tela com a última cor HPLOTada |
| 63538 (-1998) | Limpar a tela inteira de baixa resolução |
| 63542 (-1994) | Limpar a tela superior de baixa resolução |
| 63583 (-1953) | Somar 3 a COLOR |
| 63809 (-1727) | Imprimir conteúdo dos registros X e Y |
| 64215 (-1321) | Mostrar todos os registros |
| 64303 (-1233) | TEXT |



| | |
|---|---|
| 64320 (-1216) | GR |
| 64331 (-1205) | Colocar a janela de texto normal |
| 64352 (-1184) | Imprimir "APPLE ][" |
| 64484 (-1052) | Tocar um "bip" |
| 64500 (-1036) | Mover o cursor para a direita |
| 64528 (-1008) | Mover o cursor para a esquerda |
| 64538 (-998) | Mover o cursor para cima |
| 64578 (-958) | Limpar a tela a partir do cursor |
| 64600 (-936) | Limpar a tela (HOME) |
| 64614 (-922) | Mover o cursor para baixo |
| 64624 (-912) | Levantar o texto uma linha |
| 64661 (-875) | Limpar a linha do cursor |
| 64668 (-868) | Limpar a linha a partir do cursor |
| 64780 (-756) | Esperar por uma tecla |
| 64858 (-678) | Esperar por um RETURN |
| 64860 (-676) | Tocar "bip" e esperar por um RETURN |
| 64879 (-657) | INPUT (aceita vírgulas e dois pontos) |
| 64986 (-550) | Imprimir valor do registro A |
| 65152 (-384) | INVERSE |
| 65156 (-380) | NORMAL |
| 65325 (-211) | Imprimir "ERR" e tocar "bip" |
| 65338 (-198) | Tocar "bip" |
| 65369 (-167) | Voltar ao monitor no modo de texto |
| 65381 (-155) | Entrar no monitor, com "bip" |
| 65385 (-151) | Entrar no monitor, sem "bip" |
| 65392 (-144) | Ler *buffer* de entrada |

**COMUTADORES NO MODO DE VÍDEO**

| | |
|---|---|
| 49232 (-16304) | Gráficos |
| 49233 (-16303) | Texto |
| 49234 (-16302) | Página inteira |
| 49235 (-16301) | Texto com gráficos |
| 49236 (-16300) | Página 1 |
| 49237 (-16299) | Página 2 |
| 49238 (-16298) | Baixa resolução |
| 49239 (-16297) | Alta resolução |

**TECLADO, ALTO-FALANTE E CONTROLADORES DE JOGOS**

| | |
|---|---|
| 49152 (-16384) | Ler teclado |
| 49168 (-16368) | Limpar *strobe* do teclado |
| 49200 (-16336) | Acionar o alto-falante |

(Obs.: Se as posições abaixo forem > 127, o botão está acionado)

| | |
|---|---|
| 49249 (-16287) | Botão 1 |
| 49250 (-16286) | Botão 2 |
| 49251 (-16285) | Botão 3 |

**PEEKs, POKEs e CALLs do DOS**

PRINT (PEEK (978) + 35)/4; "K"
Imprimir tamanho da memória

POKE 4Ø514,52
Permitir um "programa de início" (HELLO) binário

POKE 4Ø514,2Ø
Permitir um "HELLO" de texto

CALL 4235Ø
CATALOG

POKE 42768,234: POKE 42769,234: POKE 4277Ø,234
Cancelar todas as mensagens de erro do DOS

POKE 44452,24: POKE 446Ø5,23
Mostrar 2Ø arquivos no catálogo antes da pausa

POKE 445Ø5,234: POKE 445Ø6,234
Mostrar arquivos deletados no catálogo

POKE 44578,234: POKE 44579,234: POKE 4458Ø,234
Cancelar RETURNs seguindo os nomes no catálogo

POKE 44596,234: POKE 44597,234: POKE 44598,234
Cancelar a parada do catálogo

POKE 44599,234: POKE 446ØØ,234
Parar o catálogo depois de cada arquivo

POKE 491Ø7,234: POKE 491Ø8,234: POKE 491Ø9,234
Impedir o recarregamento do cartão de 16 K num segundo *boot*

**POKES DIVERSOS**

POKE 33,33
Compactar as listagens de programas

POKE 5Ø,128
Tornar listagens e catálogo invisíveis

POKE 5Ø, aleatório
Misturar a saída de texto

POKE 82,128
Rodar programa de cassete assim que for carregado



POKE 214,255
Rodar programa de disco quando é dado qualquer comando

POKE 216,Ø
Cancelar o ONERR

POKE 1Ø1Ø,1Ø2: POKE 1Ø11,213: POKE 1Ø12,112
Fazer RESET = RUN

POKE 1Ø14,165: POKE 1Ø15,214
Fazer & = LIST

POKE 2Ø49,1
Fazer com que a primeira linha seja listada continuamente

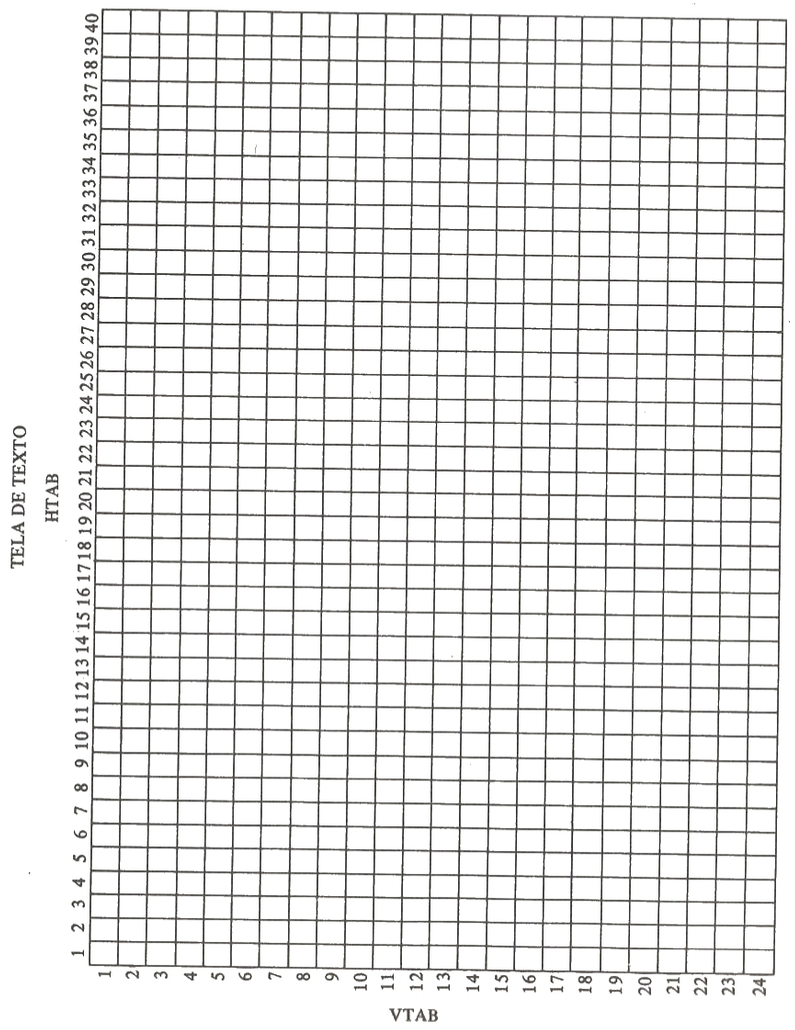
# 5. Mensagens de erro do APPLESOFT

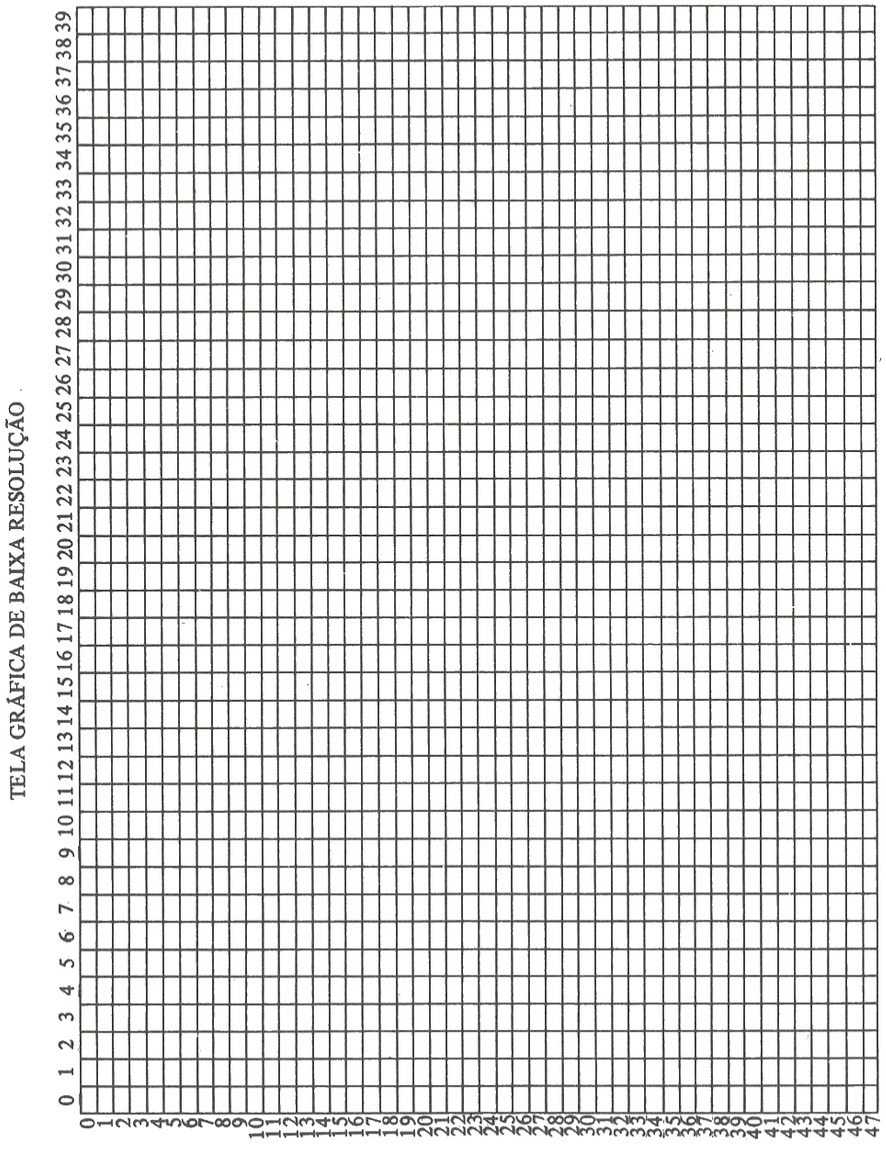
| *erro:* | *explicação:* |
|---|---|
| **BAD SUBSCRIPT** | Foi feita uma tentativa de referência a um elemento de matriz que está fora das dimensões desta. O erro pode também ocorrer se um número errado de dimensões for usado nessa referência. |
| **CAN'T CONTINUE** | Foi feita uma tentativa de reiniciar a execução de um programa com *CONT* quando este não existia, ou depois de um erro, ou após o programa ter sido modificado. |
| **DIVISION BY ZERO** | Houve uma divisão por zero no cálculo de uma expressão numérica. |
| **FORMULA TOO COMPLEX** | Foram executados mais de dois comandos do tipo *IF... THEN.* |
| **ILLEGAL DIRECT** | Os comandos *INPUT, DEF FN, GET* ou *DATA* não podem ser usados no modo de execução direta. |
| **ILLEGAL QUANTITY** | O(s) parâmetro(s) passado(s) para uma função numérica ou *string* estavam fora dos seus limites. |
| **NEXT WITHOUT FOR** | A variável num comando *NEXT* não correspondeu a nenhum *FOR* ainda em efeito. |
| **OUT OF DATA** | Um *READ* tentou ler além do final de uma lista de *DATA.* |
| **OUT OF MEMORY** | Pode ser causado por: programa muito grande; muitas variáveis; mais de 1Ø *FORs* pendentes; mais de 24 *GOSUBs* pendentes; expressão muito complicada; mais de 36 níveis de parênteses; tentativa de colocar o *LOMEM:* muito alto, ou abaixo do valor presente; tentativa de colocar o *HIMEM:* muito baixo. |
| **OVERFLOW** | O resultado de um cálculo foi grande demais para ser representado no formato do BASIC. |
| **REDIM'D ARRAY** | Um comando *DIM* foi encontrado pela segunda vez com referência à mesma variável. |
| **RETURN WITHOUT GOSUB** | Um comando *RETURN* foi executado sem que tenha sido dado um *GOSUB.* |
| **STRING TOO LONG** | Foi feita uma tentativa, pelo uso do operador de concatenação, de se criar um *string* de tamanho maior que 255 caracteres. |
| **SYNTAX** | Pode ser causado por falta de parênteses numa expressão, caractere ilegal numa linha, pontuação incorreta etc. |
| **TYPE MISMATCH** | Foram misturados, de maneira incorreta, valores numéricos e alfanuméricos numa expressão, como na tentativa de igualar um número a um *string,* ou vice-versa. |
| **UNDEF'D FUNCTION** | Foi feita uma referência a uma função do usuário que não tinha sido definida anteriormente. |
| **UNDEF'D STATEMENT** | Foi feita uma tentativa de pular para uma linha inexistente, através de um desvio condicional ou incondicional. |

# 6. Mensagens de erro do DOS

| *erro:* | *explicação:* |
|---|---|
| **DISK FULL** | Há um excesso de arquivos no disco. |
| **END OF DATA** | Foi feita uma tentativa de ler além do final de um arquivo com *READ*. |
| **FILE LOCKED** | Foi feita uma tentativa de alterar um arquivo protegido com *LOCK*. |
| **FILE NOT FOUND** | O arquivo especificado não se encontra no disco. |
| **FILE TYPE MISMATCH** | O tipo de arquivo não serve para o comando dado. |
| **I/O ERROR** | Erro na leitura ou gravação de um arquivo no disco. |
| **LANGUAGE NOT AVAILABLE** | A linguagem que o programa especificado necessita para rodar não está na memória. |
| **NO BUFFERS AVAILABLE** | Muitos arquivos estão abertos ao mesmo tempo. |
| **NOT DIRECT COMMAND** | O comando dado não pode ser usado no modo de execução direta. |
| **PROGRAM TOO LARGE** | Memória insuficiente para carregar o programa especificado. |
| **RANGE ERROR** | O parâmetro dado para um comando está fora dos limites. |
| **SINTAX ERROR** | Nome de arquivo, parâmetro ou vírgula ilegal. |
| **VOLUME MISMATCH** | Parâmetro de volume não coincide com o do disco. |
| **WRITE PROTECTED** | O disco está protegido contra gravação. |

# 7. Formato das telas de texto e de gráfico de baixa resolução






# 8. Editando: o que eu faço se cometer enganos

## 8.1 COMO EDITAR

Para podermos corrigir os enganos cometidos durante a digitação de um programa, apresentaremos alguns procedimentos úteis para auxiliá-lo na edição desse material. A relação que se segue mostra as teclas necessárias:

A tecla ← (*backspace*), além de retornar o cursor, *apaga* o caractere sobre o qual passou. Contudo, os caracteres continuam sendo visualizados na tela do terminal de vídeo, embora eles já tenham sido *apagados* da linha do programa existente na memória do computador.

A tecla → (*retype*) move o cursor para frente e não *apaga* o caractere sobre o qual ele passa. O caractere repassado permanece tanto na tela quanto na linha de programação.

A tecla ESC (*escape*), usada em conjunto com as teclas I, J, K e M, pode mover o cursor para cima, para a esquerda, para a direita e para baixo, respectivamente.

O comando CTRL-X pode ser usado quando você quiser *dizer* ao computador para ignorar a linha de programa parcialmente digitada por você. Ele produz um "\", que será colocado na posição do cursor. Basta pressionar RETURN para reiniciar a definição de uma nova linha.

## 8.2 MUDANDO LINHAS DE PROGRAMAS

Sem dúvida, um modo simples de corrigir ou mudar uma linha de um programa é digitar essa linha inteira novamente. Este é um bom procedimento para correções, desde que a linha não seja muito *comprida.* Com linhas longas o processo de *retype* é muito demorado e tedioso, além de aumentar a probabilidade de cometermos erros de digitação. O Apple apresenta uma edição vantajosa para esses casos.

## 8.3 O MODO DE EDIÇÃO

Para que possamos entrar no modo de edição, somente quando o cursor estiver visualizado no vídeo pressionamos a tecla:

ESC

Nessa situação as quatro teclas, **I, J, K** e **M**, poderão ser usadas para mover o cursor para cima, para a esquerda, para a direita e para baixo, respectivamente, conforme mostra a figura 8.1.

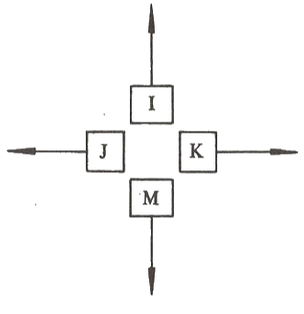



fig. 8.1

## 8.4 CORRIGINDO UM ENGANO

A seguir, veremos como o modo de edição pode ser utilizado para corrigir ou alterar linhas de programas. Inicialmente vamos digitar um programa-exemplo que, obviamente, necessitará de correção:

```
1Ø HOME
2Ø INPUT "QUAL O VALIR DE X? ";X
3Ø PRINT
4Ø PRINT "O QUADRADO DE ";X;" E' ";X*X
```

Na linha **2Ø**, a palavra *VALOR* foi digitada erradamente. Vamos **LIST**á-la separadamente, para uma melhor edição dessa linha, digitando **LIST 2Ø**:

```
2Ø INPUT "QUAL O VALIR DE X? ";X
```

Para edição, siga os seguintes passos:

1. Pressione a tecla **ESC**, entrando no modo de edição.
2. Pressione a tecla **I**, quantas vezes for necessário, para mover o cursor para cima até atingir a linha **2Ø**.
3. Pressione a tecla **J** para mover o cursor para a esquerda, até ficar à direita do primeiro dígito do número da linha.
4. Use a tecla → para copiar todo o conteúdo da linha até atingir a letra **I**. Nessas condições o vídeo estará mostrando:

```
2Ø INPUT "QUAL O VAL■R DE X? ";X
```



A tecla → copia cada caractere sem termos necessidade de redigitá-lo. Certifique-se de que você não esteja usando a tecla **K**, que desloca o cursor para a direita, mas sem copiar qualquer caractere.

5. Digite a letra correta, *O*, e, a partir desse ponto, recopie todo o restante da linha com a tecla →.
6. Pressione **RETURN** e a correção estará completada.

Para confirmar a correção, faça a listagem do programa.

## 8.5 INSERINDO CARACTERES

No modo de edição podemos também inserir caracteres (ou palavras) em qualquer ponto de uma linha, desde que tenha espaço disponível. Digamos que você tenha esquecido de colocar a mensagem da linha **2Ø**, digitada da seguinte forma:

```
2Ø INPUT X
```

Em primeiro lugar, digite a linha separadamente. Usando o mesmo procedimento descrito anteriormente, recopie a primeira porção da linha até o local onde se posicionará o primeiro caractere da inserção a ser feita. Nessas condições, teremos:

```
2Ø INPUT ■
```

Agora pressione **ESC** para entrar no modo de edição e **I** para mover o cursor um espaço para cima. Você estará vendo:

```
   ■
2Ø INPUT X
```

Agora devemos deixar o modo de edição pressionando uma tecla qualquer (menos **I, J, K, M, REPT, RESET, CTRL** ou **SHIFT**); nenhuma mudança visível será notada e todas as teclas que se seguirem serão memorizadas pelo computador. Digite:

```
"QUAL O VALOR DE X? ";
```

e você verá:

```
          "QUAL O VALOR DE X? ";■
2Ø INPUT X
```

Agora temos que entrar novamente no modo de edição, pressionando a tecla **ESC**. Em seguida, pressione a tecla **J** até o cursor ficar à direita do primeiro caractere da inserção. Assim:

```
          ■QUAL O VALOR DE X? ";
2Ø INPUT X
```

Pressione a tecla **M** para mover o cursor para baixo, até atingir a linha editada. Esta será a localização do primeiro caractere da inserção. A situação, agora, é:

```
           "QUAL O VALOR DE X? ";
20 INPUT ■
```

Novamente, teremos que sair do modo de edição, pressionando qualquer tecla. No vídeo, não acontecerá mudança alguma, mas já saímos do modo de edição. Pressione a tecla → até que o cursor ultrapasse o último caractere da linha em edição.

```
           "QUAL O VALOR DE X? ";
20 INPUT X■
```

Pressione **RETURN** e teremos a inserção completada. Para verificar, digite **LIST.**

## 8.6 COMPACTANDO A LISTAGEM

Quando listamos um programa, o computador insere espaços extras para facilitar a sua leitura. Ao corrigir uma linha muito longa (duas ou mais linhas de vídeo), esses espaços extras poderão aparecer na apresentação final. Para suprimir esses espaços, formataremos a tela através de um **POKE.** Digite:

```
POKE 33,33
```

Nessa situação, liste a linha a ser corrigida e proceda de acordo com o descrito anteriormente. Uma vez terminada a edição, para voltar à formatação normal digite:

```
POKE 33,40
```

# 9. Mensagens de erro do MAXXI

| *mensagem do Apple:* | *equivalente do Maxxi:* |
|---|---|
| APPLESOFT | POLYSOFT |
| **BAD SUBSCRIPT** | **INDICE ERRADO** |
| **CAN'T CONTINUE** | **CONT INVALIDA** |
| **DIVISION BY ZERO** | **DIVISAO POR ZERO** |
| **FORMULA TOO COMPLEX** | **CAL MUITO COMPLEXO** |
| **ILLEGAL DIRECT** | **DIRETO ILEGAL** |
| **ILLEGAL QUANTITY** | **QUANT. INVALIDA** |
| **NEXT WITHOUT FOR** | **"NEXT" SEM "FOR"** |
| **OUT OF DATA** | **FINAL DADOS** |
| **OUT OF MEMORY** | **MEM EXCEDIDA** |
| **OVERFLOW** | **OVERFLOW** |
| **REDIM'D ARRAY** | **MATRIZ REDIM** |
| **RETURN WITHOUT GOSUB** | **"RETURN" SEM "GOSUB"** |
| **STRING TOO LONG** | **STRING EXCEDIDA** |
| **SYNTAX** | **SINTAXE** |
| **TYPE MISMATCH** | **TIPO DISCORDA** |
| **UNDEF'D FUNCTION** | **FN NAO DEFINIDA** |
| **UNDEF'D STATEMENT** | **COMANDO INVALIDO** |
| | |
| DOS | |
| | |
| **DISK FULL** | **REPLETO** |
| **END OF DATA** | **FINAL DADOS** |
| **FILE LOCKED** | **ARQ PROTEG** |
| **FILE NOT FOUND** | **ARQ NAO EXISTE** |
| **FILE TYPE MISMATCH** | **TIPO ARQ DISCORDA** |
| **I/O ERROR** | **ERRO E/S** |
| **LANGUAGE NOT AVAILABLE** | **LINGUAGEM INEXISTENTE** |

| | |
|---|---|
| NO BUFFERS AVAILABLE | NAO HA MAIS BUFFERS |
| NOT DIRECT COMMAND | COMANDO NAO DIRETO |
| PROGRAM TOO LARGE | MEMORIA EXCEDIDA |
| RANGE ERROR | VAL ERRADO |
| SYNTAX ERROR | SINTAXE: ERRO |
| VOLUME MISMATCH | VOL DISCORDA |
| WRITE PROTECTED | GRAV. PROTEGIDA |

Gentileza de Polymax Sistemas e Periféricos S.A.

# 10. Profissionalizando seu Apple

## EXPANSÃO DE MEMÓRIA

Existem vários tipos de cartões de expansão de memória disponíveis no mercado para o Apple, mas a expansão de **16K** é a que mais se encontra. Este cartão permite substituir logicamente, e sempre controlado por *software*, os programas que estão gravados permanentemente em **ROM** na memória mais alta do computador pelos **16K** de **RAM** do cartão. Dessa forma, o usuário pode carregar o **RAM** com outra linguagem armazenada em disco (como **Pascal, Integer BASIC, LOGO, FORTRAN** ou **COBOL**) e usá-la como se essa fosse a linguagem residente no computador.

O cartão de expansão de memória pode ainda ser usado em conjunto com a longa série de programas aplicativos que reconhecem a sua presença no Apple, aumentando efetivamente a capacidade de armazenamento interno e portanto o tamanho do trabalho que pode ser executado pelo aplicativo. Há ainda vários pacotes de *software* que aceitam e aproveitam cartões de expansão maiores, normalmente encontrados em incrementos de **16** ou **64K** de memória, até um máximo de **128K**. Existem também programas que permitem ao programador o uso da memória adicional como se esta fosse uma unidade de disco; dessa forma, programas ou arquivos de texto freqüentemente manipulados durante uma sessão no computador podem ser transferidos do disco para a memória do cartão e acessados, como se ainda estivessem gravados em um disco, a uma velocidade várias vezes maior.

## CARTÃO Z-8Ø

O **Z-8Ø** é um microprocessador muito usado na indústria de microcomputadores. Sobre ele foi desenvolvido o sistema operacional **CP/M**, que logo se tornou o sistema mais popular entre os usuários e programadores de *micros*. Sua biblioteca inclui atualmente milhares de programas de todo tipo, desde jogos de aventura até processadores de texto profissionais e sistemas de gerenciamento de bases de dados.

O cartão **Z-8Ø** contém este processador (que é diferente do **65Ø2** do Apple) e todos os circuitos necessários para que este possa trabalhar eficientemente no Apple. O cartão pode ser ativado e desativado por controle de *software*, permitindo que o **Z-8Ø** controle o computador em lugar do **65Ø2**. Um disco que trabalha em **CP/M** automaticamente ativa o **Z-8Ø** e prepara o computador para usar o novo sistema operacional.

Além dos aplicativos prontos da biblioteca de **CP/M** e dos grupos de usuários desse sistema, o programador tem acesso a um número razoável de linguagens de programação que trabalham sob **CP/M**, incluindo várias versões estendidas de **BASIC, FORTRAN, COBOL, Pascal, ALGOL, APL** e outros.

## CARTÃO DE VÍDEO – 8Ø COLUNAS

Através deste cartão, o usuário pode ter em seu monitor de vídeo uma apresentação de texto de 24 linhas por *8Ø colunas*, exatamente o dobro da capacidade da tela normal do Apple. Seu uso facilita muito a operação de processadores de texto (sempre que, é claro, estes reconheçam a presença do cartão), permitindo que o texto seja visto na tela no mesmo formato em que aparecerá impresso. Muitos aplicativos, como as conhecidas planilhas eletrônicas, permitem a utilização deste cartão, podendo apresentar de uma só vez o dobro da informação possível sem este.

Para o programador, ter um vídeo de 8Ø colunas significa poder programar em Pascal, COBOL ou qualquer uma das linguagens que tem o seu *formato de linha* baseado no conceito do cartão perfurado de 8Ø colunas, tendo a linha completa no monitor à sua frente.

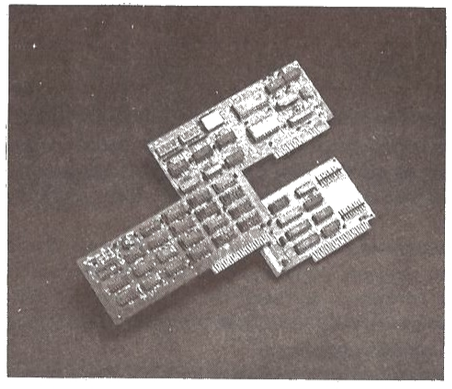
## CARTÃO DE COMUNICAÇÕES

É um cartão que permite a transmissão *serial* de dados entre um Apple e outro aparelho. Este pode ser outro Apple ou outro tipo de computador, ligado fisicamente ao primeiro por fios. Pode ainda ser uma impressora ou até um *modem*, que permite a comunicação serial entre computadores à distância, através da rede telefônica.

Através de um *modem*, o usuário do computador pode acessar bancos de dados, serviços fornecidos por empresas participantes ou bibliotecas de *software* tornadas públicas por organizações de usuários de computadores. Com o tempo, o número de serviços disponíveis por meio do computador e da rede telefônica irá se expandir, atingindo a todos os níveis de usuários.

## CARTÃO SERIAL SÍNCRONO

O cartão *serial síncrono* é um cartão de comunicações, como o descrito acima, para possibilitar a ligação entre o Apple e outros computadores. A principal diferença entre eles está na maneira com que os dados são transmitidos de um computador a outro, possibilitando a conecção direta de seu computador com outros de grande porte, do tipo *main frame*, que em geral mantêm comunicação síncrona.



Cortesia de Appletrônica – Computadores e Sistemas

# Referências bibliográficas


APPLE COMPUTER. *Basic programming reference manual.* USA, 1981. 168 p.
BERNARD, William. *Faça seu teste.* 6. ed. São Paulo, Mestre Jou, 1973. 400 p. 4 vols.
CASSEL, Don. *BASIC made easy.* USA, Prentice-Hall, 1980. 240 p.
DEITEL, Harvey. *Introduction to computer programming.* USA, Prentice-Hall, 1977. 253 p.
HEISERMAN, David. *Programming in BASIC.* USA, Prentice-Hall, 1981. 333p.
KAMINS, Scot. *Apple backpack.* USA, Byte Books, 1982. 182p.
KOFFMAN, Elliot B. *Problem solving and structured programming in BASIC.* USA, Addison-Wesley, 1979. 430 p.
LOAN, James. *Basic BASIC.* USA, Hayden Book Company, 1978. 270 p.
MIRSHAWKA, Victor. *Linguagem BASIC.* São Paulo, Livraria Nobel, 1983. 365 p.
POLIMAX SISTEMAS E PERIFÉRICOS S.A. *Operação e linguagem.* 2. ed. 1983. 227 p.
—. *Linguagem BASIC.* 1983. 237 p.
POOLE, Lon. *Apple II user's guide.* USA, Osborne, 1981. 385 p.
RICHARDSON, Caryl. *The Applesoft tutorial.* USA, 1981. 158 p.
TAHAN, Malba. *O homem que calculava.* 11. ed. Saraiva, 1949. 212 p.
TUCCI, Wilson J. et alii. *A primeira mordida.* São Paulo, Livraria Nobel, 1983. 186 p.
ZABINSKI, Michael P. *Introduction to TRS 80.* USA, Prentice-Hall, 1980. 186 p.

*Periódicos nacionais:*

*BITS.* R. Casa do Ator, 1.060 – 04546 – São Paulo.
*Dados e Idéias.* R. Major Quedinho, 90 – 01050 – São Paulo.
*INFO.* Av. Brasil, 500 – 20940 – Rio de Janeiro.
*Interface.* R. Senador Dantas, 117 – 20000 – Rio de Janeiro.
*Microhobby.* Caixa Postal 60.081 – 05096 – São Paulo.
*Micromundo.* R. Alcindo Guanabara, 25 – 20031 – Rio de Janeiro.
*Microsistema.* Al. Gabriel Monteiro da Silva, 1.229 – 01441 – São Paulo.
*RNT.* Av. Paulista, 1.159 – 01311 – São Paulo.

*Periódicos estrangeiros:*

*Apple Orchad.* P. O. Box 1493 – Beaverton, Oregon 97075.
*Byte.* P. O. Box 328 – Hancok, New Hampshire 03449.
*In Cider.* 80 Pine SE – Peterborough, New Hampshire 03458.
*Nibble.* P. O. Box 325 – Lincoln, Massachussetts 01773.

№ 1266

IMPRESSO NAS OFICINAS DA EDIÇÕES LOYOLA
RUA 1822 N.° 347 - TEL.: 914-1922 - SÃO PAULO

A sair:
POR DENTRO DO **DOS**
POR DENTRO DO **LOGO**
POR DENTRO DO **VISICALC**

POR DENTRO DO APPLE leva o leitor, passo a passo, através da linguagem do APPLE, desde um nível introdutório até a apresentação de técnicas avançadas para otimizar o processamento de programas no computador, através de exemplos e aplicações práticas.

Servindo como texto fundamental e como modelo didático-pedagógico aos cursos de BASIC e de fundamentos de processamento de dados, o livro dirige-se a estudantes, profissionais e mesmo a pessoas que não tenham conhecimento prévio de computação.


ISBN 85-213-0193-6