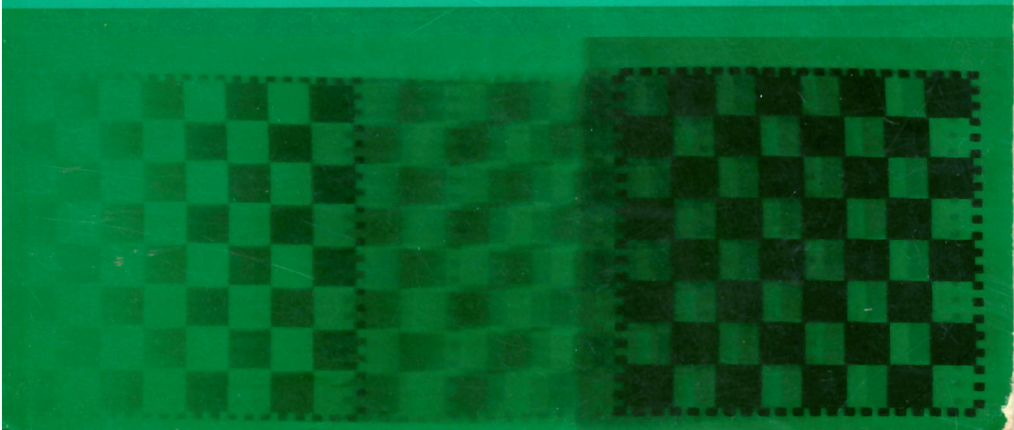
ELEMENTOS DE PROGRAMAÇÃO EM BASIC

LÉO BATISTA

GERSON M. KATAKURA



ELEMENTOS DE PROGRAMAÇÃO EM BASIC

LÉO BATISTA

Doutor em Engenharia pela Escola Politécnica da Universidade de São Paulo

GERSON M. KATAKURA

Engenheiro Eletrônico pela Escola Politécnica da Universidade de São Paulo

ELEMENTOS DE PROGRAMAÇÃO EM BASIC

EDITORA EDGARD BLÜCHER LTDA.

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

Batista, Léo, 1937-
B337e Elementos de programação em BASIC / Léo Batista, Gerson M. Katakura. -- São Paulo : Edgard Blücher, 1983.

Apêndices: A. Diagramas de blocos. -- B. Um jogo simples.
Bibliografia.

1. BASIC (Linguagem de programação para computadores) 2. Microcomputadores 3. Microcomputadores - Programação 4. Programação (Computadores eletrônicos) I. Katakura, Gerson Mitsutoshi, 1957- II. Título.

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

83-0046

Índices para catálogo sistemático:

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

2ª reimpressão 1984

EDITORA EDGARD BLÜCHER LTDA.
01000 Caixa Postal 5450
End. Telegráfico: BLÜCHERLIVRO
São Paulo - SP - Brasil

Impresso no Brasil *Printed in Brazil*

PREFÁCIO

"Longe do estéril turbilhão da rua,
Beneditino, escreve! No aconchego
Do claustro, na paciência e no sossego,
Trabalha, e teima, e lima, e sofre, e sua!

Mas que na forma se disfarce o emprego
Do esforço; e a trama viva se construa
De tal modo, que a imagem fique nua,
Rica, mas sóbria, como um templo grego.

Não se mostre na fábrica o suplício
Do mestre. E, natural, o efeito agrade,
Sem lembrar os andaimes do edifício:

Porque a Beleza, gêmea da Verdade,
Arte pura, inimiga do artifício,
É a força e a graça na simplicidade."

Olavo Bilac

Os microcomputadores estão em fase de grande desenvolvimento, tanto em melhoria de desempenho como em divulgação. Para esta é fundamental que a programação seja simples e que possam ser tratados problemas relativos a diversas áreas do conhecimento.

O objetivo principal deste livro é ensinar pessoas que não conhecem um computador, a programar no menor tempo possível e com a máxima facilidade. O método que adotamos foi omitir todo o possível sobre o maquinário e aplicar todo o esforço sobre a lógica externa ao computador. É perfeitamente possível programar e *muito bem*, sem conhecer detalhes do maquinário.

Vamos tratar da linguagem BASIC, que é atualment a principal linguagem dos microcomputadores e, portanto aquela que é utilizada por maior número de programadore em todo o mundo. O BASIC é ainda a mais simples das lin guagens e pode ser aprendida com facilidade por pessoa de qualquer idade.

Desta forma, mesmo crianças podem, com algumas hora de estudo, iniciar-se na programação de computadores, o qu será um alto benefício para sua formação.

No outro extremo estão pessoas com estudos superiore completos e que se mantêm à distância dos computadores numa espécie de receio, às quais pretendemos mostrar quanto é simples, divertido e excitante operar um computador

Existem clubes de computação onde as pessoas se di vertem programando, e podemos assegurar que, além d bom divertimento, o sentimento de realização intelectual quando um programa funciona bem, é muito gratificante

O livro está escrito em linguagem simples, direta e tã precisa quanto nos foi possível, para torná-lo acessível todas as pessoas, sem distinção de idade ou escolaridade

Os comandos e as instruções estão separados em prin cipais e avançados, para facilitar o aprendizado. Só com o principais já é possível programar e obter resultados; o avançados podem ficar para um refinamento posterior.

Muito se aprende estudando programas já prontos; po isso apresentamos alguns exemplos nos capítulos finais.

O contato direto com um computador e a análise cuida dosa de suas mensagens de erro permitirão aperfeiçoar conhecimento do programador.

Léo Batista
Gerson M. Katakura

CONTEÚDO

Capítulo 4

PROGRAMAÇÃO
(programas ilustrativos)

Capítulo 5

COMANDOS, INSTRUÇÕES E FUNÇÕES AVANÇADOS

Capítulo 6

APLICAÇÕES PRÁTICAS

Apêndice A

Apêndice B

Capítulo 1

APRESENTAÇÃO

1.1 – Conceitos fundamentais

Comando é uma ordem direta ao computador para ser executada imediatamente.

Instrução é uma ordem ao computador para ser executada numa posição definida de uma seqüência.

Programa é um conjunto ordenado de instruções que se fornece a um computador com o objetivo de levá-lo a executar uma determinada tarefa.

O conjunto de instruções que os computadores atuais estão preparados para entender é muito limitado e, por isso, a programação é uma arte na qual se reúnem lógica e criatividade.

Esses conjuntos de instruções acrescidos de alguns comandos constituem uma *linguagem*; há várias linguagens, como por exemplo: BASIC, FORTRAN, COBOL, PASCAL e muitas outras, cada uma com suas características próprias.

Freqüentemente um programa pode incluir outros menores, mais simples e que podem ser utilizados mais de uma vez, são as *sub-rotinas*.

A unidade elementar de memória é o *bit*, mas para guardar uma informação útil, são necessários alguns *bits* e freqüentemente eles são agrupados em 4, 8, 16 ou mais, formando um *byte*. Como é constituída essa memória é um assunto de interesse apenas para os construtores do *maquinário*, neste livro trataremos apenas da *lógica*; para tanto um *byte* será simplesmente uma unidade de memória e deve-se lembrar que as memórias da máquina são limitadas a alguns milhares de bytes, kbytes (quilo-bytes).

1.2 – **Meios de entrada e saída**

Para estabelecer a comunicação com a máquina é preciso elementos que, acionados por um operador, sejam inteligíveis para a máquina e vice-versa.

No estágio atual da indústria, os dados, instruções e comandos fornecidos à máquina entram por um *teclado*, na maior parte dos casos, sendo convertidos em linguagem da máquina em um processo de várias etapas.

O armazenamento de informações é, geralmente, em fitas ou discos magnéticos (antigamente em cartões ou fitas perfuradas).

A entrada de dados e instruções pelo teclado é uma fase que pode ser penosa e demorada, quando há muita informação a ser transmitida à máquina. Para evitar a repetição e a conseqüente demora deste trabalho, costuma-se gravar em discos magnéticos, ou mesmo em fitas, os programas e os dados.

Quando a função do computador é controlar outras máquinas, por exemplo, trens, aviões, semáforos, instrumentos de medida, processos químicos, robôs, então os dados podem ser obtidos diretamente através de sistemas de aquisição de dados.

Para a apresentação dos resultados do programa pelo computador podem ser utilizados uma tela de vídeo ou cristal líquido, uma impressora ou um sintetizador de voz. Atualmente, nos microcomputadores, o mais utilizado é a tela de vídeo, embora uma impressora seja essencial nas aplicações profissionais.

Os resultados também poderiam ser armazenados em discos magnéticos ou fitas, para consulta ou utilização posteriores, nos casos de programas cuja execução fosse muito demorada.

Para o controle de outras máquinas, como nos referimos anteriormente, a saída é feita através de conversores de sinais adequados, denominados interfaces, assunto que pertence à área do *maquinário* e, portanto, fora do nosso escopo que é a *lógica*.

1.3 – **Resumo**

Comando: ordem direta ao computador para execução imediata.

Instrução: ordem ao computador para execução dentro de uma seqüência.

Programa: conjunto ordenado de instruções com um objetivo.

Linguagem: conjunto de instruções e comandos.

BASIC: linguagem simples e a mais utilizada em microcomputadores.

Sub-rotina: programa menor, embutido no principal.

Byte: unidade de memória, constituída por alguns *bits*.

Maquinário: a parte física do computador, constituída por componentes eletrônicos, elétricos, eletromecânicos ou mecânicos (em inglês, *hardware;* em francês, *materiel*).

Lógica: o conjunto das diretrizes gerais, incluindo programas, estabelecidas para o funcionamento e obtenção do resultado (em inglês, *software*; em francês, *logiciel*).

Teclado: principal meio de entrada de dados, programas e comandos.

Discos magnéticos: os mais eficientes armazenadores de dados e programas.

Fitas magnéticas: armazenadores econômicos para instalações de amador.

Impressora: máquina de escrever, acionada diretamente pelo computador, para fornecer listas, resultados ou gráficos rudimentares.

Capítulo 2

VARIÁVEIS EXPRESSÕES E FUNÇÕES

2.1 – Variáveis

Uma variável é utilizada para armazenar informação que pode se alterar no decorrer do programa.

Quando a informação é um número, a variável que a representa é do tipo numérico e usualmente indicada por uma única letra como X, Y, A, ou uma mistura de letras e números sempre iniciada por uma letra como BK, A1, XY, K4, etc.

Por outro lado, se a informação a ser armazenada não é um valor numérico, então a variável que a representa é denominada alfanumérica e diferenciada das numéricas por ser seguida de um cifrão. Exemplo: A$ e X$ indicam que A e X são variáveis alfanuméricas.

As variáveis deste tipo representam uma fileira de caracteres (em inglês, *string*). X$ deve ser lido: X *cifrão*, ou X *fileira*, ou ainda X *seqüência*.

Há valores que permanecem inalterados durante a execução do programa, são as constantes.

2.2 – Expressões

Uma expressão estabelece o relacionamento entre variáveis e constantes. São exemplos de expressões:

$2C + 4$
$3K + A - B$
$Y^2 + 2Y + 5$

Os números são as constantes e as letras as variáveis.

Para que o computador entenda estas expressões é preciso fornecê-las utilizando os símbolos da seguinte tabela:

Símbolo	*Significado*
*	Multiplicar
/	Dividir
+	Somar
−	Subtrair
↑	Elevar à potência

Para a potenciação também são utilizados outros símbolos:

**, ∧ ou [

As expressões mencionadas anteriormente, escritas com estes símbolos, tornam-se:

2 * C + 4
3 * K + A − B
Y ↑ 2 + 2 * Y + 5

A separação entre a parte inteira e a fracionária de um número é feita com *um ponto*. Não se pode usar mais de um ponto; também não se pode usar vírgula.

Exemplo:

3,43 deve ser escrito 3.43

Numa expressão em que há várias operações indicadas, elas serão executadas na seguinte ordem:

1.º) Operações entre parênteses
2.º) Potenciação
3.º) Multiplicação e divisão
4.º) Soma e subtração

Exemplo:

Na expressão: 3 * D ↑ 2 + E o cálculo de D^2 é efetuado em primeiro lugar, seguido pela multiplicação por 3 e posteriormente a soma com E.

As operações com a mesma prioridade são executadas da esquerda para a direita.

Exemplo:

8 2 * 3

tem resultado 12, pois é inicialmente efetuada a divisão de 8 por 2, por estar mais à esquerda, e a seguir o produto por 3; isto porque a divisão e a multiplicação têm a mesma prioridade e, neste caso, é executada primeiro aquela que está mais à esquerda.

Qualquer que seja a operação indicada, ela será executada em primeiro lugar se estiver entre parênteses.

Exemplos:

1.º) (A * B) ↑ 2

Apesar da potenciação ter prioridade sobre o produto, este será calculado primeiro, por estar entre parênteses, e o resultado será posteriormente elevado à potência 2.

2.º) Deseja-se somar X com Y e depois multiplicar a soma por Z.

A expressão deverá ser escrita: (X + Y) * Z

Podem ser empregados também parênteses múltiplos e, neste caso, serão executados primeiro os mais internos e depois os mais externos.

Exemplo:

((A + B) * 2) ↑ 5

A execução será na ordem:

1.º) Soma de A e B

2.º) Multiplicação da soma por 2

3.º) Elevação do produto à 5.ª potência

2.3 – Notação exponencial

Para representar números que exigiriam muitos algarismos, por serem muito grandes ou muito pequenos, costuma-se multiplicá-los por 10 elevado a uma potência. Esta potência é negativa para os números menores do que um, e positiva para os maiores.

Escreve-se 1 diante da letra "E", seguida do expoente que se aplica.

Exemplo:

1E + 8 significa 10^8
1E – 5 significa 10^{-5}

De outra forma, esses números deveriam ser escritos:

100000000
e
0.00001

onde se nota inconveniente para apresentação e também para leitura. A notação exponencial é mais concisa e elegante.

No caso de números que não sejam uma potência exata de 10, pode-se escrevê-los como um produto de dois fatores.

Exemplo:

0.0002894 pode ser escrito 2.894 E – 04

O computador optará automaticamente por esta notação ao apresentar certos resultados.

2.4 – Funções definidas

A maioria dos computadores permite a utilização de algumas funções definidas em uma expressão. A disponibi-

lidade e a quantidade destas variam com o tipo da máquina; são apresentadas a seguir algumas das mais comuns.

ABS (X) Fornece o valor absoluto de X, ou seja, o seu módulo; portanto o resultado quando se aplica esta função é sempre um número positivo.

ATN (X) Fornece o arco-tangente X, em radianos.

ACS (X) Esta função fornece o arco-coseno X, em radianos.

ASN (X) Fornece o arco-seno X, em radianos.

COS (X) Fornece o valor da função co-seno de X. O argumento X deve estar em radianos.

EXP (X) Realiza o cálculo de e^x.

FRA (X) Ao se aplicar esta função sobre X, tem-se como resultado a parte fracionária do argumento X.
Esta função mantém o sinal do argumento.

INT (X) Obtém-se pela aplicação desta função o maior inteiro que se obtém arredondando para baixo.
Esta função mantém o sinal do argumento.

LOG (X)
ou
LN (X) Fornece o valor do logaritmo natural ou neperiano.

SGN (X) Esta função fornece como resultado o valor +1 quando X é positivo, −1 quando X é negativo e zero quando X é zero.

SIN (X) Fornece o valor do seno de X. O argumento X deve estar em radianos.

SQR (X) O resultado desta função é a raiz quadrada de X. O valor do argumento X não pode ser negativo.

As funções apresentadas têm a forma geral F (X), onde X é o argumento sobre o qual se calcula a função X. Há ainda:

PI Fornece o valor =3,14159...

RND Fornece um valor pseudo-aleatório entre 0 e 1. Em algumas versões esta função exige argumento.

Exemplo:

Função	*Resultado*
ABS (4.5)	4.5
ABS (−3)	3
FRA (2.14)	0.14
FRA (−3.17)	−0.17
INT (4.7)	4
INT (−3.2)	−4

Conforme já mencionado, as funções podem ser utilizadas em uma expressão.

2 * INT (6.17) tem como resultado 12, pois INT (6.17) é 6.

Em uma função F(X), o argumento X pode ser uma expressão, sendo que esta deve, obrigatoriamente, estar em parentêses.

ABS (−4 * 1.2) tem como resultado 4.8, pois o cálculo da função é feito sobre o resultado da expressão utilizada como argumento.

2 * SQR (4 * D) para D = 2.25, o valor desta expressão é 6, pois será calculada a raiz quadrada de 9 e, em seguida, feita a multiplicação por 2.

Pode ocorrer a formação de funções encadeadas, desde que estas sejam convenientemente delimitadas por parênteses.

Exemplo:

SQR (4 * INT (2.12) + 8) cujo resultado é 4,
10 * COS ((3 * PI + PI) / 2) cujo resultado é 10.

2.5 — **Sumário de conceitos**

Variável: armazenador de informação, esta em forma de número ou fileira. Seu conteúdo pode ser alterado durante a execução do programa.

Constante: valor utilizado sem alteração durante a execução do programa.

Expressão: estabelece a forma pela qual as constantes e as variáveis estão relacionadas.

Operadores aritméticos: símbolos utilizados para indicar as operações de soma, subtração, multiplicação, divisão e potenciação.

. (ponto): separa a parte inteira da parte fracionária de um número. Deve ser utilizado em lugar da vírgula.

( ) (parênteses): utilizado para dar a maior prioridade às operações sobre as quais se aplicam.

Argumento: valor sobre o qual será calculada uma função.

Capítulo 3

LINGUAGEM BASIC

3.1 – **Apresentação**

A linguagem BASIC é uma das mais simples entre as linguagens de programação, porque é constituída por poucas instruções e comandos; passou também a ser a utilizada pelo maior número de pessoas, após a produção em grande escala de microcomputadores que a empregam.

Não há uma linguagem BASIC única, mas sim uma família de dialetos, os quais são, entretanto, muito semelhantes. Por esse motivo, ao passar de um computador para outro, é sempre necessário verificar quais são as instruções e comandos existentes e também eventuais diferenças entre eles. Quase sempre os programas necessitam alguma adaptação.

3.2 – **Mensagens de erro**

Há as mensagens de erro que são emitidas toda vez que um erro é detectado pelo computador; essas mensagens são muito diferentes de um computador para outro, e dessa forma não serão tratadas neste livro.

Recomenda-se, ao iniciar o trabalho com um determinado computador, ler rapidamente o glossário de mensagens de erro e tê-lo sempre a mão, de forma a poder ser consultado com facilidade.

Os tipos de erro que podem ser detectados são aqueles que desrespeitam alguma regra explícita.

Erros na elaboração de um programa que não envolvam violação de regras formais, não serão detectados e po-

dem ser apresentados resultados falsos; portanto é necessário muito cuidado na programação.

Ocorrem também erros de arredondamento, mas estes só são importantes em alguns casos especiais.

3.3 — **Elementos de programação**

As instruções e comandos são escritas em inglês ou são suas abreviações; é conveniente manter assim pela sua universalidade e por ser difícil a mudança.

Um programa é constituído por linhas, cada uma recebendo um número. É de boa prática numerar de dez em dez, deixando espaço para encaixe de linhas suplementares.

Cada instrução em um programa deve ser obrigatoriamente precedida por um número de linha.

As linhas podem ser numeradas de 1 a 9999 na maioria dos sistemas, em alguns até 99999 ou ainda outro número.

As instruções serão executadas na ordem do número das linhas e não na ordem de entrada. Dessa forma, as instruções podem ser escritas em qualquer ordem, desde que numeradas na ordem correta para execução.

Não pode haver duas linhas com o mesmo número; se forem escritas assim, a última é a que vale, e a anterior é perdida.

Um comando, conforme definido no capítulo 1, é uma ordem direta ao computador para ser executada imediatamente e, portanto, não deve nem pode ser precedido por um número de linha. Os comandos somente são executados após ser pressionada a tecla que corresponde ao fim de linha.

Nos parágrafos seguintes são apresentados comandos e instruções BASIC mais comuns. Em alguns computadores nem todos poderão ser utilizados, podendo ocorrer também o contrário, isto é, haver instruções específicas, não descritas aqui, que possam ser aplicadas.

No caso de se tentar empregar uma instrução para a qual a máquina não foi preparada, geralmente ela emitirá uma mensagem de erro.

3.4 — Comandos principais

Os comandos somente são executados após a tecla "NEWLINE" ou "RETURN" ser acionada. O nome "RETURN", ou abreviadamente "CR", significa retorno do carro e está associado aos teclados mecânicos antigos, teletipos, onde havia um carro que retornava no fim da linha, como uma máquina de escrever. Nos teclados modernos, com vídeo eletrônico, o nome perdeu o sentido e, por isso, a tecla é, em alguns equipamentos, denominada "NEWLINE", "EOL" (end of line) ou "ENTER", indicando o fim da linha ou do comando. Em português poderia ser "FL", fim de linha, abreviação esta que será usada, já que não há uniformidade entre os fabricantes.

A seguir passa-se à explicação do efeito dos comandos principais, com seus nomes mais comuns e sinônimos.

FL, RETURN, NEWLINE, EOL, ENTER ou CR

Este é o comando mais utilizado; ele tem que ser usado obrigatoriamente para iniciar a ação de qualquer outro comando. Para emitir este comando basta pressionar uma única tecla, geralmente à direita do teclado e com qualquer dos nomes acima.

RUN

É o comando para a execução de um programa.

Só será efetivado após pressionada a tecla FL (ver sinônimos no comando anterior).

LIST

Lista o programa que está na memória.

Não esquecer de pressionar a tecla FL, após LIST.

BREAK

Interrompe o processamento que está sendo executado ou a lista que está sendo fornecida. É acionado por apenas uma tecla e não precisa ser seguida por FL.

CONT

Continua a execução ou a lista interrompida. Exige FL.

NEW ou **SCRATCH**

Este comando elimina todos os programas e valores de variáveis existentes na memória.

É necessário aplicá-lo para iniciar um novo programa, para evitar que se misturem instruções e dados anteriores. Deve ser seguido por FL.

RUBOUT ou **BACK SPACE**

É utilizado para corrigir erros. Produz um retrocesso e eliminação dos caracteres indicados por um cursor que se apresenta no vídeo. Não exige FL.

3.5 – **Instruções principais**

Uma instrução deve obrigatoriamente ser numerada e escrita em uma *linha de programa*; para indicar que a linha

terminou é exigido o uso da tecla de fim de linha, FL (ver sinônimos no parágrafo 3.4).

Uma *linha de programa* pode estender-se a várias linhas na tela.

As instruções podem, também, ser utilizadas como *comandos*, neste caso não podem ser numeradas.

Seguem-se as instruções principais.

REM

Permite incluir comentário, não produz efeito sobre o programa, mas o comentário será reproduzido quando se pedir lista, LIST, do programa. É um elemento muito útil para o entendimento dos objetivos de um programa, métodos utilizados, unidades, etc.

Exemplo:

```
10 REM PROGRAMA PARA CALCULO DE JUROS
```

PRINT

É uma instrução que determina ao computador a apresentação no vídeo de:

1) variáveis no seu valor atual,
2) resultados de expressões já efetuados os cálculos,
3) textos desde que fornecidos entre aspas.

Exemplo:

```
10 PRINT A
20 PRINT 3 * 2
30 PRINT "ESTES SAO OS VALORES"
```

Para executar este programa aplica-se o comando RUN, seguido por FL.

O resultado apresentado no vídeo será:

```
0
6
ESTES SAO OS VALORES
```

O zero é o valor atual de A, por não ter sido ainda atribuído valor algum; 6 é o valor da expressão indicada na linha 20 e o texto apresentado é somente o que estava entre aspas, na linha 30. Há computadores nos quais todas as variáveis devem obrigatoriamente ser pré-definidas; desta forma a linha 10 geraria mensagem de erro.

É possível apresentar também valores de variáveis, expressões ou textos em colunas, para isto basta separá-las por vírgulas.

Exemplo:

```
100 PRINT 5, 5/2, 5 * 5
```

Após o comando RUN, seguido de FL, o resultado apresentado será:

```
5          2.5          25
```

A separação por ponto e vírgula implica em não haver espaço entre uma e outra coluna. Usa-se ";", toda vez que se desejar apresentar as variáveis ou textos juntos.

Pode-se usar o PRINT sem nada escrito adiante para pular uma linha de tela, pois, quando o programa for executado, esta instrução deixa a linha em branco e passa à linha seguinte. Pode ser utilizado várias vezes para pular várias linhas.

Exemplo:

```
200 PRINT
210 PRINT
220 PRINT
```

Nesse caso vão ficar 3 linhas em branco.

LET

Define uma variável. Efetua também o cálculo de expressões. Não produz apresentação no vídeo, mas deixa a variável preparada para uso ou para apresentação, se solicitada por outra instrução.

Exemplo:

```
10 LET   R = 5
20 LET   L = (R-4) / 3
```

Após o comando RUN, seguido por FL, *nada será apresentado* no vídeo, mas estarão guardados na memória os valores de R = 5 e L = 0,33333. Para verificar isto basta aplicar PRINT como comando

```
PRINT R, L
```

agora pressionando FL, será apresentado:

```
5              .33333
```

Para a definição de variável alfanumérica, a fileira de caracteres deve ser colocada entre aspas.

Exemplo:

```
10 LET      A$ = "LIMITE"
```

Observações

1.ª) No cálculo das expressões o valor armazenado e apresentado é sempre o último, após efetuadas todas as operações indicadas à direita, dessa forma:

```
90  LET     X = 10
100 LET     X = X + 1
```

Na linha 90, X valia 10, na linha 100 a expressão indica que ele deve ser somado a 1, ou seja, após linha 100, X valerá 11.

2.ª) Enquanto não for atribuído um valor às variáveis numéricas, elas valem zero.

3.ª) Em muitas máquinas as variáveis podem ser definidas mesmo sem escrever a instrução LET.

Exemplo:

```
10    R = 5
20    L = (R – 4) / 3
```

GO TO

Esta instrução permite encaminhar para uma determinada linha, saindo da seqüência normal de linhas em sentido da numeração crescente.

Ela tem que ser seguida pelo número de linha para o qual se deseja saltar. É denominada desvio incondicional.

Exemplo:

```
500 GO TO 120
```

Fará com que a execução do programa ao chegar à linha 500 volte para a linha 120. Nota-se que é preciso alguma providência para evitar que o programa entre em um anel fechado, sem poder sair; adiante exemplifica-se o que pode ser feito para evitar o anel (anel é tradução do inglês *loop*).

Nota – também se pode escrever GOTO.

IF ... THEN

Encaminha para a instrução indicada após o THEN, se a condição entre o IF e o THEN for verdadeira; se for

falsa, o programa prossegue para a linha seguinte como se nada tivesse encontrado.

A condição referida pode ser uma relação de igualdade ou desigualdade, e os símbolos utilizados são:

Símbolo	*Significado*
=	igual a
<	menor do que
>	maior do que
< =	menor ou igual a
> =	maior ou igual a
< >	diferente de

Exemplo:

```
80 IF A < B  THEN GOTO 220
```

Isto é, se o valor atual de A for menor que o de B, a execução do programa, após a linha 80, seguirá a instrução após o THEN, saltando para linha 220; em caso contrário, ou seja, se A for igual ou maior do que B, após a linha 80, a execução irá para a linha 81, e assim por diante.

Em alguns computadores, a instrução GOTO, após o THEN, é subentendida, bastando escrever o número da linha para a qual se deseja saltar.

Exemplo:

```
80 IF A < B  THEN 220
```

STOP

Esta instrução determina parada no processamento, na linha em que for encontrada.

Para prosseguir pode ser usado o comando CONT.

Exemplo:

```
2000 STOP
```

O processamento cessa quando atingida a linha 2000.

Nota: A instrução END produz efeito similar e, nos computadores mais antigos, devia ser usada obrigatoriamente no fim dos programas.

GOSUB

É a instrução que permite o acesso a uma sub-rotina. Na linha em que for encontrada, a execução do programa salta para a linha indicada após a instrução.

No final da sub-rotina tem que haver uma instrução RETURN (não confundir com o comando RETURN), com a qual o programa volta à *linha seguinte* àquela onde foi encontrada a instrução GOSUB.

Para que o processamento não entre indevidamente na sub-rotina, ela deve ser precedida por uma instrução STOP, desta forma ela somente será acessível através de GOSUB.

Exemplo:

```
120 GOSUB 800
 .
 .
 .
790 STOP
800
 .
 .                          SUB-ROTINA
 .
950 RETURN
```

Ao chegar na linha 120, o processamento enviado pela instrução GOSUB salta para a linha 800, prosseguindo até a linha 950, onde encontra a instrução RETURN, voltando para a linha 121 e seguintes, até ser interrompido na linha 790 pela instrução STOP.

INPUT

É utilizada para a entrada de dados, durante o processamento. Ao chegar na linha onde está a instrução INPUT, o processamento é interrompido e na tela aparece um ponto de interrogação, ou algum outro símbolo, indicando que os dados devem ser fornecidos. O computador aguardará indefinidamente a entrada dos dados.

Após entrar com os dados, separados por vírgulas, é preciso pressionar FL.

Exemplo:

Se durante o processamento de um programa for encontrada:

```
50 INPUT X, Y, Z
```

no vídeo será apresentado:

```
?
```

A seguir entra-se com os dados, que ficarão à direita do ponto de interrogação,

```
? 6, 8, 30
```

Pressionando FL o processamento continua.

Em alguns computadores, a instrução INPUT só admite uma única variável. No caso do exemplo anterior seria ne-

cessário utilizar três vezes a instrução, uma para cada variável:

```
50 INPUT X
60 INPUT Y
70 INPUT Z
```

Há a possibilidade de apresentação de explicação sobre a variável pedida; para isto ela deve ser fornecida entre aspas, antes da variável e isolada por ponto e vírgula.

Exemplo: 100 INPUT "ENTRE COM O DIÂMETRO"; X

READ

É uma instrução que permite atribuir (ler) valores ou fileiras de caracteres a uma ou mais variáveis, valores estes que estejam contidos em uma instrução DATA. Essas duas instruções operam sempre em conjunto.

O valor atribuído é o que estiver na primeira instrução DATA encontrada, na ordem em que os valores estiverem colocados.

Exemplo:

```
50 READ X
80 READ Y, Z
```

DATA

É uma instrução que permite entrar com os valores (ou fileiras) a serem atribuídos a variáveis precedidas por uma instrução READ.

Quando o programa é executado, na primeira instrução READ encontrada são atribuídos os valores do primeiro DATA, seqüencialmente; se o programa tornar a passar pela mesma instrução READ, serão considerados os valores ainda não utilizados contidos em DATA, sempre na ordem em que foram colocados. Terminados os elementos deste DATA, a leitura prossegue no conteúdo do DATA seguinte.

Exemplo:
```
300 DATA 8
310 DATA 15, 4
```

RESTORE

Esta é a instrução que permite estabelecer o primeiro dos valores (ou fileiras) a ser atribuído quando se usa a instrução READ. Ela permite a leitura dos dados a partir do começo, novamente, ou a partir de uma outra linha cujo número esteja indicado adiante da instrução.

Exemplos:

1.º)

```
100 RESTORE
```

Conduz a leitura ao início do primeiro DATA encontrado em todo o programa.

2.º)

```
200 RESTORE 50
```

Conduz a leitura ao início do primeiro DATA da linha 50 em diante.

CLS

Esta instrução apaga tudo o que está apresentado na tela do vídeo, mas conserva o programa e os dados na memória.

FOR-NEXT

É um par de instruções que são usadas, em linhas diferentes, vindo em primeiro lugar o FOR e algumas linhas adiante o NEXT.

O objetivo destas instruções é permitir a repetição de um conjunto de operações com alteração de um elemento, em cada vez que o trecho de programa entre o FOR e o NEXT é repetido.

Por operações entende-se: cálculos, ordenamento, leitura, impressões, etc.

No capítulo seguinte o uso destas instruções é exemplificado.

3.6 — **Glossário das instruções e comandos**

O significado em português das instruções e comandos, embora não seja fundamental, auxilia a memorizar o efeito de cada um; segue-se uma lista dos mais comuns:

BACK SPACE	RETROCESSO (com apagamento)
BREAK	INTERROMPA
CLS (abrev. CLEAR SCREEN)	LIMPE A TELA
CONT	CONTINUE
DATA	DADOS
END	FIM
FOR	PARA
GOSUB	VÁ PARA SUB-ROTINA
GOTO	VÁ PARA
IF	SE (condição)
INPUT	ENTRE (entrar)
LET	SEJA
LIST	LISTE
NEW	NOVO (apague o antigo)
NEXT	SEGUINTE, PRÓXIMO ADIANTE
PRINT	IMPRIMA (escreva na tela)
READ	LEIA (atribua)
REM (abrev. REMARK)	OBSERVAÇÃO (comentário)
RESTORE	RESTAURE
RETURN	RETORNE
RUN	CORRA (execute)
STOP	PARE
THEN	ENTÃO

Capítulo 4

PROGRAMAÇÃO

Os comandos e instruções vistos no capítulo 3 possibilitam a elaboração de programas; neste capítulo, eles são apresentados em ordem de dificuldade crescente, para que se adquira, de forma gradativa, as noções básicas necessárias à programação.

Ao aplicá-los a um computador podem ocorrer erros de dois tipos:

A) os que são detectados pelo próprio operador;

B) os que são detectados pela máquina, com emissão de mensagem de erro.

Para a correção dos erros detectados pelo operador, utilizam-se dois métodos: se o erro for percebido imediatamente após digitar algo errado, pressiona-se a tecla de retrocesso corretivo (RUBOUT ou BACK SPACE), a qual apaga os caracteres à medida que for sendo aplicada; se o erro estiver numa linha já escrita, então o recurso é reescrevê-la corretamente, precedida pelo *mesmo número da linha* que se deseja substituir; desta forma, a linha que continha o erro é apagada e somente a nova será considerada.

Para a correção dos erros detectados pela máquina é preciso recorrer ao seu manual para interpretação da mensagem de erro emitida; isto é necessário porque não há concordância entre os diversos fabricantes.

Antes de colocar um programa novo é importante eliminar um eventual programa existente, o qual poderia misturar-se ao novo provocando erro; para isso aplica-se o comando NEW, seguido por FL. Note-se que, desta forma, o programa anterior e todos os valores das variáveis serão definitivamente perdidos.

Seguem-se alguns programas ilustrativos.

4.1 – Cálculo da média aritmética de dois números

4.1.A – Uma solução é utilizar PRINT como comando, isto é, sem o número de linha.

Assim a média entre 5 e 7 pode ser obtida com:

```
PRINT  (5 + 7) / 2
```

Após pressionar a tecla correspondente ao fim de linha (NEWLINE, RETURN), tem-se o resultado:

```
6
```

O computador executou primeiramente a soma, porque ela está indicada entre parênteses e, finalmente, efetuou a divisão.

4.1.B – Pode-se, também, atribuir os valores a variáveis, realizar o cálculo e, posteriormente, apresentá-lo no vídeo, como no programa seguinte:

```
10 LET      A = 5
20 LET      B = 7
30 LET      M = (A + B) / 2
40 PRINT    M
50 END
```

Com o comando RUN seguido de FL, tem-se no vídeo:

```
6
```

A média dos valores armazenados nas variáveis A e B foi calculada na linha 30 e seu resultado foi conservado em outra variável, M, por meio da instrução LET.

Alguns computadores requerem obrigatoriamente a instrução END para reconhecer o término do programa. Como são muito poucos os que possuem esta característica, a instrução END não será utilizada nos outros programas.

4.1.C – O cálculo de (A + B)/2 no programa adiante é feito após a instrução PRINT.

Toda a vez que se começa um novo programa, digita-se NEW e pressiona-se a tecla FL, fim de linha.

```
10 LET    A = 5
20 LET    B = 7
30 PRINT  (A + B) / 2
```

O resultado ao se executar o programa através do comando RUN, e a seguir FL, será o mesmo:

```
6
```

4.1.D – Outra alternativa é a leitura dos valores através da instrução READ.

```
10 READ   A, B
20 DATA   5, 7
30 PRINT  (A + B) / 2
```

Na instrução READ são lidos os valores de A e B contidos na instrução DATA.

Conforme já visto no capítulo 3, a instrução DATA pode estar em qualquer posição dentro do programa. No exemplo optou-se pela colocação logo após o READ.

4.2 – **Cálculo de expressão algébrica**

Apresentam-se a seguir três programas para cálculo da expressão

$$\frac{5X^2 + 4X}{2X + 3},$$

utilizando a instrução INPUT para a entrada do valor de X.

4.2.A – Realizando o cálculo da expressão através da instrução PRINT tem-se:

```
10 INPUT  X
20 PRINT  (5 * X ↑ 2 + 4 * X)/(2 * X + 3)
```

Após o comando RUN seguido de FL, aparece um ponto de interrogação como sinal de que o computador está aguardando os dados, neste caso o valor de X.

Depois de fornecer o valor de X, deve-se pressionar a tecla de fim de linha e, após o processamento (que pode ser muito rápido), o resultado aparecerá na tela.

Com X = 2, na tela será apresentado:

```
10 INPUT    X
20 PRINT    (5 * X ↑ 2 + 4 * X) / (2 * X + 3)
RUN
? 2          <-------- pergunta do computador
4            <-------- resposta do operador
```

Nota – Alguns computadores podem apresentar somente o resultado sem apresentar o programa, e, neste caso, para se saber qual o programa utilizado, deve-se usar o comando LIST.

4.2.B – Muitas vezes, em um programa mais extenso, é preciso guardar o valor de uma expressão. Isto é feito através da instrução LET, definida uma variável auxiliar.

```
10 INPUT    X
20 LET      Y = (5 * X ↑ 2 + 4 * X) / (2 * X + 3)
30 PRINT    Y
```

Neste caso em que o programa termina sem utilização posterior do resultado da expressão, a solução 4.2.A é a mais simples.

4.2.C – A instrução REM permite a colocação de comentários que ajudam a entender os procedimentos adotados no programa.

```
10 REM      ENTRADA DO VALOR DE X
20 INPUT    X
30 REM      CALCULO DA EXPRESSAO
40 LET      Y = (5 * X ↑ 2 + 4 * X) / (2 * X + 3)
50 REM      APRESENTACAO DO RESULTADO
60 PRINT    Y
```

No presente exemplo, os comentários seriam dispensáveis devido à simplicidade do problema, mas, à medida que a complexidade do programa aumenta, a colocação da instrução REM é de grande utilidade.

4.3 – **Tomada de decisão**

Seja o problema: dados dois números diferentes imprimir o maior deles.

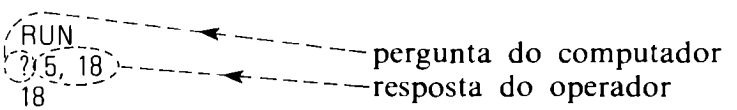
Toda a vez que é preciso tomar decisões que dependem do resultado de uma comparação, utiliza-se a instrução IF ... THEN.

Admitindo o caso que os números são armazenados nas variáveis N1 e N2, é necessário compará-los e se N1 for maior que N2, deve-se imprimir N1 ou em caso contrário, N2.

Programa

```
10 INPUT    N1, N2
20 IF       N1 > N2 THEN GOTO 50
30 PRINT    N2
40 STOP
50 PRINT    N1
```

Fornecendo os valores 5 e 18 para N1 e N2 respectivamente, depois do comando RUN o programa mostrará na tela o resultado 18, resumindo:



Durante a execução, o computador imprime o ponto de interrogação, devido à instrução INPUT da linha 10, pedindo os valores de N1 e N2.

Uma vez fornecidos estes valores, o programa prossegue; os valores são comparados na linha 20 e como N1, valendo 5, é menor do que N2, valendo 18, a próxima instrução execu-

tada é a da linha 30, ou seja, a impressão de N2 e posteriormente ocorrerá a parada em virtude da instrução STOP na linha 40.

A parada no processamento por uma instrução STOP vem normalmente acompanhada por uma mensagem do tipo: BREAK IN n, onde n é o número da linha na qual ocorreu a parada.

Então a apresentação no vídeo de forma completa será:

```
RUN
? 5, 18
18
BREAK IN 40
```

As mensagens ligadas com a instrução STOP, em outras versões, podem ser:

```
STOPPED AT n
STOP AT LINE n
```

ou outras, as quais devem ser interpretadas com auxílio do manual da máquina, pois os vários fabricantes usam mensagens diferentes.

Considere-se, agora, que sejam dados os números 210 e 11 para N1 e N2 respectivamente.

```
RUN
? 210, 11
210
```

Na linha 20 do programa é verificado que N1, valendo 210, é maior do que N2, valendo 11, e assim será executada a instrução logo após o THEN, ou seja, o programa saltará para a linha 50 por causa da instrução GOTO, sendo impresso o valor atual de N1, que é 210.

Observe-se que, nesse caso, não ocorrerá a mensagem relacionada com o STOP, pois o término do programa deu-se por não existir mais nenhuma instrução após a linha 50.

4.4 – Cálculo de prestações

Os programas que se seguem permitem determinar o valor das prestações a pagar a partir do preço à vista, taxa de juros e o número de parcelas.

4.4.A – Considerando forma de pagamento sem entrada, pode ser utilizada a seguinte fórmula:

$$P = V \frac{I}{1 - \frac{1}{(1 + I)^N}},$$

onde:

N – número de parcelas a pagar,
P – valor de cada uma das N parcelas,
I – taxa de juros aplicada,
V – valor à vista.

A expressão em BASIC equivalente é:

```
P = (V * I)  (1 – 1  (1 + I) ↑ N)
```

A solução consiste então na simples aplicação desta expressão, uma vez lidos os valores de N, V e I, para isso pode ser utilizado o programa:

```
10 READ    V, I, N
20 LET     P = (V * I) / (1 - 1 / (1 + I) ↑ N)
30 PRINT   P
40 DATA    10000.00 , 0.08 , 3
```

Nota-se, na linha 40, que as *vírgulas separam um dado do outro*; para a separação da parte decimal da fracionária dos números, usa-se *somente o ponto*, quando se escrevem programas.

Na instrução DATA estão estabelecidos os valores de V, I e N, ou seja, o programa determinará o valor das prestações que deverão ser pagas em 3 parcelas iguais, considerando uma taxa de juros de 8% (0,08) sobre um valor à vista de Cr$ 10 000,00.

Executando o programa obtém-se:

```
RUN
3880.34
```

4.4.B – Supondo, neste outro caso, que a primeira parcela seja paga como entrada, a nova fórmula é:

$$P = V \frac{I}{1 + I - \frac{1}{(1+I)^{N-1}}}.$$

```
10 READ    V, I, N
20 LET     P = (V * I) / (1 + I - 1 / (1 + I) ↑ (N - 1) )
30 PRINT   P
40 DATA    10000.00 , 0.08 , 3
```

O novo programa com os mesmos valores do anterior fornece como resultado:

```
RUN
3592.9
```

onde o valor da prestação é Cr$ 3 592,90. Note-se que os espaços entre números foram perdidos e o número de algarismos decimais após o ponto é incontrolável, podendo ser maior ou menor do que as tradicionais duas casas de centavos. No capítulo 6 é apresentada uma instrução que permite resolver este problema.

4.4.C – Admitindo que se precise fazer uma tabela onde o número de prestações seja variável, como mostrado na tabela abaixo, é necessário efetuar o cálculo da fórmula para os diversos valores de N.

Número de parcelas	*Valor de cada parcela*
2	p_2
3	p_3
4	p_4
5	p_5
6	p_6
7	p_7

Na maioria dos casos em que se tem uma repetição de cálculos ou de procedimento, utiliza-se o par de instruções FOR – NEXT.

Supondo ainda que as parcelas desejadas são para N variando de 2 até 7 e utilizando a fórmula para primeiro pagamento a vista, o programa fica:

```
10 READ    V, I
20 DATA    10000.00 , 0.08
30 FOR     N = 2 TO 7
40 LET     P = (V * I) / (1 + I - 1 / (1 + I) ↑ (N - 1) )
50 PRINT   N, P
60 NEXT    N
```

São lidos apenas o valor à vista e a taxa de juros, porque o número de prestações agora é variável.

As instruções contidas nas linhas 40 e 50 serão executadas para N igual a 2, 3, 4, ... até 7, onde a malha FOR – NEXT termina e o programa também, pois não há mais instruções após a linha 60.

A tabela mostrada no vídeo será:

```
2    5192.31
3    3592.9
4    2795.56
5    2319.04
6    2002.92
7    1778.45
```

Outras tabelas variando a taxa de juros ou o valor à vista podem ser feitas com auxílio de FOR – NEXT.

Note-se que, ao se utilizar o par FOR – NEXT, o FOR é aplicado em uma linha e o NEXT em linha posterior. A aplicação de FOR sem o correspondente NEXT em alguma linha seguinte gera mensagem de erro.

4.5 – **Fileiras de caracteres**

As fileiras de caracteres são utilizadas toda vez que é preciso apresentar resultados numéricos acompanhados por um texto, fazer programa conversacional ou manipular

informações não-numéricas. Seguem-se programas nos quais se fazem essas aplicações das *fileiras*, ou seja,de *seqüências* de caracteres alfanuméricos.

4.5.A — *Apresentação de valor numérico em conjunto com explicação do significado*

O programa já visto para o cálculo da média aritmética pode ser aperfeiçoado para:

```
10 READ    X, Y
20 DATA    5, 7
30 LET     M = (X + Y) / 2
40 PRINT"A MEDIA ENTRE X E Y E IGUAL A " ; M
```

De forma que ao se executar o programa, a apresentação no vídeo será:

```
RUN
A MEDIA ENTRE X E Y E IGUAL A 6
```

O texto para ser apresentado no vídeo foi colocado adiante da instrução PRINT, na linha 40, delimitado com aspas. Além do texto foi mostrado o valor da média, a qual estava armazenada na variável M.

A separação com ponto e vírgula entre o texto e a variável fez com que estes fossem impressos sem separação, isto é, sem salto de coluna ou linha.

4.5.B — *Programa conversacional*

Neste tipo de programa estabelece-se um diálogo entre o operador e a máquina.

Ainda o mesmo programa para cálculo da média aritmética pode ser passado para o estilo conversacional, como segue:

```
10 PRINT "FORNECER O PRIMEIRO NUMERO, X"
20 INPUT X
```

```
30 PRINT "FORNECER O SEGUNDO NUMERO, Y"
40 INPUT Y
50 LET M = (X + Y) / 2
60 PRINT "A MEDIA ENTRE X E Y E IGUAL A"; M
```

Neste caso, ao se aplicar o comando RUN (e FL), tem-se:

```
RUN
FORNECER O PRIMEIRO NUMERO, X
?
```

Então o operador deve teclar o primeiro número, por exemplo 5, e após acionar a tecla de fim de linha, aparecerá:

```
RUN
FORNECER O PRIMEIRO NUMERO, X
? 5
FORNECER O SEGUNDO NUMERO, Y
?
```

Uma vez fornecido o outro número, tem-se no vídeo:

```
RUN
FORNECER O PRIMEIRO NUMERO, X
? 5
FORNECER O SEGUNDO NUMERO, Y
? 7
A MEDIA ENTRE X E Y E IGUAL A 6
```

Neste exemplo, o operador escolheu os números 5 e 7, resultando a média 6.

4.5.C — *Manipulação de informações não-numéricas*

Seja a tarefa de colocar duas palavras na ordem alfabética. Deve-se ler as palavras, armazená-las nas variáveis alfanuméricas e imprimi-las na ordem alfabética segundo o resultado de uma comparação.

Na maioria dos computadores têm-se que, se A, B e C forem definidos como fileiras, A < B (A menor do que B,

significa A precede B), B < C (B menor do que C, significa B precede C) e assim, sucessivamente, na ordem alfabética.

Quando se comparam palavras com várias letras, a precedência é estabelecida pela primeira letra; se houver coincidência da primeira letra, então a precedência é feita pela segunda letra e assim por diante.

Exemplo:

São relações verdadeiras, entre fileiras:

```
OURO       < PLATINA
ATLÂNTICO  < INDICO
AZUL       > AMARELO
```

Para ordenar alfabeticamente as palavras PLATINA e OURO, pode ser utilizado o programa:

```
10 READ A$, B$
20 DATA "PLATINA", "OURO"
30 IF A$ > B$ THEN GOTO 70
40 PRINT A$
50 PRINT B$
60 STOP
70 PRINT B$
80 PRINT A$
```

Ao se executar o programa, obtêm-se:

```
RUN
OURO
PLATINA
```

Note-se que a inversão, na linha 20, das palavras PLATINA e OURO, não alteraria o resultado do programa.

Lembre-se que o caractere cifrão é obrigatório, após as variáveis alfanuméricas.

4.6 – Despesa com cigarros em um ano

O objetivo deste programa é estabelecer um diálogo com o operador, e após perguntar se ele fuma e quanto custam os cigarros que ele consome, informá-lo de sua despesa anual ou ainda cumprimentá-lo por ter um comportamento normal, conforme a Organização Mundial de Saúde.

```
 10 PRINT "VOCE FUMA?"
 20 PRINT "RESPONDA-ME SIM OU NAO, POR FAVOR"
 30 INPUT A$
 40 LET N$ = "NAO"
 50 LET S$ = "SIM"
 60 IF A$ = N$ THEN GOTO 200
 70 IF A$ = S$ THEN GOTO 100
 80 PRINT "POR FAVOR, APENAS SIM OU NAO, PARA QUE
    EU POSSA ENTENDER"
 90 GOTO 10
100 REM D E A DESPESA ANUAL
110 REM B E O PRECO DE 20 CIGARROS
120 PRINT "QUANTOS CRUZEIROS VOCE PAGA
    POR 20 CIGARROS?"
130 INPUT B
140 REM C E O NUMERO DE CIGARROS CONSUMIDOS POR DIA
150 PRINT "QUANTOS CIGARROS VOCE FUMA POR DIA ?"
160 INPUT C
170 LET D = 365 * (B/20) * C
180 PRINT "SUA DESPESA DIRETA ANUAL
    COM CIGARROS E "; D;" CRUZEIROS"
190 STOP
200 PRINT "PARABENS, SEU COMPORTAMENTO E NORMAL"
```

Capítulo 5

COMANDOS, INSTRUÇÕES E FUNÇÕES AVANÇADOS

Neste capítulo, são apresentados comandos, instruções e funções que correspondem a um estágio mais avançado de utilização das variantes da linguagem BASIC e alguns não são disponíveis em todos os computadores. Considere-se, ainda, que o assunto não fica esgotado.

São elas:

COMANDOS: LIST n – m
LLIST
SAVE
LOAD
DELETE
EDIT
RUN parcial
COPY
AUTO
RENUMBER
CTRL ou CONTROL

INSTRUÇÕES: LINHA COM MÚLTIPLAS INSTRUÇÕES
LET subentendido
PRINT ponto de interrogação
PRINT USING
PRINT TAB
LPRINT
FOR... TO... STEP
ELSE
RANDOM, RANDOMIZE
CLEAR

DIM
MAT INPUT
MAT READ
MAT PRINT
MAT operações
DEF FN
SET ou PLOT
RESET ou UNPLOT
ON... GOTO
ON... GOSUB

FUNÇÕES: LEN
VAL
STR$
MID$

5.1 – **Comandos avançados**

LIST n – m

Lista o programa que estiver na memória desde a linha n até a linha m.

Há variantes:

LIST n–, lista de n em diante até o fim.
LIST –m, lista desde o começo até a linha m.

Exemplos:

LIST 500 – 650
Lista desde a linha de número 500 até a de número 650.
LIST –100
Lista até a linha de número 100.
LIST 400 –
Lista a partir da linha número 400 até o fim.

LLIST

Produz a lista do programa através da impressora.

Pode ser aplicado entre linhas, analogamente ao caso anterior, aplicando-se para isso LLIST n – m.

Também pode ser aplicado a partir de uma linha n, ou ainda até uma linha m.

Se for aplicado este comando e não houver impressora ligada, o computador fica aguardando indefinidamente; para prosseguir deve ser utilizada a tecla RESET.

Exemplo:

LLIST

Imprime toda a lista do programa.

LLIST 40 – 120

Imprime desde a linha 40 até a 120.

LLIST – 200

Lista, através da impressora, até a linha 200.

LLIST 200 –

Imprime a lista do programa a partir da linha 200 até o fim.

SAVE

Grava o programa da memória, em fita ou disco magnético.

Pode também atribuir um nome ao programa; neste caso, SAVE deve ser seguido do nome. O nome, em algumas máquinas, deve ser uma única letra, em outras pode ser com várias letras; é preciso consultar o manual de cada máquina para saber o que é permitido como nome, em cada uma.

É preciso cuidado porque, se a gravação for feita sobre um programa ou dados existentes, os anteriormente gravados serão perdidos.

Exemplo:

SAVE "AGENDA"

Para a gravação em fita deste programa, aciona-se o gravador e posteriormente a tecla de fim de linha, FL; desta forma, o programa que estiver na memória será transferido para a fita e receberá o nome de AGENDA.

Observação: Em algumas máquinas, o comando para gravação em fita cassete é CSAVE.

LOAD

Através deste comando é que se transfere (carrega) um programa de uma fita ou disco magnético para a memória do computador. Em alguns equipamentos deve ser utilizado o LOAD seguido do nome do programa a ser transferido. Este comando pode ser uma letra apenas ou várias, conforme visto para a instrução SAVE.

Observações:

A) Em algumas máquinas, o comando para transferência (carga) de programa de fita cassete para a memória do computador é CLOAD.
B) Na aplicação deste comando é necessário ler atentamente o manual do aparelho utilizado, pois as diferenças de um para outro são grandes.

DELETE n – m

É o comando utilizado para a eliminação de uma linha ou de um conjunto de linhas.

Após o comando indicam-se a linha n, primeira a ser apagada, e a linha m, última a ser apagada. Pode ser usada apenas n para apagar somente esta. Havendo o hífen, se n for omitido, é considerado 1; se m não for escrito, é apagado desde n até a última linha do programa.

Exemplos:

a) DELETE 500
Elimina a linha de número 500.

b) DELETE 100 – 500
Elimina todas as linhas desde 100 até a 500.

c) DELETE – 90
Elimina desde a linha de número 1 até a de número 90.

d) DELETE 700 –
Elimina todas as linhas desde a de número 700 até a última linha de programa.

EDIT

O objetivo deste comando é permitir alterações em linhas já escritas do programa trazendo-as para ser editadas novamente.

Em alguns computadores, o comando é EDIT n, onde n é o número da linha que se pretende modificar ou corrigir: em outros há cursor de edição que pode ser deslocado para linha acima ou abaixo e para a direita ou esquerda.

Uma vez trazida a linha para edição, a aplicação das correções é bastante variada, devendo ser consultado o manual da máquina específica.

A principal utilidade do EDIT é dispensar que se escreva a linha inteira onde se aplica a correção.

Após completadas as alterações pressiona-se a tecla de fim de linha, FL, então a linha corrigida é incorporada ao programa, substituindo a antiga.

RUN parcial

Para executar um programa que esteja na memória, mas apenas a partir de uma certa linha, aplica-se o comando

RUN – n, onde n é o número da linha onde se pretende iniciar a execução. É necessário lembrar que os resultados obtidos com somente uma parte do programa costumam ser totalmente diferentes dos usuais; a utilidade deste comando está na verificação do funcionamento de certos trechos do programa e sua aplicação exige que se pondere cuidadosamente o resultado esperado, lembrando ainda que as variáveis definidas antes da linha n serão limpas por meio de uma instrução CLEAR, aplicada automaticamente pelo computador, quando se utiliza a instrução RUN – n. A instrução CLEAR é descrita adiante, no parágrafo 5.2.

Exemplo:

RUN – 200

Executará o programa a partir da linha de número 200 até o final. As variáveis definidas nas linhas anteriores a 200 serão consideradas nulas (se numéricas) ou brancos (se fileiras alfanuméricas), ou ainda indefinidas em algumas máquinas mais raras.

COPY

Aciona a impressora de forma a tirar uma cópia impressa de uma imagem estacionária na tela do vídeo.

AUTO

Este comando produz a numeração automática de linhas, começando por 10 e com passo também 10. Cada vez que é pressionada a tecla de fim de linha, FL, já surge o novo número de linha.

Para interromper a numeração automatica usa-se o comando BREAK em algumas máquinas, em outras pressionam-se simultaneamente a tecla CTRL e a tecla C.

Há possibilidade de começar em uma linha diferente de 10 e com passo escolhido, fazendo por exemplo AUTO 100 STEP 20. Neste caso, as linhas serão numeradas a partir de 100 e com passo 20, automaticamente.

RENUMBER

Renumera automaticamente as linhas de um programa presente na memória, começando pela linha 10 e prosseguindo com passo também 10.

É possível começar em uma outra linha especificada e com passo, STEP, escolhido.

Sua principal função é tornar a versão final de um programa mais elegante e de entendimento mais fácil, graças à numeração com passo fixo.

São poucos os microcomputadores que dispõem deste recurso.

CTRL ou CONTROL

É acionado, geralmente, através de uma tecla em conjunto com outra, produzindo diferentes efeitos conforme a segunda tecla utilizada.

CTRL C, onde estas duas teclas são pressionadas em conjunto, produz uma parada na execução de um programa. Esta interrupção será necessária quando o programa tiver entrado em um anel fechado (em inglês, *loop*) no qual ficaria indefinidamente. Esses anéis fechados originam-se de algum erro ou imprevisão na elaboração de um programa. É preciso, então, interromper a execução através de um comando que tenha prioridade sobre a seqüência de instruções do programa; é somente o CTRL C ou o BREAK (quando houver) que têm esta capacidade, pois todos os outros comandos estarão inoperantes durante a execução do programa.

Outra situação em que este comando é útil ocorre quando a execução de um programa torna-se muito demorada e pretende-se desistir do processamento.

Após a aplicação deste comando, o processamento é interrompido e emitida a mensagem: BREAK IN LINE N.

Em seqüência a um comando de edição, EDIT, usam-se CTRL e outra tecla, para vários efeitos; não é possível descrevê-los dada a sua variedade de um equipamento para outro.

5.2 – **Instruções avançadas**

Neste parágrafo são apresentadas algumas instruções mais raras, as quais não são disponíveis em todos os computadores ou apresentam diferenças, às vezes muito grandes, de um tipo de máquina para outro.

Também são tratados alguns detalhes de instruções principais, abreviações e formas condensadas para escrever programas.

LINHA COM MÚLTIPLAS INSTRUÇÕES

Para reduzir o número de linhas de um programa, pode-se utilizar várias instruções em uma mesma linha, separando-as por dois pontos.

Exemplo:

Um programa para atribuir valores a duas variáveis, efetuar seu produto e apresentá-lo no vídeo, pode ser escrito em uma única linha:

```
10 LET X = 14 : LET Y = 27 : LET A = X * Y : PRINT A
```

Nesta linha estão quatro instruções, que serão executadas da esquerda para a direita. Ao executar este programa de apenas uma linha, será apresentado o resultado

378,

que é o produto de 14 por 27.

LET subentendido

A instrução LET, em muitos computadores, mas não em todos, não precisa ser escrita. Simplifica-se, desta forma, a tarefa da digitação, ganhando-se tempo e reduzindo-se a ocupação de memória.

Exemplo:

```
10 X = 14
20 Y = 27
30 A = X * Y
40 PRINT A
```

Ou reunindo com a técnica de múltiplas instruções por linha:

```
10 X = 14 : Y = 27 : A = X * Y : PRINT A
```

PRINT representado por ponto de interrogação

A instrução PRINT é uma das mais utilizadas e, por isso, ela é freqüentemente substituída por um único símbolo, o ponto de interrogação.

Exemplos:

1.º) Em vez de

```
10 PRINT A
```

pode ser escrito

```
10 ? A
```

2.º) O exemplo da instrução anterior ficaria ainda mais compacto:

```
10 X = 14 : Y = 27 : A = X * Y : ? A
```

Em alguns computadores, ao se pedir uma lista do programa ou edição da linha onde foi utilizado o símbolo ponto de interrogação, a lista será fornecida no vídeo ou impressa

com a palavra PRINT e não mais "?", como estava na lista original.

É importante *não confundir* o "?" que se fornece ao computador, *significando PRINT*, com o "?" que o computador fornece ao pedir dados, quando é encontrada uma instrução INPUT e que significa: *entre com os dados.*

PRINT USING

Esta é uma variante da instrução PRINT pela qual se pode determinar o formato no qual serão apresentados os caracteres.

No caso de resultados numéricos, pode-se escolher quantas casas após o ponto decimal se deseja, e os arredondamentos ou preenchimentos com zeros são automáticos. Também se pode escolher quantas casas antes do ponto decimal deve haver, sendo preenchidos com brancos os espaços vazios.

Para determinar o formato é utilizado o símbolo #, que em inglês significa *número* ou *n.º*.

Exemplo:

```
10 LET A = 2000 / 3
50 PRINT USING "# # # # . # #" ; A
```

Este programa produzirá o resultado:

```
666.67
```

onde o resultado está apresentado com as duas casas decimais pedidas na instrução e o arredondamento automático da última.

A notação exponencial também pode ser imposta através desta instrução, com o símbolo de exponenciação repetido quatro vezes.

Exemplo:

```
100 PRINT USING "# # . # #↑↑↑↑" ; 2326439
```

e o resultado será:

2.33 E + 06

Um cifrão também pode ser escrito na frente dos números, bastando para isso colocar $$ antes do símbolo de número.

Exemplo:

```
50 J = .08 * 70000.00
100 PRINT USING "$$###.##" ; J
```

e o resultado será:

$ 5600.00

A separação por vírgulas a cada grupo de três algarismos também pode ser efetuada; para isso basta colocar uma vírgula entre os símbolos # , em qualquer posição, antes do ponto decimal.

Exemplo:

```
10 A = 123456
20 PRINT USING "#, #####"; A
```

e o resultado será:

123,456

Note-se que a vírgula está como é utilizada em inglês, isto é, como simples separação dos grupos de números e não indica a separação da parte fracionária.

O número acima deve ser lido: cento e vinte e três mil, quatrocentos e cinqüenta e seis.

PRINT TAB

Para a elaboração de tabelas pode-se usar PRINT TAB (X), onde o argumento X é o número da coluna a partir da qual começará a impressão.

Exemplo:

```
10 LET   A$ = "JOGADOR A"
20 LET   B$ = "JOGADOR B"
30 PRINT TAB (5) A$  TAB (20)  B$
```

Como resultado serão apresentadas no vídeo as fileiras correspondentes a A$ e B$, começando nas colunas 5 e 20 respectivamente:

JOGADOR A JOGADOR B

O argumento da função TAB pode ser uma expressão cujo valor será calculado pelo computador e arredondado para inteiro e servirá para indicar a coluna em que deve ser efetuada a apresentação do resultado.

LPRINT

Produz o mesmo efeito que PRINT, porém com acionamento da impressora.

Em algumas máquinas não existe esta instrução e seu efeito é obtido pelo mesmo PRINT, quando a impressora estiver ligada.

FOR ... TO ... STEP

É uma variante do par de instruções FOR – NEXT, já visto no capítulo 3. A diferença consiste em que, neste caso, se pode usar *PASSO* (em inglês, *STEP*) variável. Quando o passo não é especificado, é admitido como sendo 1.

Quando se usa passo fracionário, pode ser exigido colocar o valor limite superior ligeiramente acima para absorver erros de arredondamento, os quais poderiam levar à supressão do último valor.

Exemplo:

Seja calcular a área de um círculo de diâmetro variável, desde 80 até 120 mm, com passo 5 mm, em metros quadrados. Pode-se utilizar o seguinte programa:

```
10 FOR   D = .08 TO .121 STEP .005
20 PRINT D,  3.1416 * D ↑ 2/4
30 NEXT  D
```

e o resultado apresentado será:

```
.08          .00502656
.085         .005674515
.09          .00636174
.095         .007088235
.1           .007854
.105         .008659035
.11          .00950334
.115         .010386915
.12          .01130976
```

Neste exemplo podemos observar que, na linha 10, embora o maior valor para o diâmetro seja .12 m, foi colocado um valor um pouco acima .121, menos do que um passo acima, como provisão para erros de arredondamento. De fato, executando o programa uma outra vez, porém com valor final mudado para .12, a última linha não chegou a ser apresentada.

Na linha 20 a expressão não necessita parênteses, pois a exponenciação é executada primeiro, conforme a prioridade já apresentada no capítulo 2.

ELSE

Opera no conjunto IF... THEN... ELSE, oferecendo então duas alternativas, como também havia com apenas IF... THEN, todavia decorre alguma simplificação com o seu uso.

ELSE em inglês significa: em caso contrário; desta forma o conjunto apresentado IF... THEN... ELSE significa SE... ENTÃO... CASO CONTRÁRIO.

Exemplo:

Seja um trecho de programa:

```
100 IF L < M  THEN PRINT "M MAIOR" ELSE
      PRINT "L MAIOR OU IGUAL"
```

Ao executar o programa, imediatamente após a linha 100, uma das duas alternativas será escrita. Sem a instrução ELSE, seria necessário uma linha a mais de programa.

RANDOM

Esta instrução é auxiliar da função pseudo-aleatória RND e permite que ela produza valores aleatórios (ao acaso, em inglês, *random*). Pode também ser escrita RANDOMIZE ou RAND, em algumas versões.

Ela deve ser usada como uma instrução que precede o cálculo da função pseudo-aleatória RND. O exemplo seguinte mostra uma aplicação ao jogo de dados.

Exemplo:

```
10 RAND
20 LET D = RND
30 PRINT INT ( (D *6) + 1)
```

A cada execução será fornecido um número ao acaso, entre 1 e 6, como ocorreria jogando um dado.

CLEAR

Produz a eliminação de todos os valores atribuídos às variáveis; as variáveis numéricas são levadas a zero e as fi-

leiras são reduzidas a brancos. O programa é conservado sem alteração.

É conveniente aplicá-lo com cuidado, pois, em alguns raros computadores, CLEAR tem outro significado; é idêntico a NEW, apagando variáveis e também programa.

DIM

É uma instrução utilizada para reservar espaço quando se trabalha com matrizes, ou seja, com variáveis simplesmente ou multiplamente indexadas.

DIM pode ser considerada uma forma abreviada da palavra DIMENSÃO e que tem o sentido de número de valores (ou fileiras) a ser armazenado.

Nas variáveis com apenas um índice, ou vetores, declara-se apenas uma dimensão.

Exemplo:

```
DIM    L (12)
```

Significa que a variável L tem, no máximo, 12 componentes, L_1, L_2, ... L_{12}.

Para variáveis com dois índices, ou matrizes, constitui-se uma tabela de N linhas e M colunas, e a instrução deve conter estas dimensões máximas.

Exemplo:

Para reservar espaço na memória, quando se pretende operar com uma tabela de nome G com 6 linhas e 6 colunas, faz-se:

```
20   DIM    G    (6, 6)
```

Se a declaração de DIM contiver mais linhas ou colunas do que o necessário, o programa poderá funcionar em alguns casos, mas haverá maior utilização de memória, o que poderá

ser prejudicial. Por outro lado, se a declaração DIM considerar menos linhas ou colunas do que as que serão utilizadas, será emitida mensagem de erro.

Para as variáveis alfanuméricas, isto é, fileiras, a instrução fixa o *número máximo de fileiras*.

Exemplo:

```
30 DIM    F$ (25)
```

Significa que o vetor terá, no máximo, 25 fileiras.

Nota: Em alguns poucos casos a declaração DIM pode ser omitida, entretanto *é conveniente usá-la sempre*, antes de qualquer operação com matrizes.

MAT INPUT

Nos computadores que dipõem desta instrução, a entrada de dados para uma matriz fica muito simplificada.

A instrução DIM deve ser usada em linha anterior àquela onde se utiliza MAT INPUT.

A entrada de dados com esta instrução é feita de maneira análoga à da instrução INPUT, mas, em lugar de um único valor numérico ou uma única fileira, devem ser fornecidos todos os elementos na ordem de linha por linha e de primeira à última coluna.

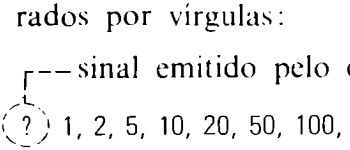
Encontrada uma instrução MAT INPUT durante a execução do programa, o processamento é interrompido e no vídeo ou impressora é apresentado um ponto de interrogação, significando que os elementos da matriz devem ser fornecidos, completando-se as linhas.

Exemplo:

```
100 DIM   C (4, 3)
200 MAT   INPUT C
```

Na linha 100, a instrução DIM estabelece que a matriz tem 4 linhas e 3 colunas.

Na linha 200, o computador pára o processamento e apresenta um ponto de interrogação; em seguida, o operador deve fornecer os elementos da matriz, separados por vírgulas:



Após pressionar FL, estarão atribuídos valores aos elementos da matriz. Neste caso, a matriz C é:

1	2	5
10	20	50
100	200	500
1000	2000	5000

MAT READ

É uma instrução para entrada de dados que são elementos de uma matriz. Esses dados devem estar em uma instrução DATA e serão considerados linha por linha, a partir da primeira.

Deve ser precedida por uma instrução DIM.

É a instrução matricial correspondente à escalar READ.

Exemplo:

```
10 DIM L (3, 3)
20 MAT READ L
30 DATA 1, 2, 3, 4, 5, 6, 7, 8, 9
```

A partir da linha 30, a matriz L passa a ser:

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix}$$

MAT PRINT

Com esta instrução obtém-se, no vídeo ou na impressora, a apresentação da matriz completa.

Exemplo:

Adicionando-se a linha

```
100 MAT PRINT L
```

ao programa apresentado no exemplo da função MAT READ, obtém-se a impressão ou apresentação no vídeo de:

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

MAT (operações)

É possível efetuar diretamente operações com matrizes, em alguns computadores de maior porte.

Assim têm-se:

ADIÇÃO DE MATRIZES

```
100 MAT A = B + C
```

onde A, B e C são matrizes de mesmas dimensões, N × M, declaradas anteriormente em uma DIM. A matriz A será obtida como a soma das matrizes B e C.

PRODUTO DE MATRIZES

200 MAT A = B * C

onde B é uma matriz M × N e C é N × M, para que seja possível o produto.

INVERSÃO DE MATRIZ

300 MAT A = INV (B)

A matriz B deve ser quadrada e não singular.

TRANSPOSIÇÃO DE MATRIZ

400 MAT A = TRN (B)

As dimensões declaradas em DIM anterior devem ser M × N e N × M para permitir a transposição.

MATRIZ IDENTIDADE

500 MAT A = IDN

Transforma uma matriz quadrada A na matriz identidade, com a mesma dimensão.

MATRIZ NULA

600 MAT A = ZER

Anula todos os elementos da matriz A.

DEF FN

Esta instrução permite definir funções novas e é útil quando se vai fazer uso freqüente de uma certa expressão.

A função deverá receber um nome, por exemplo W, e o seu argumento, ou seja, a variável, outro nome, por exemplo X. A forma fica então:

DEF FN W (X) = (expressão)

Quando for necessário usar a expressão, basta chamá-la por FNW (X).

SET ou PLOT

Produz o desenho de um pequeno retângulo para representar um ponto de coordenadas x e y, na tela do vídeo.

As coordenadas podem ser definidas por variáveis, constantes ou expressões. Quando o resultado não for inteiro, será considerada apenas a parte inteira.

O eixo x é freqüentemente numerado de 0 a 127 ou de 0 a 63 e tem início no lado esquerdo da tela, vista de frente.

O eixo y dispõe de 0 a 47, 43 ou ainda outro número de posições. Em algumas máquinas, o eixo x tem início no lado de cima do vídeo, em outras no lado de baixo, neste último caso, orientado como em geometria analítica.

O argumento deve ser fornecido entre parênteses, geralmente.

Exemplo:

```
500 SET (80, 20)
```

Localizará um ponto (retângulo) com coordenadas x = 80 e y = 20.

RESET ou UNPLOT

Esta instrução opera como corretivo da SET ou PLOT, com as mesmas características, quanto ao argumento.

Seu efeito é apagar os elementos das coordenadas especificadas.

ON... GOTO

Produz desvio da execução do programa para uma das linhas indicada adiante de GOTO.

Sua forma geral é:

ON L GOTO N_1, N_2, N_3...

onde L é uma variável que pode assumir valores 1, 2, 3..., e N_1, N_2, N_3 são números de linha.

Quando L = 1, a execução do programa salta para a linha N_1, que é a primeira indicada adiante do GOTO; quando L = 2, salta para a linha N_2, que é a segunda linha após GOTO, e assim por adiante.

Sua utilidade é reduzir o número de linhas de um programa e simplificar a digitação.

Exemplo:

Considere-se a linha:

```
500 ON J GOTO  1500, 4000, 6000
```

Quando ao passar pela linha 500 e sendo J = 1, a execução saltará para linha 1500; para J = 2, a execução prosseguirá na linha 4000, e se J = 3, o salto será para a linha 6000.

Note-se que, sem se utilizar esta instrução, o mesmo resultado poderia ser obtido utilizando *três vezes* as instruções IF... THEN GOTO, o que teria o inconveniente de produzir um programa mais longo.

Em lugar da variável L da forma geral, também pode ser colocada uma expressão cujo resultado seja inteiro, e a seguir utilizado da maneira anterior.

ON... GOSUB

Produz o encaminhamento para uma das sub-rotinas, cujo número de linha inicial esteja indicado adiante de GOSUB.

Sua forma geral é:

ON L GOSUB N_1, N_2, N_3...

onde L é uma variável ou expressão que pode assumir os valores inteiros 1, 2, 3.... Para L = 1 é escolhida a primeira sub-rotina, iniciada em N_1 ; para L = 2, a segunda sub-rotina com início em N_2, e assim por adiante.

Ao final de cada sub-rotina haverá uma instrução RETURN, a qual produzirá o retorno à linha seguinte àquela onde foi encontrada a instrução ON... GOSUB.

Exemplo:

Ao ser atingida a linha seguinte:

100 ON G GOSUB 800, 2000

se G = 2, o programa se encaminha para a sub-rotina que se inicia na linha 2000; esta sub-rotina obrigatoriamente terminará com uma instrução RETURN, que conduzirá o programa à linha de número seguinte ao número 100.

A utilidade principal desta instrução é permitir que se produzam programas mais compactos.

5.3 — **Funções avançadas**

LEN

É uma função que fornece o *número de caracteres* de uma fileira, incluídos os brancos ou espaços e os sinais de pontuação.

O seu argumento, ou seja, a fileira, deve ser colocado entre parênteses, adiante de LEN.

Ela é freqüentemente utilizada em conjunto com as instruções PRINT, LET e IF... THEN.

LEN é abreviação de LENGTH (em inglês significa comprimento), que, no caso, é medido em espaços ou colunas utilizadas.

Exemplos:

A)
```
10 PRINT LEN ("LEMA")
```

e após a execução será fornecido o resultado:

```
4
```

que é o número de caracteres da fileira LEMA.

B)
```
20 PRINT LEN ("CR$ 200 000")
```

e o resultado será:

```
11
```

onde os espaços em branco também estão incluídos no número de caracteres da fileira.

C)
```
100 LET A$ = "1,2 E 3"
110 PRINT LEN (A$)
```

e o resultado será:

```
7
```

pois estão incluídas a vírgula e os espaços em branco.

VAL

Esta função pode ser aplicada a fileiras cujos caracteres são números, somente até um ponto fracionário decimal, e a espaços em qualquer posição, sendo estes últimos igno-

rados. O resultado fornecido é a fileira de caracteres transformada em número.

O seu argumento deve ser fornecido entre parênteses.

Ao apresentar o número é reservado um espaço à esquerda para seu sinal algébrico, o qual se for negativo é colocado à esquerda, e se for positivo é omitido.

Exemplos:

```
A)  10 LET     A$ = "  123.5"
    20 PRINT   VAL (A$)
    30 LET     B$ = "  123.5  "
    40 PRINT   VAL (B$)
    50 LET     C$ = "  12 3 . 5 "
    60 PRINT   VAL (C$)
```

Após a execução deste programa, obtêm-se:

```
123.5
123.5
123.5
```

Note-se que os espaços foram ignorados ao aplicar VAL.

```
B)  100 LET    A$ = "223"
    110 PRINT  A$
    120 PRINT  VAL (A$)
```

e o resultado será:

```
223
 223
```

Note-se o deslocamento para a direita de VAL (A$), como espaço reservado para a colocação de sinal algébrico.

```
C)  200 PRINT  VAL ("52 00")
    220 PRINT  VAL ("– 4380")
```

Resultado:

```
 5200
-4380
```

STR$

Leia-se STR cifrão, STR fileira, ou STR seqüência.

Converte *um número*, fornecido como seu argumento, *em uma fileira* de caracteres alfanuméricos.

Uma vez efetuada a conversão, a fileira só poderá ser manipulada como fileira e não mais como número.

A soma de duas fileiras é uma fileira que se obtém por junção das duas.

A utilidade principal desta função é para a apresentação de resultados em um determinado formato.

Para voltar a poder operar com a *fileira* como *número* usa-se a função VAL.

Exemplo:

```
10 LET     A = 123
20 LET     B = 456
30 PRINT   STR$ (A) + STR$ (B)
40 LET     C = VAL (STR$ (A) + STR$ (B))
50 PRINT   C
```

cujo resultado após a execução é:

```
123 456
123456
```

onde a primeira linha apresentada corresponde à linha 30, e o espaço que aparece separando as duas fileiras de números é o reservado para o sinal algébrico; já na segunda linha apresentada, o efeito da função VAL foi transformar novamente em número a junção (soma) das duas fileiras e o espaço foi ignorado, como ocorre quando se aplica VAL, conforme já foi exposto ao tratar da função anterior.

MID$

Leia-se MID cifrão, MID fileira, ou MID seqüência.

Esta função retira uma subfileira, constituída pelos caracteres entre o de ordem n e o de ordem n + m, de uma fileira.

Tanto o nome da fileira como os números que indicam o início, n, e o comprimento, m, da nova fileira devem estar entre parênteses, logo adiante de MID$.

Podem ser utilizados valores numéricos ou também expressões para n e m. No caso de o resultado das expressões não ser inteiro, será considerada apenas a parte inteira.

Exemplo:

Considere-se o programa:

```
10 LET     P$ = "3/16"
20 LET     D$ = MID$ (P$, 3, 2)
30 PRINT   "DENOMINADOR = "; D$
```

o resultado será:

```
DENOMINADOR = 16
```

Capítulo 6

APLICAÇÕES PRÁTICAS

6.1 – **Apresentação**

Neste capítulo, apresentam-se exemplos de programas completos, para aplicações práticas. Alguns dos programas são de utilidade em diversos campos, como economia, matemática, engenharia e outros.

Muito se aprende examinando e, às vezes, melhorando programas já prontos. Esta é a razão principal pela qual estes programas, embora tão variados, foram reunidos neste capítulo.

Foram utilizadas diversas marcas de computadores e, por isso, ao aplicá-los em uma máquina diferente, talvez possa haver necessidade de alguma pequena adaptação. Naturalmente para uma aplicação profissional específica, convém desenvolver um programa especial, para máxima eficiência.

6.2 – **Gráficos**

Apresentam-se, a seguir, alguns exemplos de gráficos para um computador no qual o eixo x tem 63 posições e o y, 43 posições, iniciando-se este de baixo para cima.

A) *Geração de linhas contínuas, paralelas a x e a y*

```
10 FOR  I = 1 TO 20
20 PLOT I, 20
30 PLOT 20, I
40 NEXT I
```

E, após RUN seguido de FL, tem-se no vídeo:

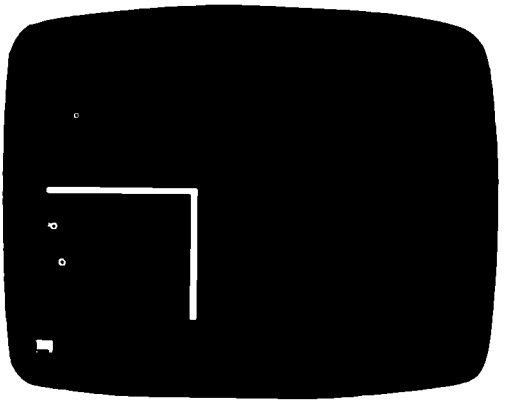


Figura 6.1

B) *Geração de eixos com cruzamento no centro*

```
10 FOR  I = 1 TO 63
20 PLOT I, 20
30 NEXT I
40 FOR  J = 1 TO 43
50 PLOT 32, J
60 NEXT J
```

E no vídeo, tem-se:

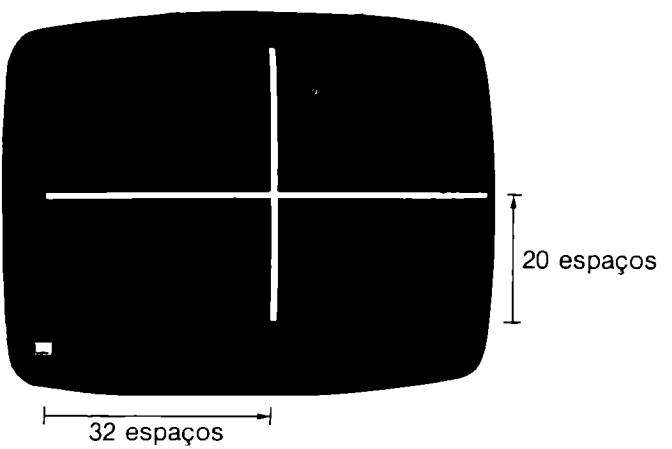


Figura 6.2

C) *Geração de eixos deslocados*

Quando se pretende apresentar somente o quarto quadrante, desloca-se o eixo x para baixo e o y para a direita. O seguinte programa pode ser utilizado:

```
10 FOR  I = 1 TO 63
20 PLOT I, 3
30 NEXT I
40 FOR  J = 1 TO 43
50 PLOT 61, J
60 NEXT J
```

E o resultado será:

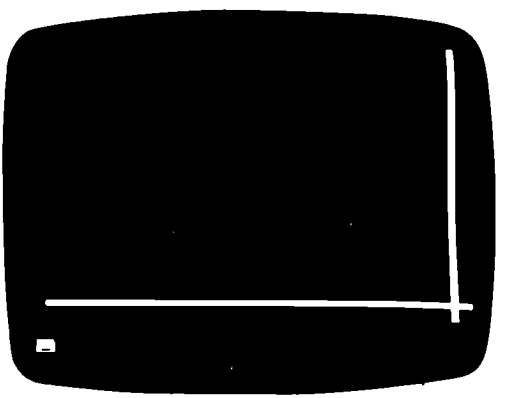


Figura 6.3

D) *Geração de um losango*

Este programa gera a figura de um losango a partir das equações das retas que o compõem.

Foi utilizado um computador onde a instrução para produzir um ponto (retângulo) é SET em vez de PLOT, como nos programas anteriores. Para as máquinas que usam PLOT é só colocá-lo em substituição a SET.

O recurso para manter na tela somente o losango sem apresentar as mensagens normalmente emitidas pelo computador ao terminar o programa, foi aplicar uma instrução CLS na linha 10 e provocar a entrada em um anel fechado,

com a instrução da linha 110. Conforme o computador utilizado pode ser necessário ou conveniente eliminá-la.

```
10 CLS
20 FOR  I = 0 TO 20
30 LET  J = 20 + I/2
40 LET  K = 20 – I/2
50 SET  (I, J)
60 SET  (I + 1, K)
70 LET  L = 41 – I
80 SET  (L – 1, K)
90 SET  (L, J)
100 NEXT I
110 GOTO 110
```

No vídeo, apresenta-se a seguinte figura:

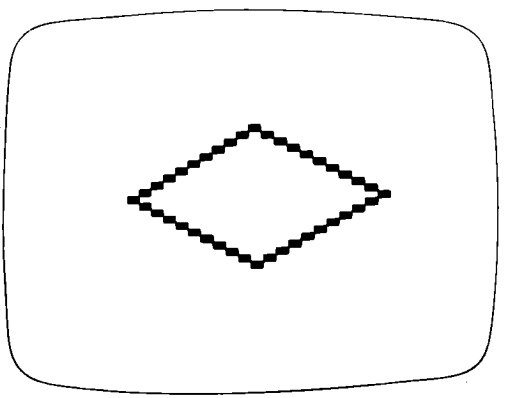


Figura 6.4

E) *Função co-seno*

Para computador com instrução PLOT e 63 × 43 posições As instruções 30, 40 e 50 servem para colocar os eixos de coordenadas.

```
10 FOR  N = 0 TO 63
20 PLOT N, 20 + 20 * COS (N/16 * PI)
30 PLOT N, 20
40 IF   N > 43 THEN GOTO 60
50 PLOT 0, N
60 NEXT N
```

E no vídeo o resultado será:

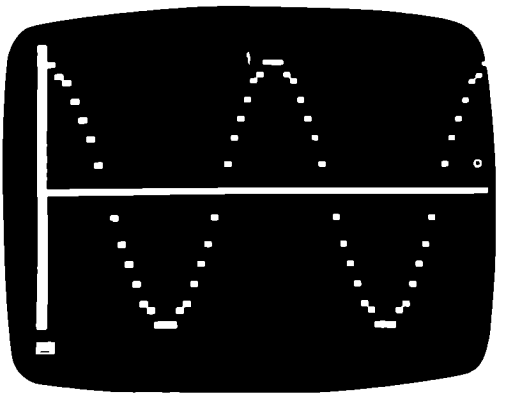


Figura 6.5

F) *Função co-seno amortecida*

```
10 FOR  N = 0 TO 63
20 PLOT N, 20 + 20 * COS (N/16 * PI) EXP (-N/24)
30 PLOT N, 20
40 IF   N > 43 THEN GOTO 60
50 PLOT 0, N
60 NEXT N
```

No vídeo, o resultado será:

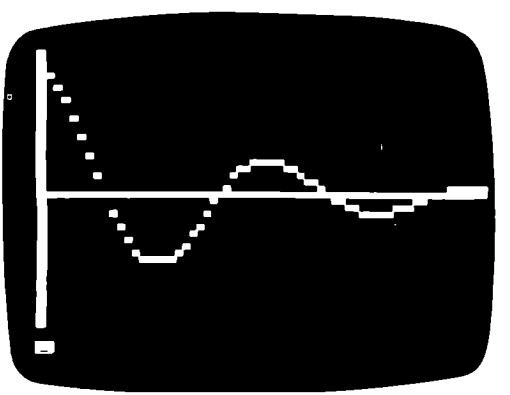


Figura 6.6

G) *Função co-seno amortecida com eixos graduados*

```
 10 FOR  N = 0 TO 63
 20 PLOT N, 20 + 20 * COS (N/16 * PI) * EXP (-N/24)
 30 PLOT N, 20
 40 IF   N > 43 THEN GOTO 60
 50 PLOT 0, N
 60 NEXT N
 70 PLOT  1, 10
 80 PLOT 2, 10
 90 PLOT 1, 15
100 PLOT 1, 25
110 PLOT 1, 30
120 PLOT 2, 30
130 FOR  M = 16 TO 49 STEP 16
140 FOR  L = 19 TO 21
150 PLOT M, L
160 NEXT L
170 NEXT M
```

A figura no vídeo fica:

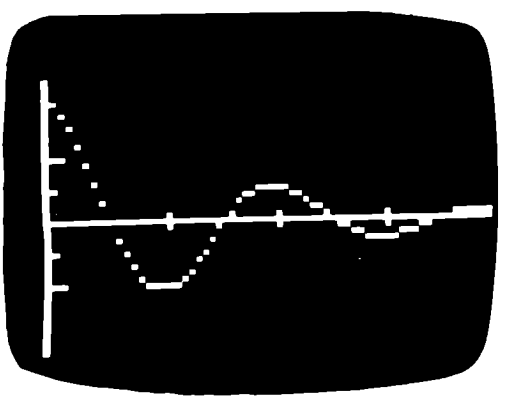


Figura 6.7

Note-se que entre as linhas 130 e 170 está a graduação do eixo x, e entre as linhas 70 e 120 a do eixo y.

H) *Tabuleiro para jogo de xadrez*

```
 10 FOR  X = 10 TO 40 STEP 10
 20 FOR  Y = 1 TO 31 STEP 10
 30 GOSUB 200
 40 NEXT Y
 50 NEXT X
 60 FOR  X = 15 TO 45 STEP 10
 70 FOR  Y = 6  TO 36 STEP 10
 80 GOSUB 200
 90 NEXT Y
100 NEXT X
190 STOP
200 FOR  I = X TO X + 4
210 FOR  J = Y TO Y + 4
220 PLOT I, J
230 NEXT J
240 NEXT I
250 RETURN
```

Observe-se que a sub-rotina que se inicia na linha 200 produz uma casa preta do tabuleiro, cujo vértice inferior esquerdo está nas coordenadas X e Y. Foi utilizado um computador com modulação de vídeo inversa, isto é, PLOT produz um quadrado preto. Se for empregada a modulação direta, PLOT produz um quadrado branco e há necessidade de alterar as linhas 10, 20, 60 e 70, para manter o quadrado branco à direita do tabuleiro, como é correto.

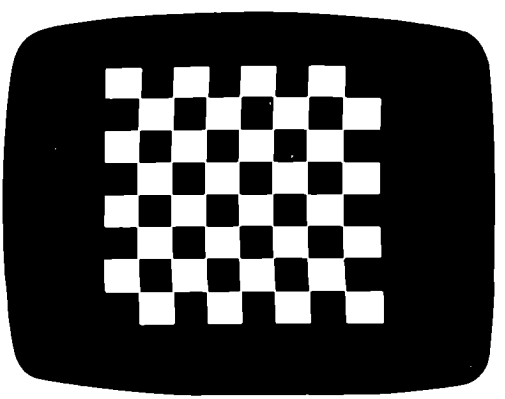


Figura 6.8

6.3 – **Matemática**

6.3.1 – *Solução de equação de segundo grau – Fórmula de Baskara*

```
 10 REM EQUACAO DO SEGUNDO GRAU
 20 PRINT "FORNECER O COEFICIENTE DO TERMO DE GRAU 2"
 30 INPUT A
 40 CLS
 50 PRINT "FORNECER O COEFICIENTE DO TERMO DE GRAU 1"
 60 INPUT B
 70 CLS
 80 PRINT "FORNECER O COEFICIENTE DO TERMO DE GRAU 0"
 90 INPUT C
100 CLS
110 PRINT "RAIZES DA EQUACAO:"
120 PRINT, , TAB 1 + LEN STR$ A; 2
130 PRINT A; "X +"; B; "X + "; C; "=0"
140 LET XR = -B / (2 * A)
150 LET D = B * B - 4 * A * C
160 IF D < 0 THEN GOTO 240
170 IF D = 0 THEN GOTO 220
180 LET XR1 = (SQR D) / (2 * A)
190 PRINT,, "X1 = "; XR + XR1
200 PRINT "X2 = "; XR - XR1
210 STOP
220 PRINT,, "X1 = X2 = "; XR
230 STOP
240 LET XI = (SQR - D) / (2 * A)
250 PRINT,, "X1 = "; XR; "+ J"; XI; TAB 0; "X2 = "; XR; " - J"; XI
```

O programa foi formatado para um microcomputador que possui vídeo de 32 colunas divididas em dois campos de 16.

Observe-se ainda que foram utilizadas variáveis com mais de 2 caracteres, o que não é aceito em qualquer computador. Um aspecto importante a considerar sobre o número de letras de uma variável é que alguns computadores aceitam um nome com mais de duas letras, porém só consideram para efeito de identificação as duas primeiras, de forma

que, no exemplo da equação de segundo grau, XR1 e XR seriam as mesmas variáveis, produzindo, portanto, resultados incorretos.

Alguns parênteses também foram omitidos pelo fato do computador utilizado não necessitar. Por exemplo, algumas versões só aceitariam as linhas 120, 180 e 240 escritas na forma:

```
120 PRINT„ TAB (1 + LEN (STR$ (A)))
180 LET XR1 = (SQR (D)) / (2 * A)
240 LET XI = (SQR(-D)) / (2*A)
```

Dado o comando RUN, o programa pede que se entre com os coeficientes da equação. A cada coeficiente fornecido deve-se pressionar a tecla correspondente ao fim de linha.

Exemplos:

```
1)
RAIZES DA EQUACAO
1X^2 + 4X + 68 = 0
X1 = -2 + J8
X2 = -2 - J8

2)
RAIZES DA EQUACAO
2X^2 + 18X + 40.5 = 0
X1 = X2 = -4.5

3)
RAIZES DA EQUACAO
3X^2 + 24X + -60 = 0
X1 = 2
X2 = -10
```

Para se evitar que a saída forneça para B ou C negativos a impressão $+-$B (C), pode ser utilizado o recurso: adicionar ao programa:

```
63 LET B$  = "X+"
66 IF  SGN (B) = -1 THEN LET B$ = "X-"
93 LET C$  = "X+"
96 IF  SGN (C) = -1 THEN LET C$ = "X-"
```

e trocar a linha 130 por:

```
130 PRINT A; B$; ABS (B); C$; ABS (C); "=0"
```

Desta forma, o exemplo 3 fornece:

$3X^2 + 24X - 60 = 0$
X1 = 2
X2 = -10

Para a equação $x^2 - 2x - 15 = 0$, obtém-se:

RAIZES DA EQUACAO
$1x^2 - 2x - 15 = 0$
X1 = 5
X2 = -3

Nota: Para o cálculo de $D = B^2 - 4AC$, foi utilizada a expressão $D = B * B - 4 * A * C$, ao invés de $D = B \uparrow 2 - 4 * A * C$, pois a primeira fornece uma precisão numérica maior.

6.3.2 – *Cálculo de determinante*

```
010 REM LEITURA DA ORDEM DA MATRIZ
020 READ N
030 DIM A (N, N)
040 REM LEITURA DOS ELEMENTOS DA MATRIZ
050 FOR I = 1 TO N
060 FOR J = 1 TO N
070 READ A (I, J)
080 NEXT J
090 NEXT I
100 FOR I = 1 TO N - 1
110 LET K = I
120 IF A (I, I) = 0 THEN GO TO 250
130 FOR I 1 = I + 1 TO N
140 LET AUX = A (I 1, I) / A (I, I)
150 FOR J = I TO N
160 LET A (I 1, J) = A (I 1, J) - AUX * A (I, J)
170 NEXT J
180 NEXT I 1
```

```
185 NEXT I
190 LET DET = 1
200 FOR I = 1 TO N
210 LET DET = DET * A (I, I)
220 NEXT I
230 PRINT "O DETERMINANTE DA MATRIZ
    DADA E IGUAL A"; DET
240 STOP
250 IF K < > N THEN GO TO 280
260 PRINT "O DETERMINANTE DA MATRIZ
    DADA E IGUAL A ZERO"
270 STOP
280 LET K = K + 1
290 FOR J = I TO N
300 LET A (I, J) = A (I, J) + A (K, J)
310 NEXT J
320 GO TO 120
330 DATA 5
340 DATA 5, 3, 3, 1, -3
350 DATA 2, 1, 2, 4, 0
360 DATA -3, 0, -3, 7, 0
370 DATA 0, -2, 0, -1, 2
380 DATA 3, -2, 1, -3, 1
```

Resultado:

```
RUN
O DETERMINANTE DA MATRIZ DADA E IGUAL A 24
```

Modo de utilizar:

Para o cálculo do determinante deve-se fornecer os dados referentes à matriz, na seguinte ordem: a ordem da matriz na instrução DATA da linha 330; os elementos da primeira linha da matriz na instrução DATA da linha 340, os da segunda linha da matriz numa instrução DATA com número 350, e assim sucessivamente até serem fornecidos os elementos de todas as linhas.

Desta forma, ao executar o programa com o comando RUN obtém-se o determinante.

A linha 320 produziria um anel fechado e a execução do programa nunca terminaria; as instruções STOP das linhas

240 e 270 evitam o anel e produzem a parada após a apresentação do resultado.

6.3.3 – *Inversão de matriz*

Programa que executa a inversão de uma matriz de ordem N.

O programa utiliza o método de GAUSS-JORDAN para a inversão da matriz.

```
010 READ N
020 DIM A (N, 2 * N)
030 FOR I = 1 TO N
040 FOR J = 1 TO N
050 READ A (I, J)
060 LET A ( I, N + J) = 0
070 NEXT J
080 LET A (I, N + I) = 1
090 NEXT I
100 FOR I = 1 TO N
110 LET K = I
120 IF A (I, I) = 0 THEN 350
130 LET DIV = A (I, I)
140 FOR J = I TO 2 * N
150 LET A (I, J) = A (I, J) / DIV
160 NEXT J
170 FOR I 1 = 1 TO N
180 IF I 1 = I THEN 230
190 LET AUX = A (I 1, I)
200 FOR J = I TO 2 * N
210 LET A (I 1, J) = A (I 1, J) - AUX * A (I, J)
220 NEXT J
230 NEXT I 1
240 NEXT I
250 PRINT "A MATRIZ INVERSA E"
260 FOR I = 1 TO N
270 PRINT
280 PRINT "LINHA"; I
290 FOR J = N + 1 TO 2 * N
300 PRINT A (I, J),
```

```
310 NEXT J
330 NEXT I
340 STOP
350 IF K < > N THEN 380
360 PRINT "NAO EXISTE A MATRIZ INVERSA"
370 STOP
380 LET K = K + 1
390 FOR J = I TO 2 * N
400 LET A (I, J) = A (I, J) + A (K, J)
410 NEXT J
420 GO TO 120
430 DATA
440 DATA
450 END
```

Os dados da ordem da matriz (N) e os elementos da matriz devem ser fornecidos em:

```
430 DATA N
440 DATA a11, a12, ... a1n, a21, ... ann
```

(fornecidos linha por linha)

O programa apresenta como resultados:

1) se a matriz possui ou não a sua inversa;
2) caso exista a matriz inversa, esta é apresentada.

Observação: Caso os resultados ultrapassem o campo do monitor, deve-se colocar uma instrução STOP na linha 320, e desta forma os resultados da matriz inversa serão mostrados linha por linha.

Para se obter uma nova linha deve-se executar o comando CONT (seguido de FL).

Exemplos:

1) $N = 3$ $\qquad A = \begin{bmatrix} 2 & 1 & -1 \\ 1 & 0 & 3 \\ 1 & 1 & -4 \end{bmatrix}$

Resultado:

```
NAO EXISTE A MATRIZ INVERSA
```

2) $N = 3$ $\quad A = \begin{bmatrix} 0 & 2 & 4 \\ 1 & 0 & 0 \\ 2 & 1 & 1 \end{bmatrix}$

Resultado:

```
 0      1    0
-0.5   -4    2
 0.5    2   -1
```

6.3.4 — *Divisão de polinômios*

$$\begin{array}{l|l} P(x) & D(x) \\ \hline R(x) & Q(x) \end{array}$$

```
 10 READ N : DIM P (N + 1)
 20 FOR I = N + 1 TO 1 STEP - 1
 30 READ P (I) : NEXT I
 40 READ M : DIM D (M + 1)
 50 FOR I = M + 1 TO 1 STEP - 1
 60 READ D (I) : NEXT I
 70 IF (N - M + 1) < = 0 GO TO 390
 80 DIM Q (N - M + 1)
 90 FOR I = N - M + 1 TO 1 STEP - 1
100 LET Q (I) = P (I + M) / D (M + 1)
110 FOR J = M + 1 TO 1 STEP - 1
120 LET P (I + J - 1) = P (I + J - 1) - Q (I) * D (J)
130 NEXT J
140 NEXT I
145 PRINT
150 PRINT "DIVIDENDO" ,,,,
160 RESTORE
```

```
170 READ K
175 DIM PK (K + 1), PP$ (K + 1)
180 FOR I = K + 1 TO 1 STEP - 1
190 READ PK (I) : NEXT I
200 GOSUB 2000
210 PRINT
220 PRINT "DIVISOR" ,,,,
230 LET K = M
240 FOR I = 1 TO M + 1
250 LET PK (I) = D (I) : NEXT I
260 GOSUB 2000
265 PRINT
270 PRINT "QUOCIENTE" ,,,,
280 LET K = N - M
290 FOR I = 1 TO N - M + 1
300 LET PK (I) = Q (I) : NEXT I
310 GOSUB 2000
320 PRINT
330 PRINT "RESTO" ,,,,
340 LET K = M - 1
350 FOR I = 1 TO M
360 LET PK (I) = P (I) : NEXT I
370 GOSUB 2000
375 FOR I = 1 TO M : IF PK (I) < > 0 THEN LET PNULO
    = 1 : NEXT I
377 IF PNULO < > 1 THEN PRINT "bbb0"
380 GO TO 410
390 CLS
395 PRINT "O QUOCIENTE DOS DOIS POLINOMIOS E
    O POLINOMIO NULO E O RESTO E O DIVIDENDO"
400 STOP
410 IF INDIC < > 1 THEN STOP
415 CLS : PRINT
420 PRINT "COEFICIENTES EM ORDEM DE
    POTENCIAS DECRESCENTES"
425 PRINT
430 PRINT "QUOCIENTE", "GRAU b = b"; N - M
440 FOR I = N - M + 1 TO 1 STEP - 1
450 PRINT Q (I), : NEXT I
460 PRINT
470 PRINT "RESTO", "GRAU b < = b"; M - 1
```

```
 480 FOR I = M TO 1 STEP - 1
 490 PRINT P (I), : NEXT I
 495 PRINT : PRINT
 500 STOP
 510 DATA 5
 520 DATA 5, 4, 2, 0, 1, -2
 530 DATA 2
 540 DATA 1, 0, 1
2000 FOR I = 1 TO K + 1
2010 LET PP$ (I) = STR$ (ABS (PK (I))) : NEXT I
2020 SUM = 0
2030 FOR I = 1 TO K + 1
2040 IF PK (I) = 0 THEN GO TO 2060
2050 SUM = SUM + LEN (PP$ (I))
2060 NEXT I
2070 IF SUM + (K * 7) + 2 > 63 GO TO 2240
2080 PRINT "bb";
2090 FOR I = K + 1 TO 3 STEP - 1
2100 IF PK (I) = 0 GO TO 2120
2105 IF I = 2 GO TO 2120
2107 IF I = 1 GO TO 2120
2110 LET B$ = "b b b b b b b b b b b b b b b b b b"
2115 PRINT MID$ (B$, 1, LEN (PP$ (I)) + 2); I -1; "b b b";
2120 NEXT I
2130 PRINT
2140 FOR I = K + 1 TO 1 STEP - 1
2150 IF PK (I) = 0 THEN GO TO 2220
2160 LET S$ = "+"
2170 IF PK (I) < 0 THEN LET S$ = "-"
2180 IF I = K + 1 THEN LET S$ = "b"
2190 PRINT S$; "b"; ABS (PK (I));
2200 IF I = 1 THEN GO TO 2220
2210 PRINT "b X b b b";
2220 NEXT I
2230 RETURN
2240 INDIC = 1 : GO TO 2230
```

O programa realiza a divisão de polinômios pelo método da chave.

Entrada de dados

Os dados são fornecidos nas instruções DATA:

N.º da linha	Instrução	Dados
510	DATA	GRAU DO DIVIDENDO
520	DATA	COEFICIENTES DO DIVIDENDO EM ORDEM DE POTÊNCIAS DECRESCENTES
530	DATA	GRAU DO DIVISOR
540	DATA	COEFICIENTES DO DIVISOR EM ORDEM DE POTÊNCIAS DECRESCENTES

Saída de dados

Exemplo 1

ENTRADA:

```
510 DATA 5
520 DATA 5, 4, 2, 0, 1, -2
530 DATA 2
540 DATA 1, 0, 1
```

SAÍDA:

```
DIVIDENDO
5X^5 + 4X^4 + 2X^3 + X - 1
DIVISOR
1X^2 + 1
QUOCIENTE
5X^3 + 4X^2 - 3X - 4
RESTO
4X + 2
```

Exemplo 2

ENTRADA:

```
510 DATA 6
520 DATA 33, 2, 0, 4, 0, 2, -1.5
530 DATA 3
540 DATA 1, 0, -2, 4
```

SAÍDA:

```
DIVIDENDO
33X^6 + 2X^5 + 4X^3 + 2X - 1.5
DIVISOR
1X^3 - 2X + 4
QUOCIENTE
33X^3 + 2X^2 + 66X - 124
RESTO
124X^2 - 510X + 494.5
```

Exemplo 3

ENTRADA:

```
510 DATA 7
520 DATA 1.12, 1, 14.2, 3, 7.3, 0.82, 0, 4.5
530 DATA 4
540 DATA 3, 2, 1, -1, 2.4
```

SAÍDA:

```
COEFICIENTES EM ORDEM DE POTENCIAS DECRESCENTES
QUOCIENTE GRAU = 3
.373333 .084444 4.55259 -1.93877
RESTO       GRAU < = 3
5.81338 7.10869 -12.865 9.15304
```

Exemplo 4

ENTRADA:

```
510 DATA 5
520 DATA 3, 3, 1.3, 2, 4, 3.7
530 DATA 5
540 DATA 2, 1, 1, 0, 3, 5
```

SAÍDA:

```
DIVIDENDO
3X^5 + 3X^4 + 1.3X^3 + 2X^2 + 4X + 3.7
DIVISOR
2X^5 + 1X^4 + 1X^3 + 3X + 5
QUOCIENTE
1.5
RESTO
1.5X^4 - .2X^3 + 2X^2 - .5X - 3.8
```

Exemplo 5

ENTRADA:

```
510 DATA 3
520 DATA 1, 1, 1, 1
530 DATA 4
540 DATA 2, -0.5, 1, 4.3, 6
```

SAÍDA:

```
O QUOCIENTE DOS DOIS POLINOMIOS E
O POLINOMIO NULO E O RESTO E O DIVIDENDO
```

Exemplo 6

ENTRADA:

```
510 DATA 5
520 DATA 2, 11, 14, 4, 20, 12
530 DATA 2
540 DATA 1, 4, 2
```

SAÍDA:

```
DIVIDENDO
2X^7 + 11X^6 + 14X^5 + 4X^4 + 19X^3 + 12X^2 + 14X + 8
DIVISOR
1X^2 + 4X + 2
QUOCIENTE
2X^5 + 3X^4 - 2X^3 + 6X^2 - 1*X + 4
RESTO
0
```

Observações:

a) Alguns computadores possuem um espaço reservado para dados alfanuméricos, o qual se excedido provoca parada no processamento. Nestas versões, a instrução CLEAR *m*, reserva um espaço, especificado pelo número *m*.

Por exemplo:

05 CLEAR 400 reserva 400 *bytes*,

o que para este programa é suficiente.

b) O símbolo b significa espaços em branco que devem ser colocados para que a formatação da saída seja a esperada.

6.3.5 — *Determinação das raízes de polinômio*

```
01 DEFINT K
02 LET K = 0
03 DEFDBL D
07 LET P = 1 E - 6
10 READ N
15 DIM A (N + 1), B (N + 1), C (N), Q (N - 1), D (3)
20 FOR I = 1 TO N + 1
22 CLS : PRINT
24 PRINT "PARTE REAL", "PARTE IMAG.",
   "PARTE REAL", "PARTE IMAG."
26 PRINT
30 READ A (I) : NEXT I
35 IF N < = 2 THEN GO TO 240
40 LET U = 0 : LET V = 0
50 LET B (1) = A (1) : LET B (2) = A (2) + U *B (1)
60 FOR I = 3 TO N + 1
70 LET B (1) = A (1) + U * B (I - 1) + V * B (I - 2) : NEXT I
80 LET C (1) = B (1) : LET C (2) = B (2) + U * C (1)
90 FOR I = 3 TO N
100 LET C (I) = B (I) + U * C (I - 1) + V * C (I - 2) : NEXT I
110 LET DELTA = C (N - 1) [2 - C (N) * C (N - 2)
120 IF DELTA = 0 THEN GO TO 180
```

```
130 LET DU = (- B (N) * C (N - 1) + B (N + 1)
    * C (N - 2)) / DELTA
140 LET DV = (- C (N - 1) * B (N + 1) + B (N)
    * C (N)) / DELTA
150 IF ABS (DU) < P AND ABS (DV) < P THEN GO TO 200
152 IF K > 50 THEN LET P = P * 10
155 LET K = K + 1
160 LET U = DU + U : LET V = DV + V
170 GO TO 50
180 LET U = U + 1 : LET V = V + 1
190 GO TO 50
200 GOSUB 500
205 PRINT
210 GOSUB 1000
215 LET K = 0 : LET P = 1 E - 6
217 IF P 1 < P LET P 1 = P
220 LET N = N - 2
230 IF N > 2 THEN GO TO 40
240 IF N = 1 GO TO 270
250 GOSUB 500
260 GO TO 275
270 PRINT – A (2) / A (1), 0
275 PRINT : PRINT "PRECISAO NOS COEFICIENTES
    DA QUADRATICA = b", P 1
280 STOP
290 DATA 3
300 DATA 1, -5, 9, -5
500 REM FORMULA DE BASKARA
502 IF N < > 2 THEN GO TO 510
504 LET U = -A (2) / A (1) : LET V = -A (3) / A (1)
510 LET X R = U / 2
520 LET D = U [ 2 + 4 * V
530 IF D < 0 THEN GO TO 600
540 IF D = 0 THEN GO TO 580
550 LEX X 1 = (SQR (D)) / 2
560 PRINT X 1 + X R, 0, X R – X 1, 0
570 RETURN
580 PRINT X R, 0, X R, 0
590 GO TO 570
600 LET XI = (SQR (– D)) / 2
610 PRINT XR, XI, XR, – XI
```

```
 620 GO TO 570
1000 REM DIVISAO DE POLINOMIOS
1010 LET D (1) = 1 : LET D (2) = - U : LET D (3) = - V
1020 FOR I = I TO N - 1
1025 LET Q (I) = A (I)
1030 FOR J = 1 TO 3
1040 LET A (I + J - 1) - A (I + J - 1) — Q (I) * D (J)
1050 NEXT J
1060 NEXT I
1070 FOR I = 1 TO N - 1
1080 LET A (I) = Q (I) : NEXT I
1090 RETURN
```

O programa utiliza o método de Lin-Bairstow para a determinação das raizes do polinômio.

Os dados referentes ao polinômio devem ser fornecidos como indica a seguinte tabela:

290	DATA	GRAU DO POLINÔMIO
300	DATA	COEFICIENTES EM ORDEM DE POTÊNCIAS DECRESCENTES

A saída de dados foi feita para o vídeo dividido em quatro campos.

Exemplo 1

```
290 DATA 3
300 DATA 2, 7, 0, -9
```

Resultado:

```
PARTE REAL  PARTE IMAG.  PARTE REAL  PARTE IMAG.
    1           0           -1.5          0
   -3           0
```

Exemplo 2

```
290 DATA 3
300 DATA 1, -5, 9, -5
```

Resultado:

```
PARTE REAL  PARTE IMAG.  PARTE REAL  PARTE IMAG.
    2           1            2          -1
    1           0
PRECISAO NOS COEFICIENTES DA QUADRATICA = 1 E - 06
```

6.3.6 — *Ajuste de polinômios pelo método dos mínimos quadrados*

```
 10 CLS
 20 READ N : REM N E O NUMERO DE PARES DE PONTO
 30 READ G : REM G E O GRAU DO POLINOMIO
 40 REM LEITURA DOS PARES DE PONTO X, Y
 50 DIM X (N), Y (N), YC (N), R (N)
 60 FOR I = 1 TO N
 70 READ X (I), Y (I)
 80 NEXT I
 90 REM CALCULO DOS COEFICIENTES
    DO POLINOMIO DE GRAU G
100 DIM A (G + 1, 2 * (G + 1)), B (G + 1), C (G + 1)
110 FOR I = 1 TO G + 1
120 FOR J = I TO G + 1
130 FOR K = 1 TO N
140 LET A (I, J) = A (I, J) + X (K) ↑ (I + J - 2)
150 NEXT K
160 LET A (J, I) = A (I, J)
170 NEXT J
180 LET A (I, G + I + 1) = 1
190 NEXT I
200 FOR I = 1 TO G + 1
210 FOR K = 1 TO N
220 LET B (I) = (X (K) ↑ (I - 1)) * Y (K) + B (I)
230 NEXT K
240 NEXT I
250 FOR I = 1 TO (G + 1)
```

```
260 LET K = I
270 IF A (I, I) = 0 THEN GOTO 630
280 LET DIV = A (I, I)
290 FOR J = I TO 2 * (G + 1)
300 LET A (I, J) = A (I, J) / DIV
310 NEXT J
320 FOR I 1 = 1 TO (G + 1)
330 IF I 1 = I THEN GOTO 380
340 LET AUX = A (I 1, I)
350 FOR J = I TO 2 * (G + 1)
360 LET A (I 1, J) = A (I 1, J) -•AUX * A (I, J)
370 NEXT J
380 NEXT I 1
390 NEXT I
400 FOR I = 1 TO G + 1
410 FOR J = G + 2 TO 2 * (G + 1)
420 LET C (I) = C (I) + A (I, J) * B (J - G - 1)
430 NEXT J
440 NEXT I
450 PRINT "COEFICIENTES DO POLINOMIO
    EM ORDEM DE POTENCIAS DECRESCENTES"
460 PRINT
470 FOR I = 1 TO G + 1
480 PRINT C (G + 2 - I)
490 NEXT I
500 PRINT : PRINT
510 STOP : CLS
520 PRINT "APRESENTACAO DOS RESIDUOS"
530 PRINT : PRINT "X", "R (X)"
540 FOR K = 1 TO N
550 FOR J = 1 TO G + 1
560 LET YC (K) = YC (K) + C (J) * (X (K) ↑ (J - 1))
570 NEXT J
580 LET R (K) = (YC (K) - Y (K))
590 PRINT X (K), R (K)
600 IF K = 10 THEN STOP
610 NEXT K
620 STOP
630 IF K <> G + 1 THEN GOTO 650
640 STOP
650 LET K = K + 1
```

```
660 FOR J = I TO 2 * (G + 1)
670 LET A (I, J) = A (I, J) + A (K, J)
680 NEXT J
690 GOTO 270
790 DATA . . .
800 DATA . . .
810 DATA . . .
```

Modo de utilização

O programa ajusta um polinômio de grau G por N pares de pontos X, Y.

A entrada de dados deve ser feita na seguinte ordem:

a) Número de pares de ponto, na instrução DATA da linha 790.
b) Grau do polinômio, na instrução DATA da linha 800.
c) Pares de ponto (X, Y), na instrução DATA da linha 810.

Se o número de pontos for muito grande, pode-se, para maior comodidade, desdobrar a instrução DATA da linha 810 em várias linhas, numerando-as em seqüência crescente, por exemplo: 820, 830, ...

A saída do programa é dada por partes.

Inicialmente são fornecidos os coeficientes do polinômio ajustado em ordem de potências decrescentes. Por exemplo, se o polinômio desejado for:

$$C_4X^3 + C_3X^2 + C_2X + C_1,$$

então os coeficientes serão fornecidos um em cada linha na ordem C_4, C_3, C_2 e C_1.

Através do comando CONT obtém-se a segunda parte dos resultados onde são apresentados os resíduos, ou seja, a diferença entre os valores de tabela (Y) e os obtidos do polinômio, para todos os valores de X fornecidos ao programa.

Caso o número de pares de pontos entrados exceda a 10, deve-se utilizar novamente o comando CONT até que se obtenham todos os valores.

Exemplo A

Deseja-se ajustar um polinômio de grau 1, $C_2X + C_1$ para os pontos da seguinte tabela:

X	Y
0	0
1	2.3
2	4

As instruções DATA, neste caso, devem ser:

```
790 DATA 3
800 DATA 1
810 DATA 0, 0, 1, 2.3, 2, 4
```

Após o comando RUN, tem-se:

```
COEFICIENTES DO POLINOMIO EM ORDEM
DE POTENCIAS DECRESCENTES
2
.1
```

O polinômio obtido é então $2X + 0.1$. Com o comando CONT obtém-se:

Apresentação dos resíduos:

```
X       R (X)
0        .1
1      - .2
2        .1
```

Exemplo B

Ajuste de um polinômio de grau 2 para os mesmos pontos do exemplo A.

```
790 DATA 3
800 DATA 2
810 DATA 0, 0, 1, 2.3, 2, 4
```

Resultados:

```
COEFICIENTES DO POLINOMIO EM ORDEM
DE POTENCIAS DECRESCENTES
- .300003
  2.60001
- 9.53674E - 07
```

Apresentação dos resíduos:

```
X    R (X)
0    - 9.53674E - 07
1      2.14577E - 06
2    - 9.53674E - 07
```

O polinômio obtido foi:

$-.300003X^2+2.60001X-9.53674E-07$, ao invés de $-0.3X^2+2.6X$, este último seria a solução exata; devido a problemas de precisão numérica do computador, houve pequena divergência.

Exemplo C

Ajuste de um polinômio de grau 4 para os pontos da seguinte tabela:

X	Y
0	− 7
.2	− 6.68
.4	− 6.27
.6	− 5.60
.8	− 3.94
1	− 1
1.2	3.85
1.4	11.3
1.6	22.1
1.8	37
2	57

Resultados:

```
COEFICIENTES DO POLINOMIO EM ORDEM
DE POTENCIAS DECRESCENTES
2.00684
4.98828
-2.99219
1.99219
-6.99756
```

Apresentação dos resíduos:

```
X     R (X)
0     2.44141E - 03
.2    4.30918E - 03
.4  - .0388088
.6    .0581212
.8  - 2.80833E - 03
1   - 2.44141E - 03
1.2   .0154414
1.4   .0241165
1.6   .0139503
1.8   .0523262
2     .03366914
```

6.4 — **Engenharia**

6.4.1 — *Conversão de polegadas para milímetros*

```
 10 CLS
 20 PRINT "POLEGADAS", "MILIMETROS"
 30 LET TS = 0
 40 LET IN = 0
 50 FOR I = 1 TO 10
 60 LET DN = 16
 70 LET DI = 1
 80 IF I > = 9 THEN LET DI = 2
 90 LET TS = TS + DI
100 LET NU = TS
110 LET NU = NU/2
120 LET DN = DN/2
```

```
130 IF (INT(NU) - NU) = 0 THEN GOTO 110
140 PRINT 2 * NU; "/"; 2 * DN, (NU/DN) * 25.399
150 NEXT I
160 IF IN = 1 THEN GOTO 240
170 PRINT "SE DESEJAR ALGUM OUTRO VALOR, ENTRE
    COM ELE NA FORMA A/B, PELO TECLADO E
    PRESSIONE A TECLA DE FIM DE LINHA"
180 INPUT A$
190 CLS
200 LET IN = 1
210 LET TS = 0
220 PRINT "POLEGADAS", "MILIMETROS"
230 GOTO 50
240 FOR K = 1 TO LEN (A$)
250 IF MID $ (A$, K, 1) ="/" THEN GOTO 290
260 NEXT K
270 PRINT "O VALOR ENTRADO NAO ESTA
    NA FORMA A/B "; "VALOR IGUAL A"; " "; A$
280 GOTO 170
290 LET A = INT (VAL (MID $ (A$, 1, K - 1)))
300 LET B = INT (VAL (MID $ (A$, K + 1, LEN (A$) - K)))
310 PRINT "***"; A $, "***"; (A / B) * 25.399
320 GOTO 170
```

Com a execução do programa será apresentado:

```
POLEGADAS        MILIMETROS
    1/16            1.58744
    1/8             3.17488
    3/16            4.76231
    1/4             6.34975
    5/16            7.93719
    3/8             9.52463
    7/16           11.1121
    1/2            12.6995
    5/8            15.8744
    3/4            19.0493

SE DESEJAR ALGUM OUTRO VALOR, ENTRE COM ELE NA
FORMA A/B, PELO TECLADO E PRESSIONE
A TECLA DE FIM DE LINHA
?
```

Fornecendo o valor desejado, por exemplo 5/32, será apresentada a tabela acima com a inclusão, destacado por asteriscos, deste novo valor:

```
.
.
.
3/4            19.0493
***5/32        ***3.96859
SE DESEJAR . . .
```

Para interromper o programa deve-se utilizar o comando BREAK.

6.4.2 — *Cálculo da secção, resistência elétrica e peso de fios de cobre — AWG ou B & S*

```
 10 LET S0 = 53.4
 20 CLS
 30 PRINT "NUMERO", "SECCAO", "RESISTENCIA", "PESO"
 40 PRINT, "(MM2)", "ELET. (OHM/KM)", "(KG/KM)
 50 PRINT 0, " 53.4", "  0.32", "475.00"
 60 FOR I = 1 TO 27
 70 LET S = S0 * (1 / 2) ↑ (I / 3) - 0.001
 80 LET P = 8.895 * S
 90 LET R = 17.2482 / S
100 PRINT I, USING "# # # . # #"; S;
110 PRINT, USING "# # # . # #"; R;
120 PRINT, USING "# # # . # #"; P
130 IF I = 10 THEN STOP
140 IF I = 20 THEN STOP
150 NEXT I
```

Este programa produz tabelas que fornecem a resistência elétrica e o peso de fios desde o número 0 até 27.

Quando se executa o programa é apresentada uma tabela para fios do n.º 0 ao n.º 10. As outras tabelas são obtidas pelo comando CONT.

6.4.3 – *Cálculo analítico da área de terrenos*

```
 10 READ N
 20 DIM X (N), Y (N)
 30 FOR I = 1 TO N
 40 READ X (I), Y (I)
 50 NEXT I
 60 LET S = 0
 70 FOR I = 1 TO N - 1
 80 LET S = S + Y (I) * X (I + 1) - Y (I + 1) * X (I)
 90 NEXT I
100 LET S = S + Y (N) * X (1) - Y (1) * X (N)
110 LET S = S/2
120 CLS
130 PRINT "A AREA DO TERRENO E IGUAL A "; S; "M 2"
140 DATA . . .
150 DATA . . .
```

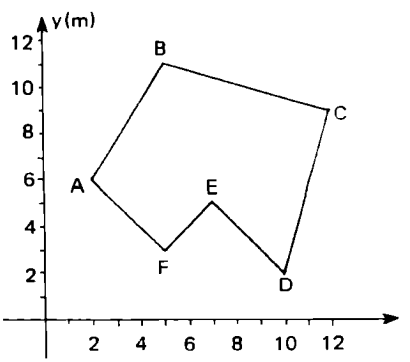
O programa calcula a área de uma poligonal pela fórmula de Gauss.

Os dados a serem fornecidos são:

a) o número de vértices do terreno, na instrução DATA da linha 140;
b) as coordenadas dos vértices em metros, ordenados no sentido horário, na instrução DATA da linha 150.

Exemplo:

Deseja-se calcular a área do terreno da figura abaixo.



VÉRTICE	X	Y
A	2	6
B	5	11
C	12	9
D	10	2
E	7	5
F	5	3

As instruções DATA devem ser:

```
140 DATA 6
150 DATA 2, 6, 5, 11, 12, 9, 10, 2, 7, 5, 5, 3
```

Resultado:

```
A AREA DO TERRENO E IGUAL A 52.5 M 2
```

6.4.4 — *O critério de estabilidade de Routh*

O estabelecimento de condições gerais para que um sistema linear seja estável deve-se a J. E. Routh (1877).

Seja o polinômio característico:

$$a_0 s^n + a_1 s^{n-1} + \ldots + a_{n-1} s + a_n$$

Dos teoremas de Routh e pela definição de estabilidade pode-se concluir que: "a condição necessária e suficiente para que um sistema linear seja estável é que todos os elementos da primeira coluna da matriz de Routh tenham mesmo sinal e nenhum deles seja nulo".

A matriz de Routh é:

$$\begin{bmatrix} b_{1,1} & b_{1,2} & \ldots & b_{1,m} \\ b_{2,1} & b_{2,2} & \ldots & b_{2,m} \\ \vdots & \vdots & & \vdots \\ b_{n+1,1} & b_{n+1,2} & & b_{n+1,m} \end{bmatrix}$$

onde: n = grau do polinômio característico

$$m = \text{INT}\left(\frac{n+2}{2}\right)$$

$$b_{1,j} = a_{2j-2} \qquad j = 1,2, \ldots m$$

$$b_{2,j} = a_{2j-1} \qquad j = 1,2, \ldots m$$

No caso de n ser par, $b_{2,m} = 0$

O termo genérico para $i > 3$ é dado por:

$$b_{i,j} = \frac{-\begin{vmatrix} b_{i-2,1} & b_{i-2,j+1} \\ b_{i-1,1} & b_{i-1,j+1} \end{vmatrix}}{b_{i-1,1}}$$

Note-se que a matriz de Routh tem sempre $n + 1$ linhas. Ocorrendo um elemento nulo ou uma troca de sinal na primeira coluna, conclui-se que o sistema não é estável e não há necessidade de se prosseguir calculando os elementos restantes.

ALGORITMO

1.°) Fazer $m = \text{INT}\left(\frac{n+2}{2}\right)$.
2.°) Fazer $i = 3$.
3.°) Se $b_{i-1,1} = 0$, ir para 9.°.
4.°) Fazer $b_{i,j} = \frac{1}{b_{i-1,1}} (b_{i-1,1} \cdot b_{i-2,j+1}) -$
$- (b_{i-2,1} \cdot b_{i-1,j+1})$ para $j = 1, 2, \ldots m$.
5.°) Se $i = n + 1$, ir para 8.°.
6.°) Fazer $i = i + 1$.
7.°) Ir para 3.°.
8.°) Se sinal $(b_{i,1}) =$ sinal $(b_{i+1,1})$ para $i = 1, 2, \ldots n$ ir para 11.°.
9.°) Imprimir "SISTEMA NÃO ESTÁVEL".
10.°) Parar.
11.°) Imprimir "SISTEMA ESTÁVEL".
12.°) Parar.

Inicializados os valores da primeira e segunda linha da matriz de Routh, que são os coeficientes do polinômio característico e o grau do polinômio, a aplicação do algoritmo fornece como resultado informação sobre a estabilidade ou não do sistema.

Uma implementação do algoritmo em linguagem BASIC é mostrada a seguir.

```
 05 PRINT "FORNECER O GRAU"; "DO POLINOMIO"
 10 INPUT N
 15 CLS
 20 DIM B (N + 1, INT (N + 2) / 2)
 30 FOR K = 1 TO N + 1
 40 LET J = INT ((K + 1) / 2)
 50 LET I = 2 * INT (K / 2) - K + 2
 55 PRINT "FORNECER O COEFICIENTE DO TERMO
    DE GRAU"; N + 1 - K
 60 INPUT B (I, J)
 65 CLS
 70 NEXT K
 80 FOR I = 3 TO N + 1
 90 FOR J = 1 TO INT (N/2)
100 IF B (I - 1, 1) = 0 THEN GOTO 140
110 LET B (I, J) = (B (I - 1, 1) * B (I - 2, J + 1) - B (I - 2, 1)
    * B (I - 1, J + 1)) / B (I - 1, 1)
120 NEXT J
130 NEXT I
140 LET AUX = 0
150 FOR I = 1 TO N
160 IF NOT SGN (B (I, 1)) = SGN (B (I + 1, 1))
    THEN LET AUX = 1
170 NEXT I
180 LET R$ = "SISTEMA ESTAVEL"
190 IF AUX = 1 THEN LET R$ = "SISTEMA NAO ESTAVEL"
200 PRINT "COEFICIENTES DO POLINOMIO CARACTERISTICO
    EM ORDEM DE"; "POTENCIAS DECRESCENTES:"
210 FOR K = 1 TO N + 1
220 LET J = INT ((K + 1) / 2)
230 LET I = 2 * INT (K/2) - K + 2
240 PRINT "    "; B (I, J);
250 NEXT K
260 PRINT,, TAB 0; "GRAU DO POLINOMIO: "; N ,,,
270 PRINT "OS COEFICIENTES DA SERIE DE ROUTH; SAO DADOS
    PELO"; "PRIMEIRO NUMERO DE CADA BLOCO DE DADOS"
280 FOR I = 1 TO N + 1
```

```
290 PRINT ,,,, TAB 0; B (I, 1),,
300 FOR J = 2 TO INT (N + 2) / 2
310 PRINT B (I, J),
320 NEXT J
330 IF B (I, 1) = 0 THEN GOTO 350
340 NEXT I
350 FOR I = 1 TO N + 1
360 IF ABS (B (I, 1)) < 1 E - 5 THEN GOTO 400
370 NEXT I
380 PRINT ,, TAB 0; "RESULTADO: R$"
390 STOP
400 IF B (I, 1) = 0 THEN GOTO 370
410 PRINT,, TAB 0; "NAO E POSSIVEL CONCLUIR A
    ESTABILIDADE DEVIDO A PRECISAO NUMERICA"
420 STOP
```

DOCUMENTAÇÃO DO PROGRAMA

Nome do programa: "ROUTH".
Variáveis simples:

N – Indica o grau do polinômio característico.
AUX – Indica a estabilidade ou não do sistema.

$$\begin{cases} AUX = 0 & \text{SISTEMA ESTÁVEL} \\ AUX = 1 & \text{SISTEMA NÃO-ESTÁVEL} \end{cases}$$

I – Variável relacionada com a linha da matriz de Routh.
J – Variável relacionada com a coluna da matriz de Routh.
K – Variável contadora, assume valores inteiros de 1 a N + 1.
R$ – Variável que contém uma fileira de caracteres.

$$\text{R\$ pode ser } \begin{cases} \text{R\$ = "SISTEMA ESTÁVEL"} \\ \text{ou} \\ \text{R\$ = "SISTEMA NÃO-ESTÁVEL"} \end{cases}$$

Variáveis indexadas ou matriciais:

B: Matriz de Routh com N + 1 linhas e INT $\left(\frac{N+2}{2}\right)$ colunas.

Entrada de Dados

Deve-se fornecer de maneira seqüêncial os seguintes dados:

1.º) Grau do polinômio característico.
2.º) Os coeficientes do polinômio característico em ordem de potências decrescentes.

DESCRIÇÃO INTERNA DO PROGRAMA

1. As instruções de 30 a 70 permitem que a entrada de dados seja feita na ordem de potências decrescentes.

Note-se que são obedecidas as relações já citadas:

$b_{1,j} = a_{2j-2}$ e $b_{2,j} = a_{2j-1}$.

2. As intruções de 80 a 130 calculam toda a matriz de Routh, se não ocorrer um zero na primeira coluna. Caso ocorra, as linhas restantes da matriz não serão calculadas.

3. As instruções de 140 a 170 verificam se ocorreu inversão de sinal na 1.ª coluna da matriz. Caso ocorra AUX = 1, se não AUX = 0.

4. As instruções de 280 a 340 produzem a impressão de toda a matriz de Routh, se não houver um zero na primeira coluna. Caso haja um zero, a impressão irá até esta linha, inclusive.

5. As instruções de 350 a 410 evitam que ocorra um resultado falso devido a erros de precisão numérica. O exemplo n.º 5 apresentado adiante mostra a necessidade desta precaução.

6. As instruções não citadas na descrição são simples entradas e saídas de dados.

Saída de Dados

O formato da saída de dados é a seguinte:

COEFICIENTES DO POLINÔMIO CARACTERÍSTICO EM ORDEM DE POTÊNCIAS DECRESCENTES:

a_0 a_1 ..... a_{n-1} a_n

GRAU DO POLINÔMIO: n

OS COEFICIENTES DA SÉRIE DE ROUTH SÃO DADOS PELO PRIMEIRO NÚMERO DE CADA BLOCO DE DADOS:

$b_{1,1}$

$b_{1,2}$ $b_{1,3}$

..... $b_{1,m}$

b_{n+1}

$b_{n+1,2}$ $b_{n+1,3}$

$b_{n+1,m}$

RESULTADO: SISTEMA .....

Exemplos:

1.º) $3S^4 + 2S^3 + 5S^2 + 1S + 0.5$

```
GRAU 4
3           5           0.5
2           1           0
3.5         0.5         0
0.714286    0           0
0.5         0           0
SISTEMA ESTAVEL
```

2.º) $3\,S^5 + 2\,S^4 + 5\,S^3 + 1\,S^2 + 0.5\,S$

GRAU 5

3	5	0.5
2	1	0
3.5	0.5	0
0.714286	0	0
0.5	0	0
0	0	0

SISTEMA NAO ESTAVEL

3.º) $6\,S^5 + 5\,S^4 + 4\,S^3 + 3\,S^2 + 2\,S + 1$

GRAU 5

6	4	2
5	3	1
0.4	0.8	0
-7	1	0
0.857143	0	0
1	0	0

SISTEMA NAO ESTAVEL

4.º) $S^5 - 4\,S^4 + 7\,S^3 - 6\,S^2 + 0.001\,S$

GRAU 5

1	7	0.001
-4	-6	0
5.5	0.001	0
-5.99927	0	0
0.001	0	0
0	0	0

SISTEMA NAO ESTAVEL

5.º) $2\,S^6 + 3\,S^5 + 3\,S^4 + {}^3/_2\,S^3 + S^2 + S + {}^5/_9$

2	3	1	0.5555556
3	1.5	1	0
2	0.3333333	0.5555556	0
1	0.1666667	0	0
-6.9849193E-10	0.5555556	0	0
79.5364310	0	0	0
0.5555556	0	0	0

NAO E POSSIVEL CONCLUIR A ESTABILIDADE DEVIDO
A PRECISAO NUMERICA

Um cálculo com toda precisão numérica fornece o seguinte resultado:

2	3	1	5/9
3	3/2	1	0
2	1/3	5/9	0
1	1/6	0	0
0	4	0	0

Sistema não estável.

Isto justifica a precaução tomada no programa (vide descrição interna do programa, item 5).

6.º) $S^6 + S^5 + S^4 + S^3 + 4\,S^2 + 1\,S + 8$

GRAU 6

1	1	4	8
1	1	1	0
0	3	8	0

SISTEMA NAO ESTAVEL

7.º) $10\,S^6 + 27\,S^5 + 27\,S^4 + 27/2\,S^3 + 9\,S^2 + 9\,S + 5$

GRAU 6

18	27	9	5
27	13.5	9	0
18	3	5	0
9	1.5	0	0
0	5	0	0

SISTEMA NAO ESTAVEL

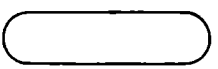
Apêndice A

DIAGRAMAS DE BLOCOS

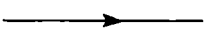
Muitos programadores preferem preparar um programa iniciando com um diagrama de blocos. Esse diagrama, também denominado fluxograma, estabelece a seqüência de operações a se efetuar em um programa.

Esta técnica permite uma posterior codificação, praticamente em qualquer linguagem de programação, pois na elaboração do fluxograma não se atinge detalhamento de instruções ou comandos específicos, os quais caracterizam uma linguagem.

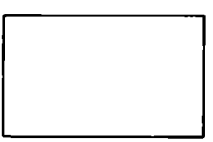
Os símbolos utilizados nos diagramas de blocos são:



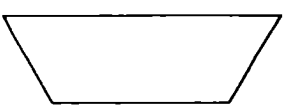
Símbolo de terminal – É utilizado para indicar o início ou o fim de um programa.



Símbolo de fluxo de dados – Indica o sentido do fluxo de dados. Serve para conectar os blocos existentes.

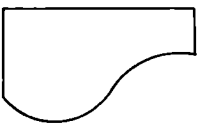


Símbolo de processamento – Bloco que se utiliza para indicar cálculos a efetuar, atribuição de valores ou qualquer manipulação de dados que não tenham um bloco específico para sua descrição.

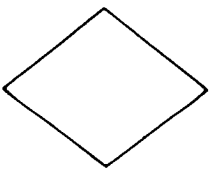


Símbolo de entrada de dados – É utilizado para ler os dados necessários ao programa.

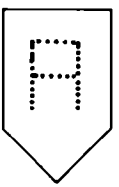
Símbolo de saída de dados em vídeo – Utiliza-se este bloco quando se quer mostrar dados na tela de vídeo.



Símbolo de saída de dados em impressora – É utilizado quando se deseja que os dados sejam impressos.



Símbolo de decisão – Indica a decisão que deve ser tomada, dependendo do resultado de uma comparação.



Símbolo de conexão – É utilizado quando é preciso particionar o diagrama. Quando ocorre mais de uma partição, é colocada uma letra dentro dos blocos de conexão para identificar os pares de ligação.

Estes poucos símbolos, mais um par de símbolos dado mais adiante, facilitam a preparação de programa de qualquer nível de complexidade, desde que sejam seguidas algumas regras:

1.ª) Os diagramas devem ser feitos em vários níveis. Os primeiros devem conter apenas as idéias gerais, deixando para as etapas posteriores os detalhamentos necessários.

2.ª) O fluxograma, sempre que possível, deve ser desenvolvido no sentido de cima para baixo e da esquerda para a direita.

3.ª) Não pode ocorrer cruzamento das linhas de fluxos.

Exemplo:

Seja o problema: Calcular a média de 4 notas e determinar a aprovação ou reprovação.

Considere-se ainda que o valor da média deva ser maior ou igual a 7 para que haja aprovação.

A primeira descrição do problema via diagrama de blocos é dada na Figura A.1.

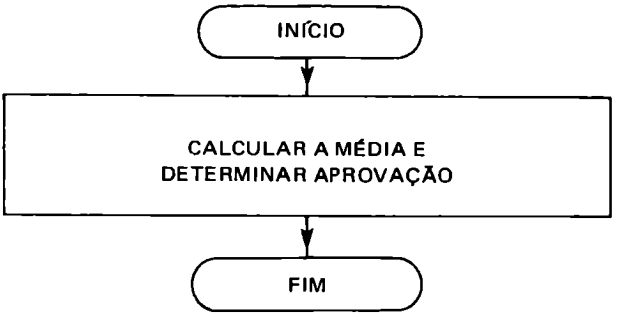


Figura A.1

Na Figura A.2 é apresentado um detalhamento no que se refere à entrada e saída, ou seja, deve-se entrar com as 4 notas e obter, como resultado, uma aprovação ou uma reprovação.

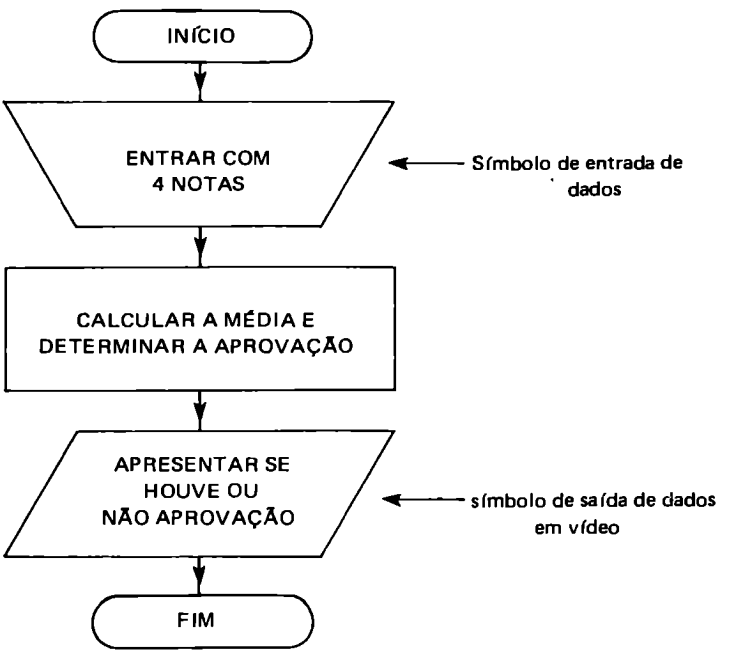


Figura A.2

A frase "determinar a aprovação" envolve uma decisão a ser tomada segundo o resultado da média. Na Figura A.3 é incluído este bloco de decisão.

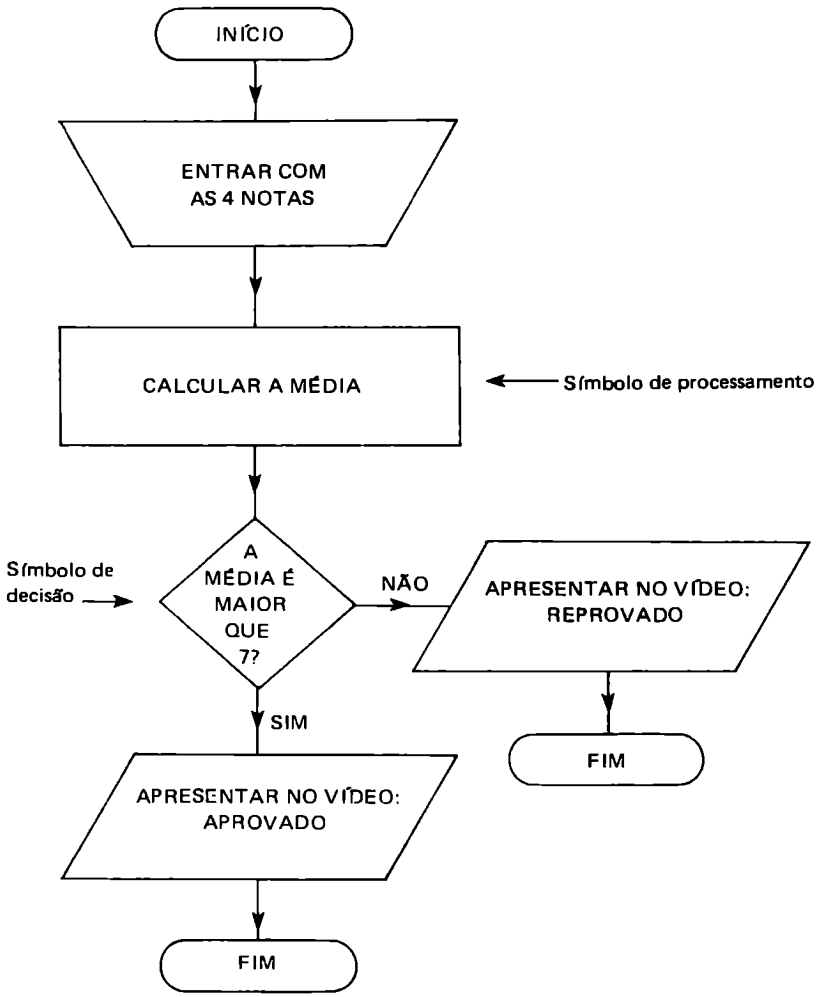


Figura A.3

Muitas vezes é preferível construir o fluxograma trabalhando com variáveis, para adiantar a etapa de codificação do programa.

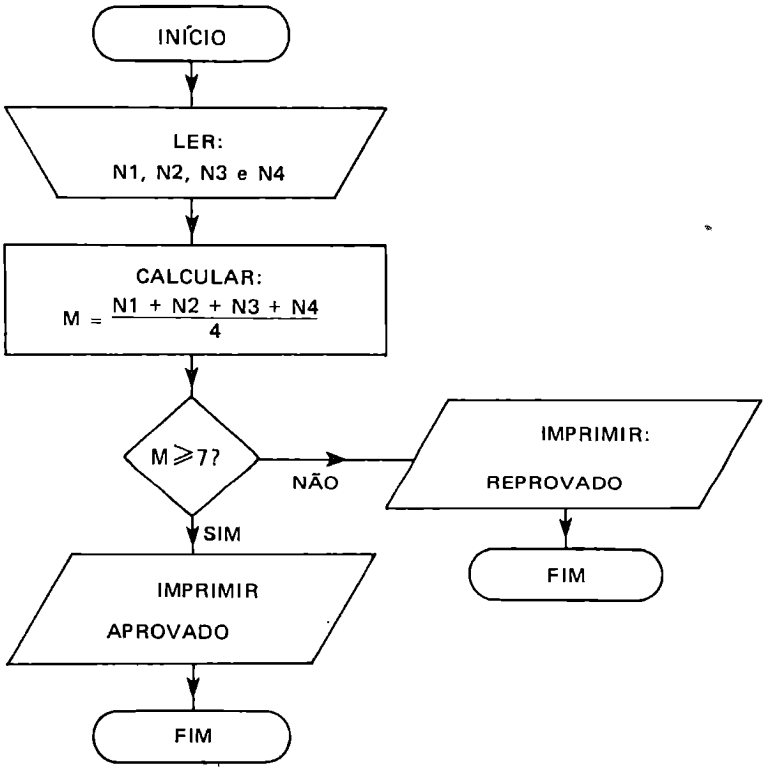


Figura A.4

A codificação em linguagem BASIC a partir da Figura A.4 torna-se imediata e é apresentada abaixo:

```
10 INPUT N1, N2, N3, N4
20 LET M = (N1 + N2 + N3 + N4) / 4
30 IF M > = 7 THEN GO TO 60
40 PRINT "REPROVADO"
50 STOP
60 PRINT "APROVADO"
```

Supondo que, ao executar o programa, sejam fornecidas as notas 7, 6, 6 e 5, será apresentado no vídeo o resultado:

```
REPROVADO
```

Já para outra execução com as notas 8, 7, 9 e 9, o resultado será:

```
APROVADO
```

A repetição controlada de um conjunto de instruções pode ser esquematizada em Diagrama de Blocos, como mostra a Figura A.5.

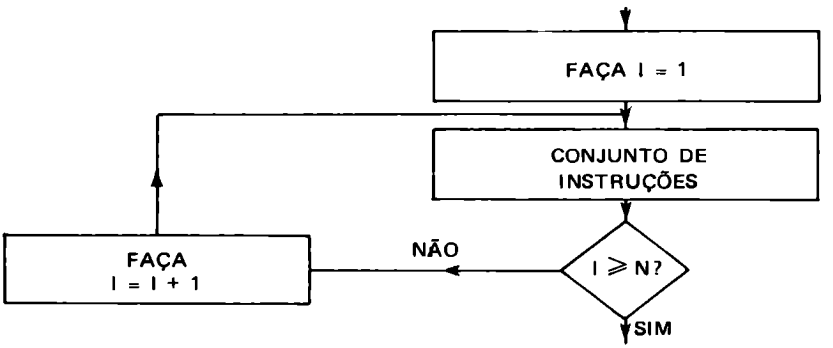


Figura A.5

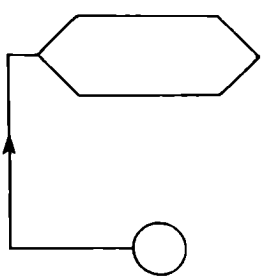
O conjunto de instruções será executado N vezes, isto é, para I igual a 1, 2, 3 ... N.

Nota-se que, qualquer que seja o valor de N, o conjunto de instruções será executado pelo menos uma vez.

Na linguagem BASIC existe um par de instruções que controla a repetição, isto é, inicialmente a variável contadora I, realiza a comparação e incrementa o contador.

Estas instruções vistas no capítulo 4 são a FOR e NEXT.

Para representá-las é utilizado um par de símbolos:



Símbolo de controle – Utilizado para explicitar a variável contadora, o seu valor inicial e o número de repetição.

Símbolo de continuidade – Utilizado para delimitar o conjunto de instruções.

Muitos autores utilizam o símbolo de continuidade como de conector simples, cuja função é evitar o cruzamento de linhas de fluxo dentro de uma mesma página, reservando o símbolo de conexão, adotado neste livro, como conector entre páginas.

Observe-se que, para cada símbolo de controle, deve-se ter obrigatoriamente associado o de continuidade.

Exemplo:

Dado um número inteiro positivo N, calcular o produto $1 \times 2 \times 3 \times \ldots \times N$.

Este produto é denominado fatorial de N, e é representado por "N !".

$3 ! = 1 \times 2 \times 3 = 6$

$5 ! = 1 \times 2 \times 3 \times 4 \times 5 = 120$

O fluxograma para realizar este cálculo é mostrado na Figura A.6:

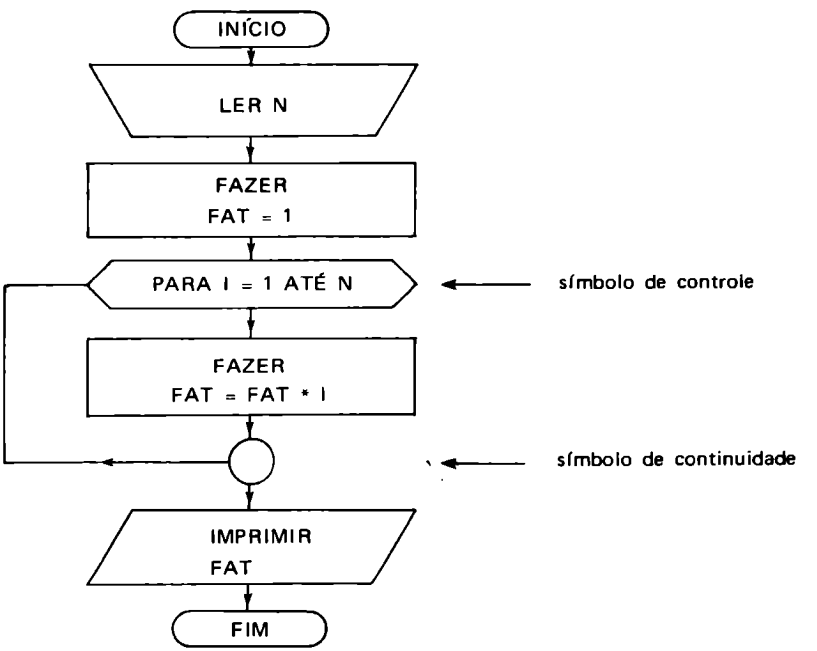


Figura A.6

O bloco FAT = FAT * I representa, no exemplo, o "conjunto de instruções" delimitado pelos símbolos de controle e continuidade.

Se N, por exemplo, valer 3, então teremos a seguinte seqüência de valores de FAT:

	FAT = 1
I = 1	FAT = 1 × 1 = 1
I = 2	FAT = 1 × 2 = 2
I = 3	FAT = 2 × 3 = 6

Uma vez atingido o valor N, no caso 3, o programa sai do anel formado e imprime o valor de FAT.

A codificação em linguagem BASIC e alguns resultados são:

```
10 INPUT N
20 LET FAT = 1
30 FOR I = 1 TO N
40 LET FAT = FAT * I
50 NEXT I
60 PRINT FAT

RUN                              (valor de N)
? 3                              (resultado)
6

RUN
? 5                              (resultado)
120
```

Os exemplos até agora vistos poderiam ser codificados diretamente devido à sua simplicidade. A seguir é apresentado a elaboração de um fluxograma para o cálculo do máximo divisor comum (MDC).

Cálculo do máximo divisor comum de dois números

O MDC de dois números A e B é calculado com o auxílio de uma tabela (Figura A.7), onde A é o maior dos dois números, R_1 é o resto da divisão de A por B, R_2 é o resto da

divisão de B por R_1 e assim sucessivamente. As divisões cessam no momento em que o resto for zero, e o MDC é o divisor da última divisão efetuada.

MDC

A	B	R_1	R_2			R_N
R_1	R_2	R_3			O	

Figura A.7

Alguns exemplos numéricos são mostrados a seguir.

Números: 32 e 28

32	28	4
4	0	

O MDC de 32 e 28 é 4.

Números: 30 e 39

39	30	9	3
9	3	0	

O MDC de 30 e 39 é 3.

Num primeiro nível, o problema é:

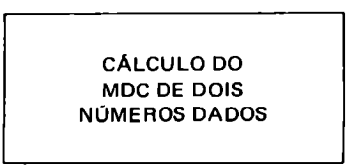


Figura A.8

O que pode ser detalhado gradativamente nos seguintes diagramas:

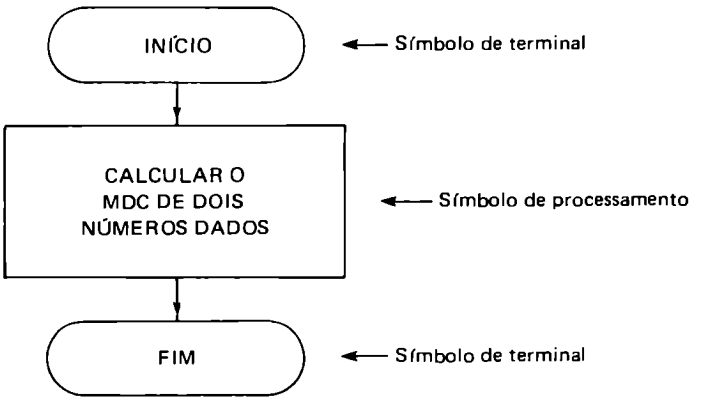


Figura A.9

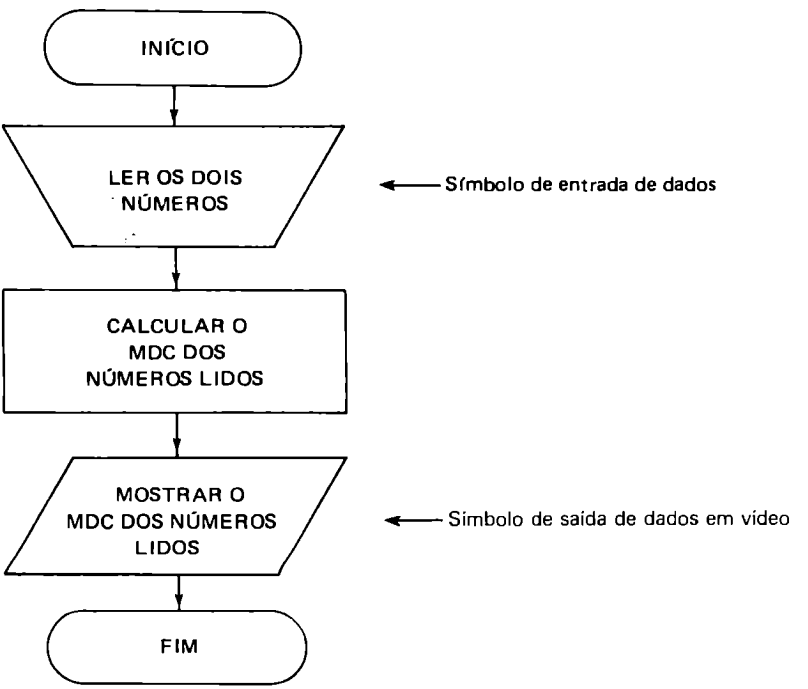


Figura A.10

Um primeiro passo para o cálculo efetivo do MDC é saber qual dos dois números lidos é o maior. Para isto é preciso uma comparação e uma decisão, isto é, atribuir a A o maior dos dois números.

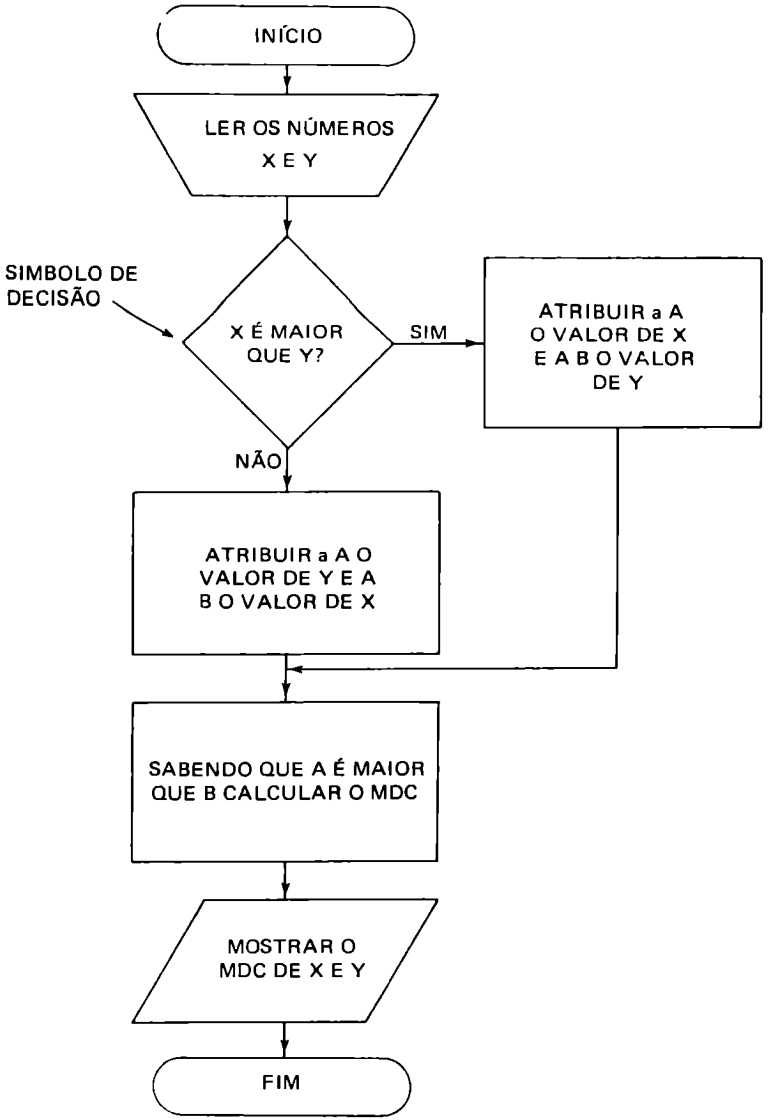


Figura A.11

Utilizando o símbolo de atribuição de valor "=" e o sinal de desigualdade ">"; esse mesmo diagrama da Figura A.11 pode ser apresentado da seguinte forma:

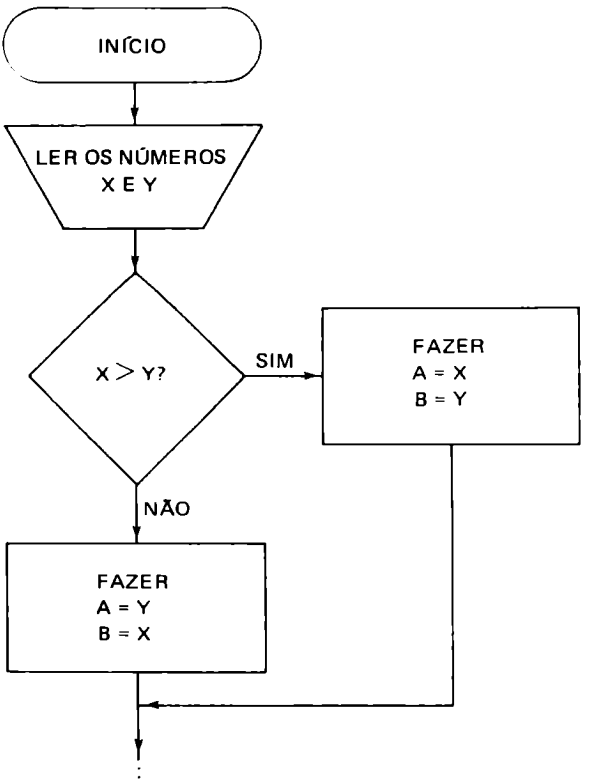


Figura A.12

O bloco que precisa um maior detalhamento é:

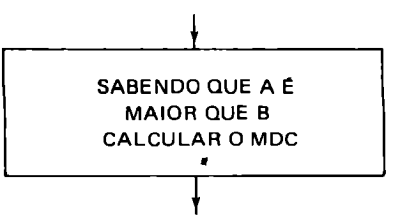


Figura A.13

Como foi visto, para o cálculo do MDC, divide-se A por B, B por R_1 e assim sucessivamente, até que o resto seja zero. Não se sabe, portanto, o número exato de divisões necessárias para a determinação do MDC, porém passo a passo é feita a divisão entre dois números e, dependendo do valor do resto, é tomada uma decisão no sentido de continuar ou não, as divisões.

Um recurso muito utilizado para este tipo de problema é indicar uma única divisão e realizar atribuições de valores convenientes para as divisões posteriores.

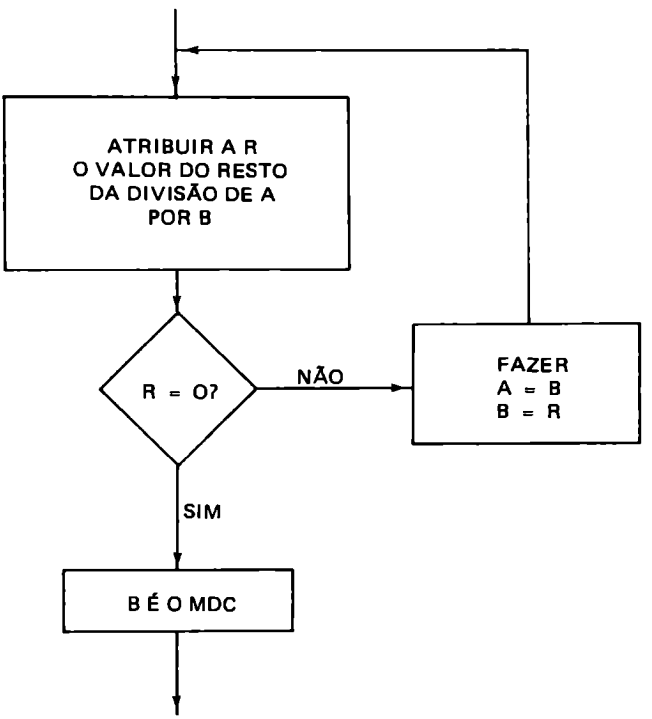


Figura A.14

O diagrama da **Figura A.14** mostra esta idéia, isto é, as divisões só cessam quando o resto é zero, caso contrário são atribuídos novos valores para o dividendo A e o divisor B, conforme a regra já citada.

É apresentado na Figura A.15 o diagrama completo.

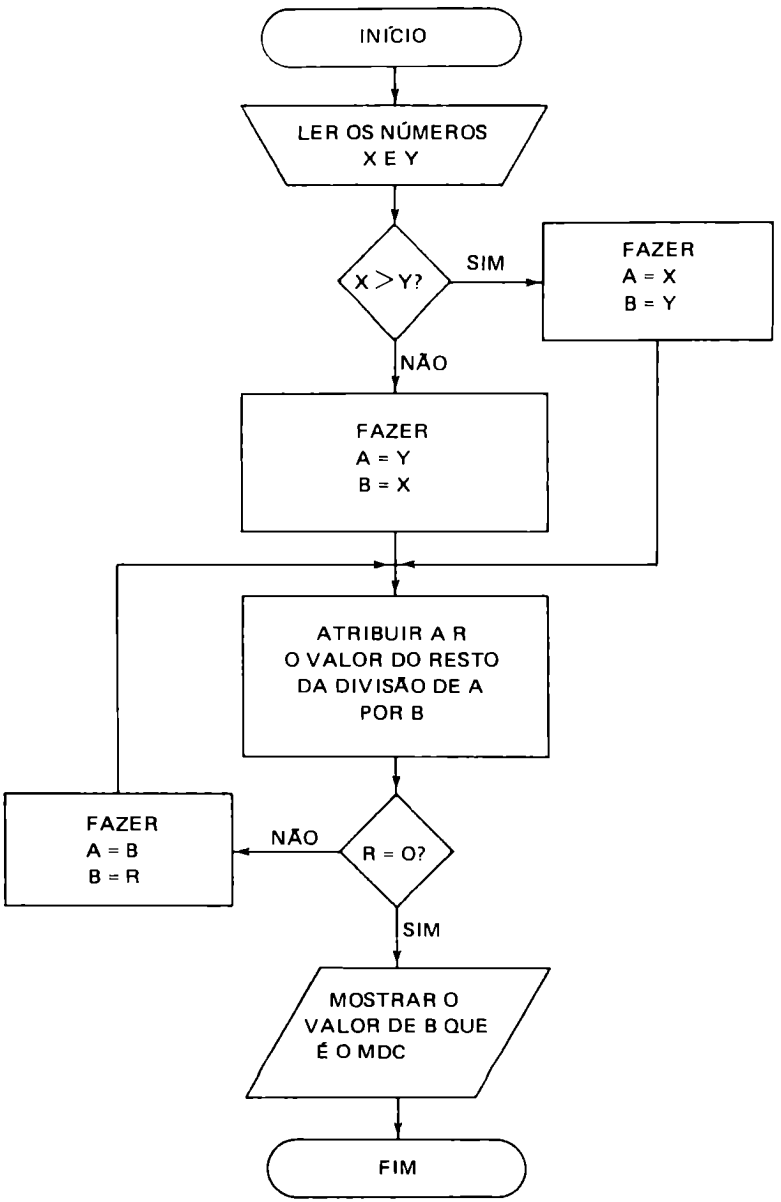
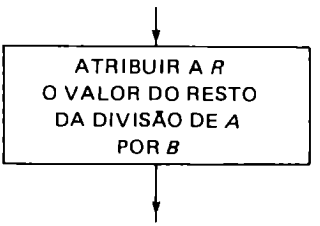


Figura A.15

Para codificar o programa é preciso verificar a existência de uma função que forneça diretamente o resto da divisão de dois números.

Na linguagem BASIC não se encontra esta função como definida, de maneira que mais uma vez deve-se particionar o bloco:



A divisão realizada pelo computador é com números reais, não fornecendo assim resto.

Foi mencionada, no capítulo 2, a existência da função INT (X), que fornece o valor inteiro de X. Utilizando-se esta função é possível obter o resto. Isto fica claro no seguinte exemplo:

$\frac{13}{3} = 4.3333$ (resultado obtido no computador)

INT $(4.333\ldots) = 4$

Na divisão inteira sabe-se que:

Dividendo (A)	Divisor (B)
Resto (R)	Quociente (Q)

$Q \cdot B + R = A \qquad R = A - Q \cdot B$

ou seja, o resto é obtido subtraindo-se do dividendo o quociente multiplicado pelo divisor, e o quociente, por sua vez, é a parte inteira da divisão de A por B.

No exemplo dado o resto R é:

$Q = \text{INT } (13\ 3) = Q = 4$

$R = 13 - 4 \cdot 3 = 1$

Em termos de diagrama de blocos têm-se:

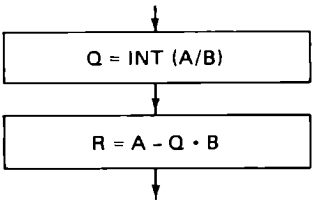


Figura A.16

O diagrama de blocos final para o cálculo do MDC é mostrado na Figura A.17.

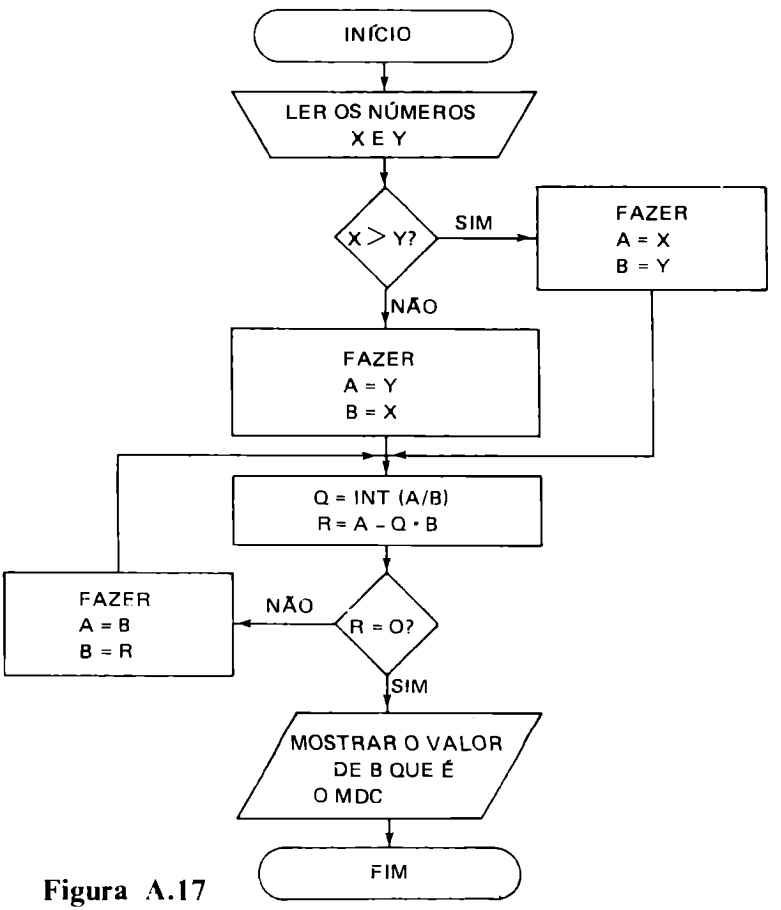


Figura A.17

Com este exemplo do fluxograma para o cálculo do máximo divisor comum, fica claro a conveniência de se resolver o problema por etapas. O fluxograma seria dispensável a pessoas que já tenham alguma experiência em programação, porém, se a complexidade do problema aumenta, mesmo estas pessoas normalmente utilizam a técnica descrita neste apêndice, ou alguma equivalente.

A codificação deste diagrama com as instruções mencionadas no capítulo 3 é:

```
 10 READ X, Y
 20 DATA 32, 28
 30 IF X > Y THEN GO TO 120
 40 LET A = Y
 50 LET B = X
 60 LET Q = INT (A/B)
 70 LET R = A - Q * B
 80 IF R = 0 THEN GO TO 150
 90 LET A = B
100 LET B = R
110 GO TO 60
120 LET A = X
130 LET B = Y
140 GO TO 60
150 PRINT B
160 STOP
```

Resultado:

```
RUN
4
```

Isto significa que o MDC entre 32 e 28, números fornecidos na instrução DATA da linha 20, é 4.

Existem muitas outras maneiras de se calcular o MDC. Por exemplo, o diagrama apresentado na Figura A.18, permite calcular o MDC de dois números sem se utilizar a função INT (X), que pode não existir em alguns microcomputadores.

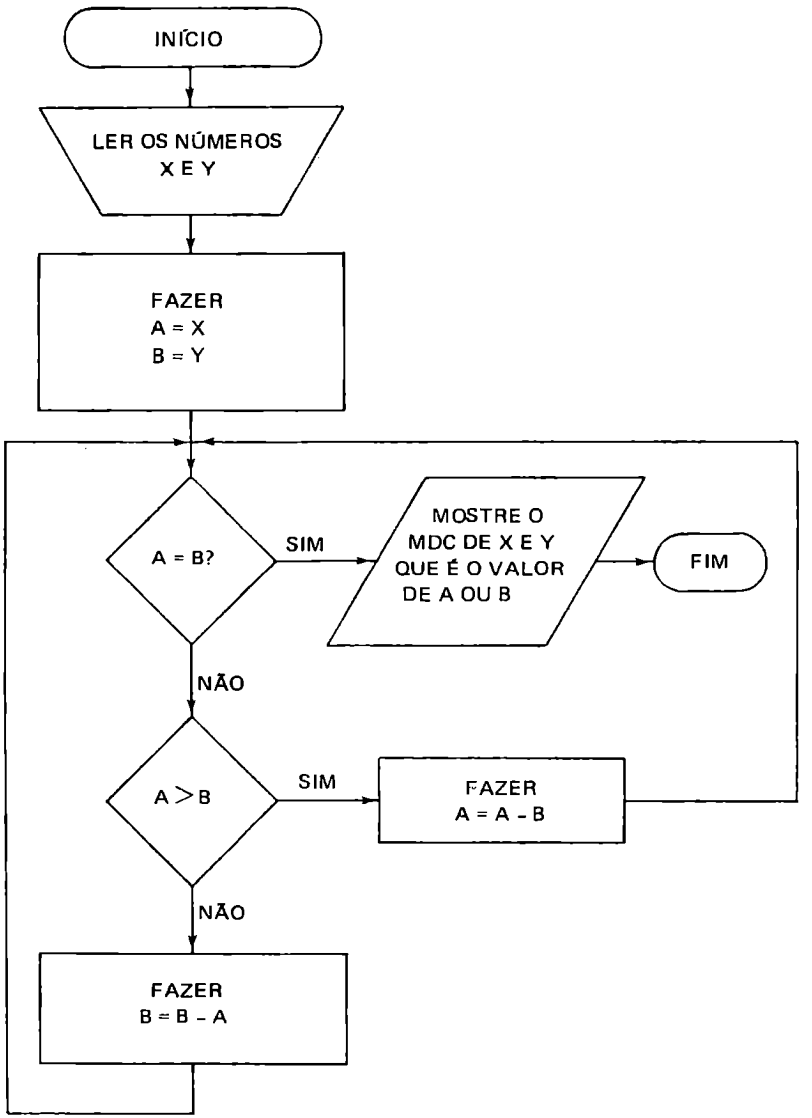
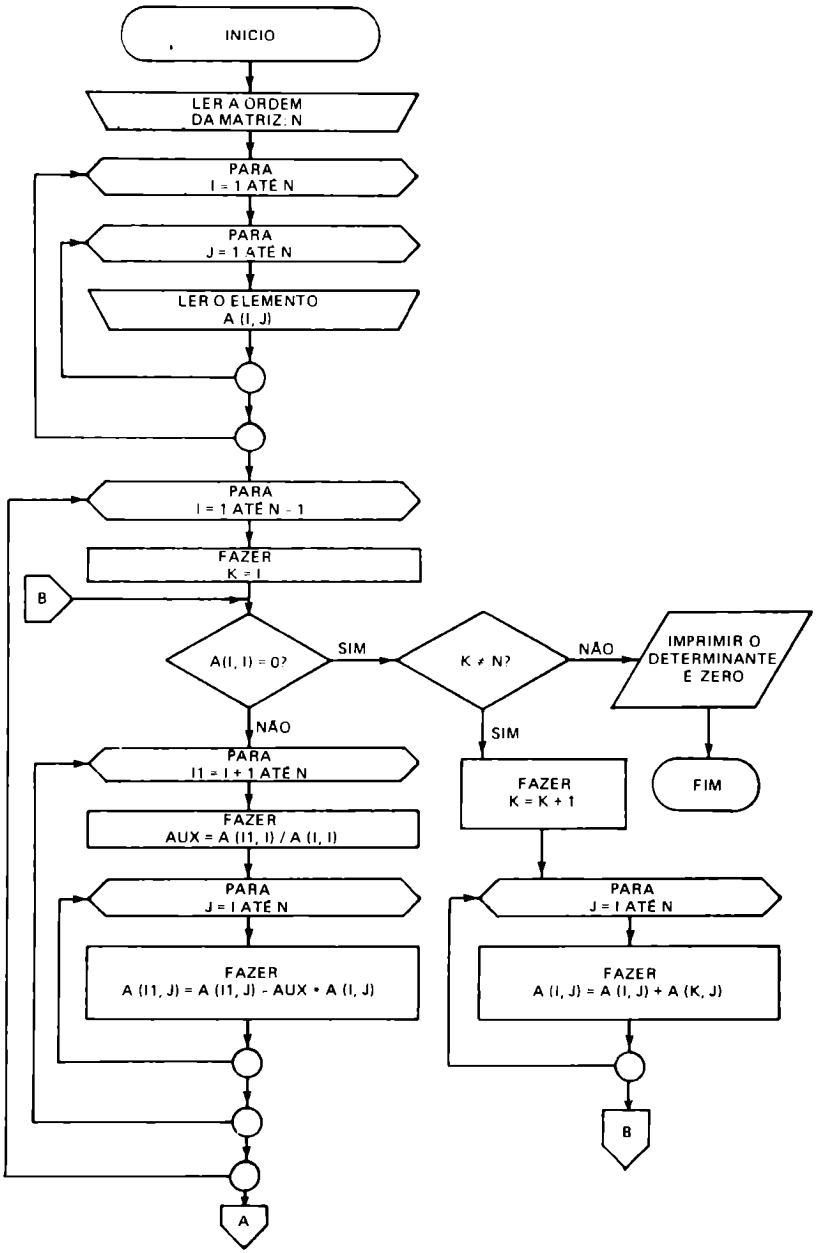
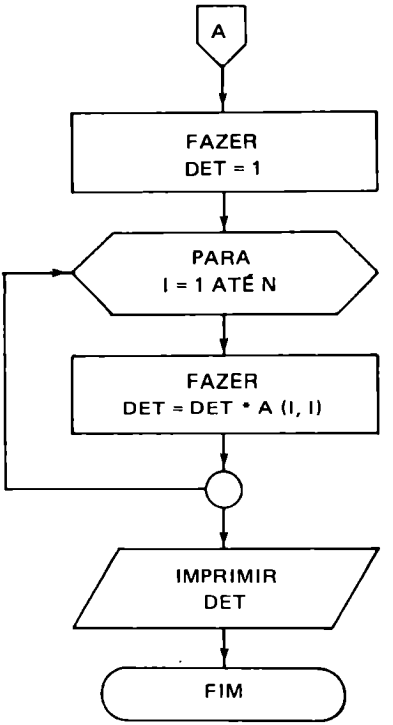


Figura A.18

Segue o fluxograma do cálculo do determinante; o programa correspondente encontra-se no capítulo 6.

FLUXOGRAMA DO CÁLCULO DO DETERMINANTE




A
FAZER
DET = 1
PARA
I = 1 ATÉ N
FAZER
DET = DET * A (I, I)
IMPRIMIR
DET
FIM

Apêndice B

UM JOGO SIMPLES

Neste apêndice é apresentado um programa eficiente para o conhecido "jogo da velha".

Como se sabe, este jogo *termina sempre empatado*, desde que nenhum dos competidores cometa algum erro. O programa apresentado é tal que o computador nunca cometerá erros, e assim, *chegar ao empate bem se pode considerar uma vitória* sobre as possibilidades de falha.

Inicialmente é feito o clássico sorteio para determinar quem começa, e o jogo prossegue conversacionalmente.

Jogo da Velha

Este jogo foi elaborado para ser utilizado nos microcomputadores que possuem tela de vídeo com 32 caracteres por linha. Ao se executar o programa em máquinas que não possuem esta característica, podem ocorrer problemas quanto à apresentação, devendo ser reestruturada a rotina que começa na linha 1 400.

Os espaços em branco são importantes para a formatação, e serão indicados pelo caractere b nos pontos onde pode haver dúvida.

```
 10 REM JOGO DA VELHA
 20 DIM B$ (9)
 30 DIM A (8, 3)
 40 FOR I = 1 TO 9
 50 LET B$ (I) = STR$ (I)
 60 NEXT I
 70 FOR I = 1 TO 3
 80 FOR J = 1 TO 3
 90 LET A (I, J) = (3 * (I - 1) + J)
100 LET A (3 + I, J) = (3 * (J - 1) + I)
```

```
110 NEXT J
120 LET A (7, I) = (4 * (I - 1) + 1)
130 LET A (8, I) = (2 * I + 1)
140 NEXT I
150 GOSUB 1400
160 PRINT "PARA SORTEAR QUEM COMECA APERTE
    A TECLA DE FIM DE LINHA"
170 GOSUB 1600
180 INPUT R$
190 RAND
200 LET S = RND
210 IF S > .5 THEN GOTO 1300
220 GOSUB 1400
230 PRINT "VOCE GANHOU O SORTEIO"
240 PRINT "PODE COMECAR – ENTRE COM O
    NUMERO DA CASA – SUA MARCA SERA O X"
250 GOSUB 1600
260 INPUT N
270 GOSUB 1700
280 LET B$ (N) = "X"
290 IF N = 5 THEN GOTO 320
300 LET B$ (5) = "0"
310 GOTO 330
320 LET B$ (3) = "0"
330 GOSUB 1400
340 PRINT "TAMBEM JA JOGUEI – E A SUA VEZ"
350 GOSUB 1600
360 INPUT N
370 GOSUB 1700
380 LET B$ (N) = "X"
390 FOR I = 1 TO 8
400 LET K = 0
410 FOR J = 1 TO 3
420 IF B$ (A (I, J)) = "0" THEN LET K = K + 1
430 IF K = 2 THEN GOTO 1220
440 NEXT J
450 NEXT I
460 FOR I = 1 TO 8
470 LET K = 0
480 FOR J = 1 TO 3
490 IF B$ (A (I, J)) = "X" THEN LET K = K + 1
```

```
500 IF K = 2 THEN GOTO 1130
510 NEXT J
520 NEXT I
530 IF B$ (1) = "X" AND B$ (9) = "X" THEN GOTO 560
540 IF B$ (3) = "X" AND B$ (7) = "X" THEN GOTO 560
550 GOTO 620
560 FOR I = 2 TO 8 STEP 2
570 IF B$ (I) = STR$ (I) THEN GOTO 600
580 NEXT I
590 GOTO 620
600 LET B$ (I) = "0"
610 GOTO 1180
620 FOR I = 1 TO 3
630 LET M = I + 1
640 IF M < = 3 THEN GOTO 660
650 LET M = M - 3
660 FOR J = 1 TO 3
670 FOR K = 1 TO 3
680 IF B$ (A (I, J)) = "X" AND B$ (A (M, K)
    = "X" THEN GOTO 730
690 NEXT K
700 NEXT J
710 NEXT I
720 GOTO 1000
730 IF J = K THEN GOTO 690
740 IF I < > 1 THEN GOTO 790
750 IF B$ (A (I, K)) = STR$ (A (I, K)) THEN GOTO 770
760 GOTO 690
770 LET B$ (A (I, K)) = "0"
780 GOTO 1180
790 IF I = 3 THEN GOTO 910
800 IF B$ (A (M, J)) = STR$ (A (M, J)) THEN GOTO 820
810 GOTO 690
820 IF J = 2 AND K = 3 THEN GOTO 850
825 IF J = 3 AND K = 1 THEN GOTO 850
830 IF B$ (1) = STR$ (1) THEN GOTO 870
840 IF B$ (9) = STR$ (9) THEN GOTO 890
850 LET B$ (A (M, J) = "0"
860 GOTO 1180
870 LET B$ (1) = "0"
880 GOTO 1180
```

```
890 LET B$ (9) = "0"
900 GOTO 1180
910 IF J = 2 THEN GOTO 960
920 IF B$ (A (I, J) = STR$ (A (I, J)) THEN GOTO 940
930 GOTO 690
940 LET B$ (A (1, J) = "0"
950 GOTO 1180
960 IF B$ (A (3, K) = STR$ (A (3, K) THEN GOTO 980
970 GOTO 690
980 LET B$ (A (3, K)) = "0"
990 GOTO 1180
1000 FOR I = 1 TO 9
1010 IF B$ (I) = STR$ (I) THEN GOTO 1110
1020 NEXT I
1030 GOSUB 1400
1040 PRINT "EMPATAMOS"
1050 PRINT "PARA JOGAR DE NOVO TECLE O NUMERO 1"
1060 GOSUB 1600
1070 INPUT Z
1080 IF Z <> 1 THEN STOP
1090 CLEAR
1100 GOTO 10
1110 LET B$ (I) = "0"
1120 GOTO 1180
1130 FOR L = 1 TO 3
1140 IF B$ (A (I, L)) = STR$ (A (I, L)) THEN GOTO 1170
1150 NEXT L
1160 GOTO 510
1170 LET B$ (A (I, L)) = "0"
1180 FOR I = 1 TO 9
1190 IF B$ (I) = STR$ (I) THEN GOTO 330
1200 NEXT I
1210 GOTO 1030
1220 FOR L = 1 TO 3
1230 IF B$ (A (I, L)) = STR$ (A (I, L)) THEN GOTO 1260
1240 NEXT L
1250 GOTO 440
1260 LET B$ (A (I, L)) = "0"
1270 GOSUB 1400
1280 PRINT "GANHEI"
1290 GOTO 1050
```

```
1300 LET B$ (5) = "0"
1310 GOSUB 1400
1320 PRINT "FUI SORTEADO E JA JOGUEI"
1330 PRINT "ENTRE COM O NUMERO DA CASA – SUA
     MARCA SERA O X"
1340 GOTO 350
1400 CLS
1410 PRINT
1420 PRINT
1430 FOR I = 1 TO 3
1440 PRINT TAB 10;
1450 FOR J = 1 TO 3
1460 PRINT B$ (3 * (I - 1) + J) ; "bbb" ;
1470 NEXT J
1480 PRINT ,, ,,
1490 NEXT I
1500 FOR K = 1 TO 5
1510 PRINT
1520 NEXT K
1530 RETURN
1600 FOR I = 20 TO 43
1610 PLOT I - 3, 36
1620 PLOT I - 3, 30
1630 IF I < 22 THEN GOTO 1660
1640 PLOT 24, I
1650 PLOT 32, I
1660 NEXT I
1670 RETURN
1700 IF N > 9 THEN GOTO 1470
1710 IF N < 1 THEN GOTO 1740
1720 IF B$ (N) <> STR$ (N) THEN GOTO 1740
1730 RETURN
1740 GOSUB 1400
1750 PRINT "JOGADA INVALIDA, TENTE OUTRA"
1760 GOSUB 1600
1770 INPUT N
1780 GOTO 1700
```

O programa é conversacional, de maneira que, ao se executar o programa com o comando RUN, o jogo tem início e os procedimentos são explicados.

ÍNDICE

Referências Bibliográficas

KEMENY, J. G. & KURTZ, T. E. – *BASIC programming*, Wiley, 1967.

SHARPE, W. F. – BASIC – *A introduction to computer programming using the BASIC language*, The Free Press, New York, 1967.

FARINA, M. V. – *Programming in BASIC*, Prentice Hall, 1968.

SPENCER, D. D. – *A guide to BASIC programming*, Addison Wesley, 1970.

CLAUDIO, D. M. & SANTOS, J. A. R. — *Microcomputadores e Minicalculadoras*, Ed. Edgard Blücher Ltda., 1983.

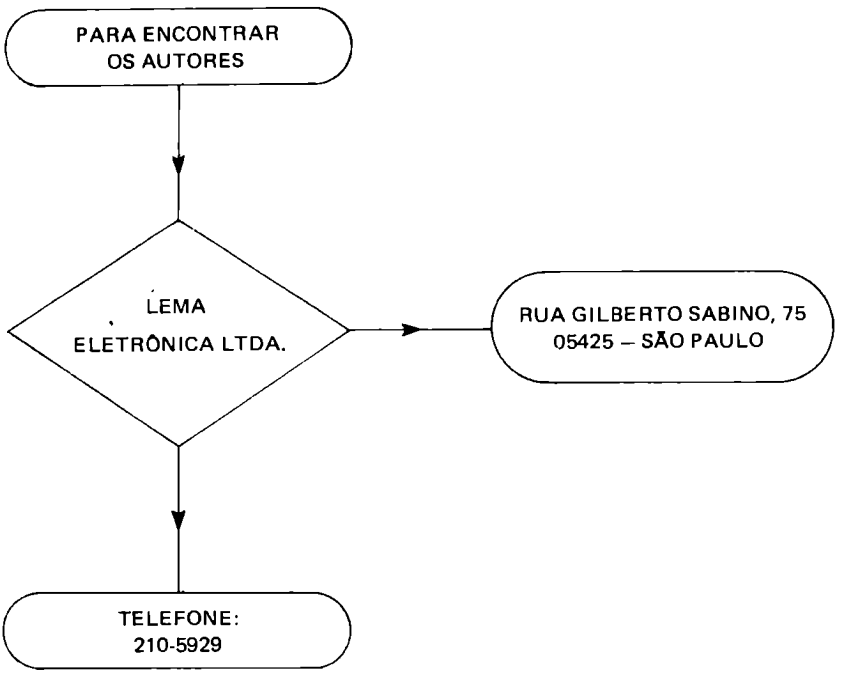
PEREIRA FILHO, J. C. – *BASIC básico*, Ed. Campus, 1982.

SWANSON, R. & CASSEL, D. – *BASIC made easy: a guide to programming micros and minis*, Prentice Hall, 1980.

UEDA, L. M. – *BASIC*, LSD, EPUSP.

PARKER, A. J. & SIBLEY, V. – *BASIC for business*, Prentice Hall, 1980.

KRESCH, R. – *Microcomputadores – Introdução à linguagem BASIC*, Ed. Rio, 1982.


PARA ENCONTRAR
OS AUTORES
LEMA
ELETRÔNICA LTDA.
RUA GILBERTO SABINO, 75
05425 – SÃO PAULO
TELEFONE:
210-5929

impresso na
planimpress gráfica e editora
rua anhaia, 247 - s.p.

ELEMENTOS DE PROGRAMAÇÃO EM BASIC

CARACTERÍSTICAS GERAIS

- Não requer conhecimento anterior de computadores ou programação.
- Escrito com simplicidade e clareza.
- A leitura até o final do capítulo 4 já permite a elaboração de muitos programas.
- Os capítulos 5 e 6 permitem o aperfeiçoamento na linguagem BASIC, incluindo representação gráfica e programas completos de aplicação prática em matemática, economia e engenharia.
- A técnica dos diagramas de blocos ou fluxogramas é apresentada em apêndice, por ser facultativa.
- Um jogo popular é apresentado no apêndice B, para ilustrar o uso do computador em divertimentos.

EDITORA EDGARD BLÜCHER LTDA. 05575