BASIC SINCLAIR

Primeira editora a lançar no Brasil um livro de Linguagem Basic — *Basic Básico* de Jorge da Cunha Pereira Filho — logo seguido por *Basic para Micros* do mesmo autor, a CAMPUS vem, mais uma vez, ao encontro das necessidades do leitor brasileiro, oferecendo-lhe a primeira obra especificamente destinada aos possuidores e usuários de micros da popular Linha Sinclair.

O conteúdo é de fácil compreensão e pode ser rapidamente assimilado sem orientação direta de um professor. Inclui numerosos exemplos, muitas "dicas" e problemas propostos, sendo os assuntos abordados em ordem crescente de dificuldade.

Bastante original é a seção "DESAFIO", ao final dos capítulos, que leva o leitor mais curioso a tentar entender o programa exposto, o qual apresenta, propositadamente, um grau de dificuldade um pouco maior que a do capítulo onde consta.

Alguns apêndices bastante úteis complementam o texto, trazendo um resumo de *todas* as funções e comandos dos micros da Linha Sinclair; um roteiro para entender e contornar os erros de sintaxe; uma sugestão de folha de codificação que facilita a elaboração de programas, o uso da tela e a exposição dos resultados; uma explicação dos principais termos e nomes técnicos existentes na área e de uso corrente nas revistas especializadas.

Raul Udo Christmann, Engenheiro Mecânico com Mestrado em Engenharia Industrial, tem tido extensa e variada atuação na área acadêmica, é Coordenador Regional da Sociedade Brasileira de Pesquisa Operacional no Rio Grande do Sul, sendo também autor de diversos trabalhos já publicados.

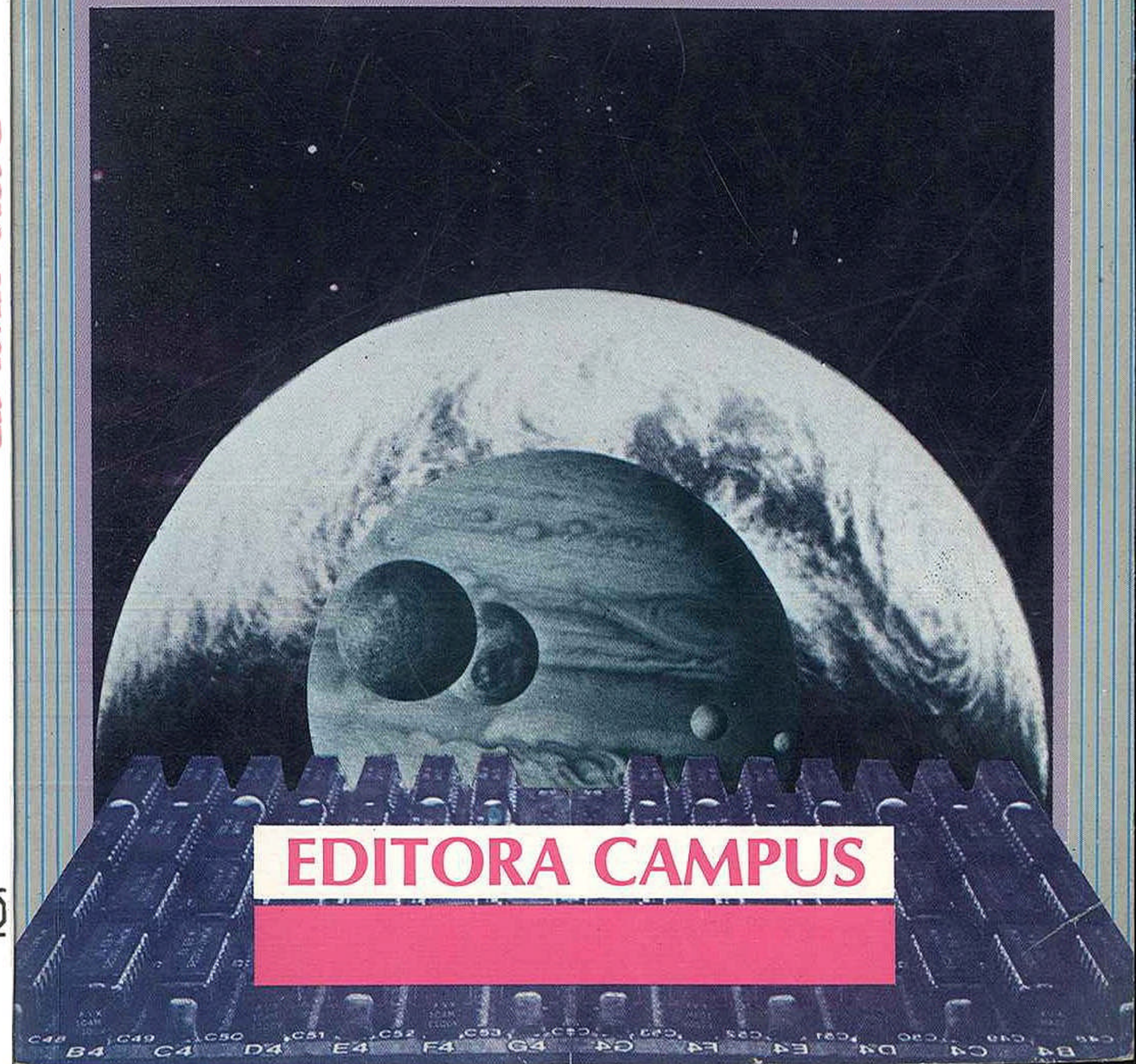
ISBN 85-7001-210-1

Raul Udo Christmann

BASIC SINCLAIR

Raul Udo Christmann

BASIC SINCLAIR



EDITORA CAMPUS

Títulos de Computação da Editora Campus

ADA® ; UMA INTRODUÇÃO – *H. Ledgard*

ADMINISTRAÇÃO DE SISTEMAS DE INFORMAÇÃO – *P. Ein-Dor & E. Segev*

ANÁLISE DE SISTEMAS PARA SISTEMAS DE INFORMAÇÃO POR COMPUTADOR – *J. C. Wetherbe*

APLICAÇÕES DE MICROPROCESSADORES – *J. A. Kuecken*

APLICAÇÕES DO COMPUTADOR NA MEDICINA – *N. F. Kember*

APLICALC; UM SOFTWARE EDUCACIONAL, PESSOAL E PROFISSIONAL EM BASIC – *E. A. Meili*

BASIC BÁSICO, 4ª ed. – *J. C. Pereira Filho*

BASIC PARA APLICAÇÕES COMERCIAIS – *D. Hergert*

BASIC PARA MICROS PESSOAIS, 2ª ed. – *J. C. Pereira Filho*

BASIC SINCLAIR – *R. U. Christmann*

BRINCANDO COM O COMPUTADOR – *J. A. Moreira*

COBOL PARA ESTUDANTES, 4ª ed. – *A. Parkin*

COMO LIDAR COM O COMPUTADOR – *H. C. Lucas Jr.*

COMPUTADORES BRASILEIROS; INDÚSTRIA, TECNOLOGIA E DEPENDÊNCIA – *P. B. Tigre*

COMPUTADORES PARA USUÁRIOS – *J. C. Pereira Filho, Coordenador*

Vol. I – Aplicação de Computadores
Vol. II – Equipamentos e Sistemas de Computação
Vol. III – Programas e Programação de Computadores
Vol. IV – Seleção de Sistemas de Computação

CONCEITOS DE PROCESSAMENTO DE DADOS COM BASIC – *R. A. Stern & N. Stern*

A CONSTRUÇÃO DE UM COMPILADOR – *V. W. Setzer & I. S. H. Melo*

CRIANÇA TAMBÉM FAZ PROGRAMAS, 2ª ed. – *J. A. Moreira*

DESAFIO. OS MAIS EXCITANTES JOGOS EM BASIC – *A. J. L. Botelho*

DOCUMENTAÇÃO DE SOFTWARE – *J. D. Lomax*

ESPECIFICAÇÃO DE SISTEMAS – *S. J. Waters*

ESTRUTURAS DE DADOS, 2ª ed. – *P. S. Veloso et al.*

FUNDAMENTOS DE ESTRUTURAS DE DADOS – *E. Horowitz & S. Sahni*

FUNDAMENTOS DE PROCESSAMENTO DE DADOS – *W. T. Price*

GERÊNCIA DE BASES DE DADOS PARA MICROCOMPUTADORES – *E. G. Brooner*

GRAFOS E ALGORITMOS COMPUTACIONAIS – *J. L. Szwarcfiter*

GUIA DE LINGUAGENS DE COMPUTADORES – *H. L. Helms Jr.*

GUIA PARA PROGRAMADORES, 2ª ed. – *M. Bohl*

IMPLANTAÇÃO DE MICROS E MINICOMPUTADORES COMERCIAIS – *P. A. Knight*

INTRODUÇÃO À ARQUITETURA E ORGANIZAÇÃO DE COMPUTADORES – *H. Lorin*

INTRODUÇÃO À CIÊNCIA DA COMPUTAÇÃO COM WATFIV E FORTRAN, 2ª ed. – *S. E. R. Carvalho*

INTRODUÇÃO À PROGRAMAÇÃO COM PASCAL, 2ª ed. – *S. E. R. Carvalho*

INTRODUÇÃO À PROGRAMAÇÃO DE COMPUTADORES, 3ª ed. – *H. V. R. Corrêa Silva et al.*

INTRODUÇÃO À PROGRAMAÇÃO FORTRAN – *J. C. Pereira Filho*

INTRODUÇÃO À SEGURANÇA DO COMPUTADOR – *M. B. Wood*

INTRODUÇÃO A SISTEMAS DE BANCOS DE DADOS – *C. J. Date*

INTRODUÇÃO AO PROCESSAMENTO DE TEXTOS – *G. L. Simons*

INTRODUÇÃO AO VISICALC, 2ª ed. – *E. A. Garbin*

JCL/SISTEMA 370, 3ª ed. – *G. D. Brown*

LCP – LÓGICA DE CONSTRUÇÃO DE PROGRAMAS, 2ª ed. – *J. D. Warnier*

LCS – LÓGICA DE CONSTRUÇÃO DE SISTEMAS – *J. D. Warnier*

LDR – GUIA DOS USUÁRIOS DE SISTEMAS DE INFORMAÇÃO – *J. D. Warnier*

LINGUAGEM DE PROGRAMAÇÃO ALGOL, 2ª ed. – *L. M. Segre*

LINGUAGEM PASCAL – *V. L. Strube de Lima*

MANUAL DE CP/M INCLUINDO MP/M – *R. Zaks*

MANUAL DE COBOL ESTRUTURADO – *D. D. McCracken*

MANUAL DO MICROCOMPUTADOR Z-80 – *W. Barden Jr.*

MANUAL DE LINGUAGEM C – *L. Hancock e M. Krieger.*

MATEMÁTICA PARA MICROCOMPUTADORES – *W. Barden Jr.*

MICRO-MINICOMPUTADORES BRASILEIROS – *E. L. Passos*

MICROCOMPUTADORES PARA APLICAÇÕES COMERCIAIS – *W. Barden Jr.*

MICROPROCESSADORES DE 16 BITS – *C. A. Titus et al.*

MICROPROCESSADORES/MICROCOMPUTADORES – *A. J. Khambata*

Vol. I – Arquitetura Vol. II – Software e Sistemas

1001 APLICAÇÕES PARA O SEU COMPUTADOR PESSOAL – *M. Sawusch*

ORGANIZAÇÃO DA INFORMÁTICA NA EMPRESA – *E. Rios*

ORGANIZAÇÃO DE BANCOS DE DADOS, 4ª ed. – *A. L. Furtado e C. S. Santos*

PRINCÍPIOS DE SISTEMAS OPERACIONAIS, 3ª ed. – *C. C. Guimarães*

PRODUTIVIDADE DO PROGRAMADOR – *L. J. Arthur*

PROGRAMAÇÃO EM ASSEMBLER E LINGUAGEM DE MÁQUINA – *D. C. Alexander*

PROGRAMAÇÃO ESTRUTURADA EM COBOL – *A. B. Furtado*

PROGRAMAÇÃO SISTEMÁTICA EM PASCAL, 2ª ed. – *N. Wirth*

PROGRAMAS ADMINISTRATIVOS EM BASIC SINCLAIR – *L. Karsten*

RPG-II – *J. C. Pereira Filho*

REDES DE COMPUTADORES; ASPECTOS TÉCNICOS E OPERACIONAIS – *D. A. Menascé e D. Schwabe*

ROTINAS MATEMÁTICAS EM BASIC PARA MICROS – *C. R. A. Loiola*

O SEU COMPUTADOR PESSOAL – *M. Waite e M. Pardee*

SISTEMA OPERACIONAL UNIX – *K. Christian*

SISTEMAS DE VIDEOCASSETE; TEORIA E MANUTENÇÃO – *G. P. McGinty*

SISTEMAS OPERACIONAIS PARA MICROCOMPUTADORES – *M. Dahmke*

TÉCNICAS E PRÁTICA DE PROGRAMAÇÃO – *A. Chantler*

TECNOLOGIA DA INFORMAÇÃO: UM GUIA PARA EMPRESAS, GERENTES E ADMINISTRADORES – *J. Eaton e J. Smithers*

30 PROGRAMAS EM BASIC PARA COMPUTADORES PESSOAIS – *D. Chance*

USANDO CP/M: UM GUIA EM ENSINO PROGRAMADO – *J. N. Fernandez & R. Ashley*

MICROBITS

Macroinformações e Programas para Micros TK 82-83-85, CP-200 e compatíveis.
Uma publicação bimestral com análises de hardware, aplicativos, artigos sobre linguagem de máquina, dicas de programação e respostas para suas dúvidas.

MICROINFORMÁTICA INTERNACIONAL

Uma publicação bimestral sobre Segurança e Prevenção de Fraudes em Computadores, Aplicação de Microprocessadores e Microcomputadores e Microeletrônica.
Assinaturas: 10 ORTN's

Procure nossas publicações nas boas livrarias ou comunique-se diretamente com:

EDITORA CAMPUS LTDA.
Livros Científicos e Técnicos
Rua Barão de Itapagipe, 55
20261 Rio de Janeiro – RJ – Brasil
Telefone: (021) 284 8443

Atendemos também pelo reembolso postal.

BASIC SINCLAIR

Raul Udo Christmann

BASIC SINCLAIR

EDITORA CAMPUS LTDA.
Rio de Janeiro

© 1985, Editora Campus Ltda.

Todos os direitos reservados e protegidos pela Lei 5988 de 14/12/1973.
Nenhuma parte deste livro poderá ser reproduzida ou transmitida sejam quais forem os meios empregados: eletrônicos, mecânicos, fotográficos, gravação ou quaisquer outros.

Todo o esforço foi feito para fornecer a mais completa e adequada informação. Contudo a editora não assume responsabilidades pelo uso da mesma.

A Editora Campus não é filiada a nenhum fabricante de sistemas computacionais.

Capa
Otavio Studart

Diagramação, composição, paginação e revisão
Editora Campus Ltda.
Rua Barão de Itapagipe 55 Rio Comprido
Tel.: (021) 284-8443
20261 Rio de Janeiro RJ Brasil
Endereço telegráfico: CAMPUSRIO

ISBN 85-7001-210-1

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

Christmann, Raul Udo.
C479b BASIC Sinclair / Raul Udo Christmann. — Rio de Janeiro : Campus, 1985.

Apêndices
Bibliografia
ISBN 85-7001-210-1

1. BASIC (Linguagem de programação para computadores) I. Título

CDD — 001.6424
84-0799 CDU — 800.92BASIC

Ao meu filho
Jean Marc

SUMÁRIO

1 APRESENTAÇÃO

O que é BASIC?

O nome BASIC é abreviação de "Beginner's All-purpose Symbolic Instruction Code", que traduzido fica: *Código de Instrução Simbólica de uso Geral para Principiantes.*

Esta linguagem foi desenvolvida pelos professores de Dartmouth (EUA), John G. Kemeny e Thomas Kurtz, em 1960.

É uma linguagem de uso geral, com um mínimo de formalidade quanto à sintaxe, e pode ser utilizada tanto nas áreas humanas como científicas e técnicas.

A linguagem BASIC obteve êxito maior que os autores poderiam supor, e encontra-se disponível em todos os computadores pessoais, também conhecidos por microcomputadores.

Embora a linguagem BASIC apresente um grande trunfo — sua facilidade de aprendizagem e utilização — tem algumas restrições: opera lentamente e comporta muitas versões. Cada equipamento tem a sua.

Devido à última restrição, esta obra dedica-se exclusivamente ao BASIC da linha Sinclair, uma das mais populares do mundo.

A BASIC foi projetada de tal maneira que um iniciante pode aprendê-la pelo método da tentativa-e-erro. As mensagens de erro são imediatas e simples. As alterações são fáceis de fazer e testar.

Para manter esta característica, a BASIC precisa ser *interpretada,* o que significa que o programa acessa rotinas em linguagem de máquina a cada linha, sendo este o fato que torna essa linguagem mais lenta que outras disponíveis em sistemas maiores.

Alguns computadores pessoais dispõem de BASIC compilado, onde o programa inteiro é convertido de uma só vez para linguagem de máquina. Se por um lado é mais trabalhoso depurar um programa compilado (é preciso alterar o programa-fonte, compilá-lo outra vez, e então testá-lo), em compensação seu processamento é muito mais rápido.

A filosofia da linguagem BASIC não é a da rapidez, mas sim a da facilidade de uso, compreensão e programação. Estas são as razões do seu sucesso.

ELEMENTOS BÁSICOS DA LINGUAGEM

2.1 COMPONENTES DA LINGUAGEM

Um programa em BASIC é formado pelos seguintes componentes:

- Símbolos
- Identificadores
- Números
- Palavras reservadas, comandos ou instruções
- Cadeias de caracteres (STRING)

2.2 SÍMBOLOS BÁSICOS

Os principais símbolos são:

- Letras
- Dígitos
- Delimitadores
- Branco
- Outros

As letras são:

A, B, C, D, E, F, G, H, I, J, K, L, M, N, O, P, Q, R, S, T, U, V, W, X, Y, Z

Os dígitos são:

Ø, 1, 2, 3, 4, 5, 6, 7, 8, 9

Os delimitadores são:

( ) + − = * / , ; : <=>=<>

AND OR NOT

O espaço em branco (representado por " ") é considerado como um caractere, da mesma maneira que uma letra ou um dígito.

2.3 NÚMEROS

O computador só "entende" números num sistema chamado binário, que corresponde a uma seqüência de zeros e uns (vide 10.2). Felizmente, o computador vem da fábrica com um programa chamado *interpretador,* que converte os nossos números decimais nos binários que ele entende, e vice-versa.
Todos os números são do tipo chamado *reais* ou de *ponto flutuante.* São os números que permitem o uso da vírgula (ponto em BASIC).

NÚMERO DECIMAL	EM BASIC
3,45	3.45
0,25557	Ø.25557 ou .25557
23456780	23456780
1789,00	1789

Os números também podem ser representados em *notação científica:*

3.45 E1 = $3{,}45 \times 10^{1}$ = 3,45
3.45 E2 = $3{,}45 \times 10^{2}$ = 34,5
3.45 E3 = $3{,}45 \times 10^{3}$ = 345
3.45 E7 = $3{,}45 \times 10^{7}$ = 34500000
3.45 E-2 = $3{,}45 \times 10^{-2}$ = 0,0345

Os números são expostos na tela da TV ou impressos na impressora com os oito dígitos mais significativos, embora, internamente, trabalhe com nove ou dez dígitos.

INTERNAMENTE	EXTERNAMENTE
3.4876508967	3.4876509
−Ø.ØØØØ12345678	−Ø.ØØØØ12345678

O maior número inteiro que pode ser armazenado ou manipulado é:

$2^{32} - 1 = 4294697ØØ$

2.4 IDENTIFICADORES

A BASIC permite dar nomes às variáveis — os chamados *identificadores* —, o que facilita a compreensão de um programa.

Numa calculadora podemos "guardar" um número e depois recuperá-lo com um simples apertar de teclas. No computador, damos um nome a esse número (qualquer nome) e o computador se encarrega de preservar o número juntamente com o nome que foi dado. Quando, mais tarde, precisarmos desse número, basta pedi-lo pelo nome.

Este nome, ou identificador (identificador do número), pode ser:

X	SOMA
NÚMERO	MÉDIA
ZE	A2
N25	VALOR DA COMPRA

Um identificador pode:

- Ser alfanumérico: (combinar letras e números) — N25, H235
- Ter espaços: — VALOR DA COMPRA

Um identificador não pode:

- Começar por um número: 5A
- Começar por outro símbolo que não seja uma letra.

Existem outras restrições, que serão mencionadas oportunamente.

2.5 OPERADORES ARITMÉTICOS

Os operadores aritméticos combinam números e variáveis, formando as expressões aritméticas.
Estão disponíveis os seguintes operadores:

OPERADOR	SIGNIFICADO
+	adição
–	subtração
*	multiplicação
/	divisão
**	exponenciação

Algumas observações são necessárias:

a) Entre os operadores, a prioridade de cálculo é:

1) Exponenciação
2) Multiplicação e Divisão
3) Adição e Subtração

Assim,

20 – 3 * 5 + 8 * * 2

É executado da seguinte forma:

```
20 – 3 * 5 + 64
20 –  15   + 64
   5       + 64
        69
```


b) Se uma expressão aritmética contiver vários operadores de mesma prioridade, estes são executados da *esquerda para a direita.*

```
4 * 2  * 3
  8    * 3
      24
```

c) A prioridade pode ser alterada com o uso de parênteses. Os operadores entre parênteses são executados em primeiro lugar:

```
5 *  (20 – 3) + 64
5 *     17     + 64
   85          + 64
         149
```

d) Dois operadores nunca podem aparecer juntos. É necessário o uso de parênteses, no caso de variáveis negativas. Exemplos:

```
A  *  (– B) + E
8  /  (– 2) * 1.23
5  ** (– 2)
```

e) Exponenciação com base negativa gera um número negativo:

– 5 ** 2 = – 25

2.6 FUNÇÕES ARITMÉTICAS

A linguagem BASIC dispõe de um conjunto de funções predefinidas, que facilitam muito a elaboração de um programa. As principais são:

FUNÇÃO	SIGNIFICADO	EXEMPLO
SQR	raiz quadrada	SQR 9
ABS	valor absoluto (1)	ABS VAL
LN	logaritmo natural (2)	LN 45.27
PI	pi = 3,141592653	2 * pi * R
INT	função inteiro (3)	INT 3.8
EXP	função exponencial (4)	EXP 4.5
SGN	função sinal (5)	SGN VALOR
RND	número aleatório (6)	A = RND * 10
SIN	seno (7)	SIN 0.675
COS	co-seno (7)	COS B4
TAN	tangente (7)	TAN 30
ASN	arco seno (7)	ASN 0.675
ACS	arco coseno (7)	ACS ÂNGULO
ATN	arco tangente (7)	ATN VALOR

Tabela 1. Lista das principais funções

(1) Converte números negativos em positivos. **ABS**

(2) Também conhecido por logaritmo neperiano ou de base e = 2,7182. **LN**
Para obtenção do logaritmo decimal, conhecido por *LOG* ou logaritmo natural, basta dividi-lo por *LN 10.*

LOG 4,287 = LN 4.287 / LN 1Ø

(3) Os números sempre são arredondados *para baixo* assim **INT**

INT 3.9 = 3
INT −4.2 = − 5

Para conseguir um arredondamento *para o inteiro mais próximo* basta adicionar Ø.5.

INT (3.9 + Ø.5) = INT 4.4 = 4
INT (3.2 + Ø.5) = INT 3.7 = 3
INT (−4.2 + Ø.5) = INT −3.7 = − 4

(4) Corresponde ao número e elevado a alguma potência. **EXP**

EXP 4.5 = E ** 4.5 = $2{,}7182^{4{,}5}$

EXP e LN são funções inversas entre si.

(5) Traz como resultado os valores **SGN**
− 1, Ø, ou 1
Se o argumento da função for positivo, nulo ou negativo.

SGN 38 → 1
SGN − 4.8 → − 1
SGN Ø → Ø

(6) Esta função gera um número aleatório que varia entre 0.00000000 e 0.99999999, como: **RND**

Ø.45678
Ø.99654
Ø.ØØ45

Para a obtenção de números aleatórios que variam entre Ø e 1ØØ, por exemplo, basta fazer:

RND * 1ØØ

Resultando em:

45.678
99.654
Ø.Ø45

Para a obtenção de números aleatórios inteiros entre 1 e 6, para simular o lançamento de um dado, por exemplo, é suficiente fazer:

INT (RND * 6) + 1

Assim,

Ø.45678 → INT (Ø.45678 * 6) + 1
→ INT (2.74Ø6) + 1
→ 3

Ø.97654 → INT (Ø.99654 * 6) + 1
→ INT (5.97924) + 1
→ 6

Ø.ØØ45 → INT (Ø.ØØ45 * 6) + 1
→ INT (Ø.ØØ27) + 1
→ 1

Para a obtenção de duas seqüências de números aleatórios exatamente iguais, é necessário usar a palavra-chave < *RAND* > e um número entre 1 e 65536, que as RND'S obtidas a seguir geram a mesma seqüência de números. Assim, **RAND**

RAND 345 → RND → Ø.39595Ø32
RND → Ø.69696Ø45
RND → Ø.27236939
RND → Ø.42852783

(7) Estas funções trigonométricas trabalham sempre em radianos. Para se obterem graus é necessário multiplicar o valor em radianos por *pi* e dividir por 180. **SIN COS TAN ASN ACS ATN**

Tangente 45 graus → TAN (45 * PI / 18Ø)
→ 1

2.7 EXERCÍCIOS

1) A expressão aritmética

$$\frac{3{,}45 + 5}{2} \times A,$$

Escrita em BASIC, fica: (3.45 + 5/2 *A. Faça o mesmo para as seguintes expressões:

a) $-\frac{A + C}{B}$

b) COS (A + 3,45)

c) $\dfrac{D + F}{C + \dfrac{U}{G + H}}$

d) $\dfrac{VALOR - 4}{3,2}$

e) $\left| \dfrac{X - MÉDIA}{N - 1} \right|^2$

f) $\dfrac{HORA\ X\ 60 + MINUTOS}{VELOCIDADE}$

g) $\dfrac{A}{B.C} + \dfrac{1964}{B}$

h) 2R SENO U/2

i) $2 \quad 2A - 4AC$

j) $\dfrac{A + B}{C + D} + x^2$

k) $1 + X + \dfrac{X^2}{2!} + \dfrac{X^3}{3!}$

l) LOG $(X + X^2 + 1)$

2) Sendo A = 2.1, B = 3.4, C = 5 e D = 6.
Calcular:

a) C + B/D + A

b) B * C / D – A

c) (A + B) / B + C

d) ABS (A – B) * SQR C

e) INT (D/C) + PI

f) B * SGN (C – D)

g) LN A + B ** C

h) LOG B + INT B *(A + C)

i) SQR ((B ** 2 – A * A * C) / – (2 * A))

j) $\dfrac{A + B}{C + D}$ + A ** 2 – (COS B * (A/C)) + 2 * PI

2.8 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
  1 REM )))PROGRAMA 1(((
 10 FOR A=2 TO 120
 20 LET B=A*PI/60
 30 PLOT A/2,SIN (B)*20+20
 40 PLOT A/2,COS (B)*20+20
 50 NEXT A
```

```
  1 REM )))PROGRAMA 2(((
 10 FOR A=1 TO 400
 20 LET B=PI*A/50
 30 LET C=(400-A)/400
 40 PLOT (20.5*COS (B)+30)*C,(2
0*SIN (B)+20)*C
 50 NEXT A
```

```
  1 REM )))PROGRAMA 3(((
 10 LET C=62
 20 LET D=40
 30 LET A=C*RND
 40 LET B=D*RND
 50 PLOT A,B
 60 PLOT A,D-B
 70 PLOT C-A,B
 80 PLOT C-A,D-B
 90 RUN
```

```
  1 REM )))PROGRAMA 4(((
  2 REM SOLUCAO DO EXC.2
 10 LET A=2.1
 20 LET B=3.4
 30 LET C=5
 40 LET D=6
 50 FOR E=38 TO 47
 60 PRINT CHR$ E;")  ";
 70 GOSUB E*10
 80 FOR I=1 TO 30
 85 NEXT I
 90 NEXT E
100 STOP
380 PRINT C+B/D+A
385 RETURN
390 PRINT B*C/D-A
```

```
 395 RETURN
 400 PRINT (A+B)/B+C
 405 RETURN
 410 PRINT ABS (A-B)*SQR C
 415 RETURN
 420 PRINT INT (D/C)+PI
 425 RETURN
 430 PRINT B*SGN (C-D)
 435 RETURN
 440 PRINT LN A+B**C
 445 RETURN
 450 PRINT LN B+INT B*(A+C)
 455 RETURN
 460 PRINT SQR ((B**2-A*A*C)/-(2
*A))
 465 RETURN
 470 PRINT (A+B)/(C+D)+A**2-(COS
 B*(A/C))+2*PI
 475 RETURN
```


3 ESTRUTURA DE UM PROGRAMA

3.1 VARIÁVEIS E CONSTANTES

A linguagem BASIC é composta por um conjunto de *linhas numeradas* (registros). Cada linha contém um *comando* (instrução). Os comandos são compostos de *elementos* e outros símbolos.

Os símbolos separadores entre elementos são:

Espaço em branco	→	" "
Parênteses	→	( )
Vírgula	→	,
Ponto e vírgula	→	;
Igualdade	→	=

Um valor que não se altera durante a execução do programa é chamado de *constante,* e é representado pelo próprio valor.

34.587	−47.56
0.00045	60000000

O valor que se altera durante a execução é chamado de *variável,* e é representado por um *identificador* (nome)

X	A$
NUMERO	W(3)

Resumindo, um programa em BASIC é composto por:

- Linhas numeradas em ordem crescente
- Comandos ou instruções
- Símbolos separadores, gráficos e outros
- Constantes
- Variáveis

3.2 COMANDOS

São os elementos ativos da linguagem. Indicam as operações que devem ser executadas, como somar dois números, ler um valor, fornecer certa informação.

Os comandos são executados seqüencialmente, isto é, de cima para baixo, a não ser que um desvio seja informado de maneira conveniente.

Todos os comandos, assim como as declarações, devem estar numa linha numerada de 1 a 9999. A seqüência é livre. Recomenda-se, entretanto, um intervalo de 10 ou 5, para permitir futuras inserções (o que ocorre com bastante freqüência).

Os principais comandos são:

LET	IF ... THEN
INPUT	GOTO
PRINT	FOR ... NEXT

3.3 COMENTÁRIOS

Para facilitar a compreensão de um programa, convém incluir algumas linhas de comentários explicativos. Após algum tempo, esses comentários tornam-se úteis, explicando o que determinado programa, determinado trecho de programa ou determinada variável executa ou representa.

Os comentários são desconsiderados durante o processamento. Uma linha de comentários é indicada pela palavra- chave *REM* à frente do comentário.

REM

Convém, entretanto, não abusar dos comentários, pois consomem memória do computador, que é limitada.

10 REM <<< CÁLCULO DA MEDIA >>>

3.4 FORMATO DOS PROGRAMAS

Os programas em BASIC são escritos linha por linha, em formato livre (sem posições fixas). A capacidade máxima de cada linha é de 32 caracteres (colunas 0 a 31).

Quando um comando precisar mais de 32 posições, as linhas seguintes podem ser utilizadas sem necessidade de uma indicação específica de continuação.

3.5 UM PROGRAMA SIMPLES

O seguinte programa lê dois valores (informados via teclado), calcula a média aritmética e expõe o resultado no vídeo:

```
10 REM PROGRAMA QUE CALCULA A MEDIA
   ARITMETICA ENTRE "A" E "B"
20 INPUT A
30 INPUT B
40 LET MEDIA = (A + B)/2
50 PRINT MEDIA
60 STOP
```

A linha 10 corresponde a uma linha de comentário. As linhas 20 e 30 incluem dois comandos de leitura. A linha 40, um comando de *atribuição aritmética*, e a linha 50 um comando de saída, mandando exibir o valor da variável MEDIA.

O comando de parada da linha 60 é opcional, já que o processamento pára quando não encontra mais nenhuma linha.

Todos estes comandos serão analisados com detalhes nos próximos capítulos.

3.6 COMANDO DE ATRIBUIÇÃO ARITMÉTICA

Este comando é o primeiro a ser explicado, por ser de fundamental importância. Sua forma geral é:

LET < VARIÁVEL > = < EXPRESSÃO ARITMÉTICA > **LET**

O computador atribui à variável, identificada à esquerda, o valor calculado da expressão à direita, e preserva este valor na sua memória com o nome da variável indicada.

O símbolo = indica, na realidade, armazenamento, e não igualdade como tradicionalmente é entendido.

Se algum valor já estava guardado na memória com o mesmo nome de variável, o valor antigo é perdido e substituído pelo novo valor.

O comando de atribuição:

1. Executa a expressão à direita do sinal " = ";
2. Atribui o valor obtido à variável à esquerda do sinal " = ".

Exemplos:

```
MEDIA  = (A + B)/2
     N = 1ØØ
  A(1) = A * B ** 2 + 347.ØØ6
```

Os valores das variáveis indicadas à direita do sinal de igualdade são usados e não modificados.

Assim, se A = 2 e B = 4, após o comando

MEDIA = (A + B)/2

tem-se:

```
MEDIA = 3
    A = 2
    B = 4
```

Outro exemplo:

CONTADOR = CONTADOR + 1
se CONTADOR = 5, a atribuição acima resultará em

CONTADOR = 5 + 1 = 6

Devido à variável CONTADOR encontrar-se também à direita do sinal de igualdade, deve ter sido previamente definida. Esta variável, que também poderia ter outro nome, é muito usada para a contagem do número de vezes que determinado trecho de programa ocorre. Para isto, é importante que a variável seja definida antes da primeira utilização, comumente com o valor zero ou um. Veja o exemplo abaixo:

```
1Ø  CONTADOR = Ø
2Ø  CONTADOR = CONTADOR + 1
 . . . . . .
 . . . . . .
 . . . . . .
15Ø (se contador = 1ØØ então pare o processamento)
16Ø (volte à linha 2Ø)
```

Resumindo:

1. O símbolo " = " indica armazenamento e não igualdade;
2. Os valores das variáveis que se encontram à direita do sinal de igualdade são usados, mas não modificados ou removidos;
3. O valor de uma variável somente é modificado quando a mesma aparece à esquerda do sinal de igualdade;
4. À esquerda do sinal de igualdade somente podem aparecer variáveis, e uma de cada vez.

3.7 COMANDOS SIMPLES DE ENTRADA E SAÍDA

A forma geral do comando de leitura é:

INPUT < VARIAVEL > **INPUT**

Quando, numa determinada linha, o computador encontra o comando *INPUT,*

```
6Ø  . . . . . .
7Ø  INPUT VALOR DE A
8Ø  . . . . . .
```

ele interrompe o processamento e aguarda um valor numérico que deverá ser informado via teclado. Uma vez informado, ele "guarda" esse valor com o nome da variável *VALOR DE A* e continua o processamento. Se for digitado o valor 2, o computador assume

VALOR DE A = 2

Se já existia um valor para **VALOR DE A,** este será substituído pelo novo valor informado.

Somente é possível entrar com um valor por comando.

A forma simplificada de saída é:

PRINT < EXPRESSAO ARITMETICA > **PRINT**

Quando o computador encontra este comando, expõe na tela da televisão o valor ou valores das variáveis ou o resultado da expressão aritmética.

O comando *PRINT* permite a saída de mais de um valor, simultaneamente, o que não acontece com o *INPUT.* Para isto, é suficiente fazer

PRINT < VARIAVEL 1 > , < VARIAVEL 2 > , ...

Um programa simples para o cálculo da média aritmética:

```
1Ø  REM CALCULA MEDIA ARITMETICA
2Ø  REM DE DOIS VALORES, A E B
3Ø  INPUT A
4Ø  INPUT B
5Ø  LET MEDIA = (A + B)/2
6Ø  PRINT A, B, MEDIA
```

Se a variável *MEDIA* não precisa ser preservada para uso posterior, a linha 50 do exemplo acima pode ser abolida, ficando

. . . . . .

```
30  INPUT A
40  INPUT B
60  PRINT A, B, (A + B)/2
```

Assim, é possível fazer:

```
10  LET A = 10
20  LET B = 6
30  PRINT A + B, A−B, A/B, A ** B, LN A, SQR B
```

Além de entrar e sair com valores numéricos, os comandos *INPUT* e *PRINT* também podem ser usados para entrada (armazenamento) e saída de palavras, nomes ou frases. A distinção é feita pela adição, à variável, do símbolo *$* (dólar). Assim, quando o computador encontra

INPUT A

ele sabe que vai receber um valor numérico, enquanto que com

INPUT A$

ele vai receber, para armazenamento, uma palavra ou uma frase. No programa

```
10  PRINT "DIGA QUAL O SEU NOME"
20  INPUT N$
30  PRINT N$
```

exibe na tela o nome de uma pessoa, que foi preservada na memória com nome *N$*. Se for digitado o nome Maria do Carmo Machado, o computador assume

N$ = "MARIA DO CARMO MACHADO"

Cada vez que for comandado *PRINT N$,* este nome será exibido na tela. Uma variável com *$* só pode ter uma letra:

N$, A$, Q$,
U$, X$, W$

Maiores detalhes a respeito serão vistos no Capítulo 5.

3.8 COMANDO *GOTO*

Os comandos são executados seqüencialmente. Para alterar esta seqüência, usa-se o comando *GOTO* (vá para). O fluxo é, assim, desviado para o comando que tiver o número indicado no *GOTO.* A sintaxe é:

onde a expressão pode ser uma expressão numérica, expressão lógica, ou mesmo um número.

Exemplo:

```
1Ø  REM EXEMPLO DO COMANDO < GOTO >
2Ø  REM
3Ø  INPUT A
4Ø  INPUT B
5Ø  PRINT "MEDIA = " ; (A + B)/2
6Ø  GOTO 30
```

Ao chegar à linha 60, o computador encontra a instrução *GOTO* e retorna à linha 30, permitindo a entrada de novos valores. Este exemplo, como foi mostrado, ficará "rodando" indefinidamente, até ser interrompido por alguma ação externa. Posteriormente veremos como contornar este problema.

Se o *GOTO* indicar uma linha inexistente, o programa passará ao número superior mais próximo. Assim, é possível fazer:

6Ø GOTO 25

que ele retornará à linha 30.

Outras possibilidades:

```
 2Ø  GOTO FIM
13Ø  GOTO C + B
 4Ø  GOTO I * 1ØØØ
25Ø  GOTO NUMERO * 1ØØ + 1Ø
```

3.9 EXERCÍCIOS

3. Escreva um programa que leia 10 valores, indique cada valor lido, acumule a soma dos valores lidos e a soma dos quadrados dos valores. No final, o programa deve expor os valores acumulados.
4. Ao programa anterior, inclua o cálculo da média aritmética e da variância dos *N* valores, sendo

   MEDIA = SOMATORIO X / N

   VARIANCIA = (SOMATORIA X^2 – $MEDIA^2$) / (N – 1)
5. Escreva um programa que calcule as despesas normais do mês, como as contas de água, luz, telefone, aluguel ou BNH, condomínio, condução ou combustível, alimentação, educação, atividades sociais,

etc. O programa deve mostrar o valor de cada despesa bem como a despesa total do mês.

6. Ao programa anterior, inclua o cálculo da participação de cada despesa (em porcentagem) em relação à despesa total, onde

$$\text{PARTICIPACAO DA DESPESA X} = \frac{\text{DESPESA X}}{\text{SOMA DAS DESPESAS}} * 100$$

7. Seja $A = \begin{vmatrix} A11 & A12 \\ A21 & A22 \end{vmatrix}$ uma matriz 2 X 2.

O determinante de A (DETA) é obtido pela fórmula:

DETA = A22 X A11 − A12 X A21

e o inverso de A (INVA) por

$$INVA = \begin{vmatrix} A22/DETA & -A12/DETA \\ -A21/DETA & A11/DETA \end{vmatrix}$$

Faça um programa que calcule o valor do determinante e mostre o determinante inverso.

8. Sejam AX + BY = E, e
CX + DY = F
um sistema de duas equações e duas incógnitas. A sua solução através da Regra de Cramer é:

$$X = \frac{\begin{vmatrix} E & B \\ F & D \end{vmatrix}}{\begin{vmatrix} A & B \\ C & D \end{vmatrix}} = \frac{E.D - B.F}{A.D - B.C} \qquad Y = \frac{\begin{vmatrix} A & E \\ C & F \end{vmatrix}}{\begin{vmatrix} A & B \\ C & D \end{vmatrix}} = \frac{A.F - E.C}{A.D - B.C}$$

Escreva um programa que encontre a solução.
Se (A.D − B.C) = 0, o sistema ou não tem solução ou tem mais de uma.

3.10 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
  1 REM >>>PROGRAMA 5<<<
 10 LET A=0
 20 LET A=A+1
 40 IF A=11 THEN STOP
 50 CLS
 60 LET B=0
 70 LET B=B+1
 80 PRINT A;" X ";B;" = ";A*B
```



```
 85 IF B=10 THEN PAUSE 180
 90 IF B=10 THEN GOTO 20
100 GOTO 70


  1 REM >>>PROGRAMA 6<<<
 10 PRINT "CALCULO DE JUROS SIM
PLES"
 15 PRINT "--------------------
---"
 20 PRINT
 30 PRINT "QUAL A QUANTIA EMPRE
STADA ?"
 40 INPUT QUANTIA
 50 PRINT "CR$ ";QUANTIA
 55 PRINT
 60 PRINT "POR QUANTOS MESES ?"

 70 INPUT MES
 80 PRINT MES;" MESES"
 85 PRINT
 90 PRINT "A QUE TAXA MENSAL ?"

 95 PRINT "(JUROS, EM PORCENTAG
EM)"
100 INPUT TAXA
105 PRINT TAXA;" PORCENTO"
110 LET JUROS=QUANTIA*TAXA/100*
MES
115 PRINT
120 PRINT "JUROS = CR$ ";JUROS
130 LET MONTANTE=QUANTIA+JUROS
135 PRINT
140 PRINT "MONTANTE = CR$ ";MON
TANTE
145 PRINT
150 PRINT "PAGAMENTO MENSAL = C
R$ ";MONTANTE/MES
```

COMANDOS CONDICIONAIS E ITERATIVOS

A possibilidade de tomar decisões permite ao computador resolver problemas complexos. Esta possibilidade, associada à capacidade de memória e rapidez de cálculo, faz dele um instrumento insubstituível nos tempos atuais.

4.1 EXPRESSÕES LÓGICAS

Além das expressões aritméticas e alfanuméricas, a linguagem BASIC também trabalha com *expressões lógicas,* também conhecidas por *expressões booleanas.*

Enquanto o resultado de uma expressão aritmética é um valor numérico, o de uma expressão lógica assume somente dois *valores: TRUE* (verdadeiro) ou *FALSE* (falso). (Ou seja, *certo* ou *errado*).

TRUE
FALSE

Sua sintaxe é:

< EXP. ARITMÉTICA > < OPERADOR DE RELAÇÃO > < EXP. ARITMÉTICA >

Os operadores de relação são:

=	→	IGUAL
<	→	MENOR
>	→	MAIOR
<=	→	MENOR OU IGUAL
>=	→	MAIOR OU IGUAL
<>	→	DIFERENTE

Exemplos:

Se I = 5, então o resultado das seguintes expressões lógicas será:

I > 5	→	FALSE	(errado)
I = 5	→	TRUE	(certo)
I <= 5	→	TRUE	(certo)
2 * I = 5 + 3	→	FALSE	(errado)
I + 5 <= 25 – I	→	TRUE	(certo)
I <> 5	→	FALSE	(errado)

4.2 OPERADORES LÓGICOS

O conceito de expressão lógica fica ampliado quando se associa a este conceito o de *operadores lógicos.*

Ambos podem ser combinados, formando

< EXP. LOGICA > < OPERADOR LOGICO > < EXP. LOGICA >

Os operadores lógicos são três:

AND	(E)	AND
OR	(OU)	OR
NOT	(NÃO)	NOT

Esses operadores são definidos por uma tabela conhecida por *tabela de verdades:*

1. EXP. LÓGICA = A	2. EXP. LÓGICA = B	NOT B	A AND B	A OR B
TRUE	TRUE	FALSE	TRUE	TRUE
TRUE	FALSE	TRUE	FALSE	TRUE
FALSE	TRUE	FALSE	FALSE	TRUE
FALSE	FALSE	TRUE	FALSE	FALSE

Os operadores *AND* e *OR* associam duas expressões lógicas. O operador *NOT* atua somente sobre a expressão a sua direita.

Exemplos:

Continuando com I = 5,

1. EXP. LÓGICA	OPERADOR LÓGICO	2. EXP. LÓGICA	RESULTADO
I > 5	AND	I = 5	FALSE
I < 10	AND	I > 3	TRUE
I = 5	OR	I > 8	TRUE
2 * I < 20	OR	I + 1 > 5	TRUE

Na combinação de expressões aritméticas, expressões lógicas e operadores lógicos, a ordem de execução segue a seguinte prioridade:

1. Cálculo das expressões aritméticas
2. Expressões lógicas
3. NOT
4. AND
5. OR

Para alterar esta ordem de prioridade, deve-se lançar mão dos parênteses. As operações com mesma prioridade continuam sendo executadas da esquerda para a direita.

Exemplos:

a) Se A = 5, B = 3, C = 1 e D = 4,

((A * B) > C + D OR D < (C – A)) AND A > B

Resulta em

```
( 15   > 5  + D  OR  4 <   –4 ) AND 5   > 3
(     TRUE        OR    FALSE    AND  TRUE
                 TRUE             AND  TRUE
                          TRUE
```

b) Se A = 8 e B = 1, então

```
B = B + (A = 8)          → 2
A = A + (2 AND B = 8)    → 1Ø
A = A + (B = 8)          → 8
```

Se a expressão lógica for *TRUE* (A = 8), então o programa assume o valor um (1), caso contrário (B = 8), assume o valor zero (Ø).

c) Se *T < 100, então incremente T em 1 unidade,* pode ser representado por

LET T = T + (T < 1ØØ)

se T = 50,

```
T = 5Ø + (5Ø < 1ØØ)
T = 5Ø +   TRUE
T = 5Ø +      1
T = 51
```

se T = 100,

```
T = 1ØØ + (1ØØ < 1ØØ)
T = 1ØØ +   FALSE
T = 1ØØ +     Ø
T = 1ØØ
```

d) Se *A = 8, então incremente X numa unidade, e se A = 5 decremente numa unidade.* Pode ser representado por:

LET X = X + (A = 8) – (A = 5)

se A = 8 e X = 8,

```
X = 8 + TRUE – FALSE
X = 8 +  1   –   Ø
X = 9
```

se A = 5 e X = 8,

```
X = 8 + FALSE – TRUE
X = 8 +   Ø   –   1
X = 7
```

se A = 9 e X = 8,

```
X = 8 + FALSE – FALSE
X = 8 +   Ø   –   Ø
X = 8
```

e) *Se A = 6, incremente Y em 2 unidades, se A = 7, decremente numa unidade.* Pode ser representado por:

LET Y = Y + (A = 6) * 2 – (A = 7), **OU**
LET Y = Y + (A = 6) + (A = 6) – (A = 7), **OU**
LET Y = Y + (2 AND A = 6) – (1 AND A = 7)

4.3 COMANDO *IF*

IF
THEN

O uso do comando *IF* (se) permite alterar o fluxo de um programa, com base em certas condições. Sua sintaxe é:

IF < EXP. LÓGICA > THEN < COMANDO >

e representa,

SE < ISTO > ENTÃO < FAÇA AQUILO >

O comando *IF* só será executado se, e somente se, a expressão lógica for verdadeira.

Exemplos:

```
IF A = 8  THEN LET Y = Y + 1
IF X = Ø  THEN STOP
IF A ** 2 + B ** 2 = 1 THEN GOTO 52Ø
IF VALOR = 999 THEN PRINT SOMA
```

A associação do comando *IF* com operadores lógicos aumenta sua versatilidade.

Exemplos:

```
IF I > 2 AND I < 6 THEN LET A = A + 1
IF I = 999 OR J = 999 THEN STOP
IF C = H AND Y = P OR Y = Q THEN GOTO 2Ø5
IF X = 2 OR X = 4 OR X = 6 THEN PRINT "PAR"
IF C + A > B AND (H = 5 OR H = 8) THEN GOTO 3Ø48
```

O comando *IF* pode vir aninhado com outro:

```
IF A < 1Ø THEN IF A > 5 THEN B = 1
```

é análogo a

```
IF A < 1Ø AND A > 5 THEN B = 1
```

O seguinte conjunto de *IF'S,*

```
 5Ø  .
 6Ø  INPUT I
 7Ø  IF I = 1 THEN GOTO 21Ø
 8Ø  IF I = 2 THEN GOTO 32Ø
 9Ø  IF I = 3 THEN GOTO 38Ø
1ØØ  IF I = 4 THEN GOTO 43Ø
11Ø  . . . . . .
```

Pode ser substituído por um *GOTO* (simulando um *GOTO* controlado):

```
 5Ø  .
 6Ø  INPUT I
 7Ø  GOTO I * 1ØØ
 8Ø  . . . . . .
1ØØ  . . . . . .
     . . .  . . . . . .
2ØØ  . . . . . .
     . . . . . .
3ØØ  . . . . . .
     . . . . . .
4ØØ  . . . . . .
     . . . . . .
```

Conforme o valor de I (1, 2, 3 ou 4), o comando da linha 7Ø desvia o processamento para as linhas 1ØØ (I = 1), 2ØØ (I = 2), 3ØØ (I = 3) e 4ØØ (I = 4).

Outras particularidades:

IF X >= Ø é oposto a IF X < Ø
IF X < C é oposto a IF X >= C
IF X = Ø é oposto a IF X <> Ø
IF X = A é oposto a IF NOT X = A
IF NOT X = Y AND C > Ø é igual a IF NOT X = Y OR NOT C > Ø
IF NOT X = Y OR C > Ø é igual a IF NOT X = Y AND NOT C > Ø
IF NOT A é igual a IF A = Ø
IF A é igual a IF A <> Ø

Na realidade, os operadores, = , < , > , <= , >= , e <> são operadores binários, onde o resultado é 1 se for verdadeiro, e 0 se for falso. Quando o resultado de uma relação é zero (falso), a instrução não é executada.

4.4 COMANDO *FOR*

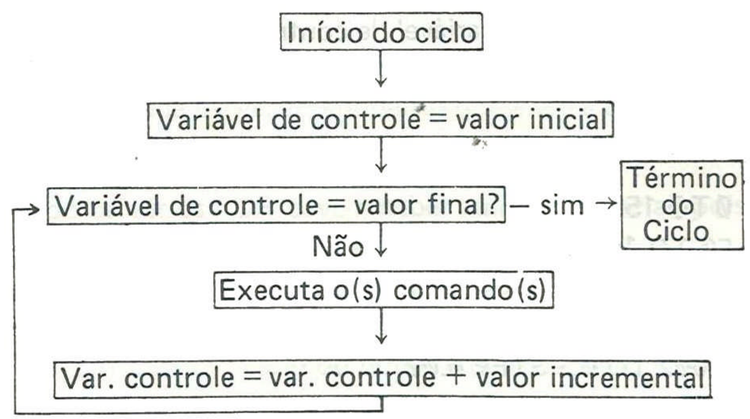
Para repetir determinado trecho de programa várias vezes, sem repetir os comandos deste, pode-se lançar mão do comando *FOR* (para). **FOR**

Este comando permite o controle de um processo iterativo. Todos os comandos a serem repetidos ficam sob controle de uma *variável de controle,* cujo valor modifica-se a cada iteração (repetição), até um determinado limite ou condição especificada.

A forma geral do comando é:

FOR < Variável de Controle > = < Valor Inicial > TO < Valor Final > STEP < Valor Incremental STEP

O fluxograma auxilia na compreensão do comando *FOR*



Quando o valor incremental é igual a 1, é desnecessário acrescentar o valor incremental; o programa assume *STEP 1*.

O trecho do programa que deverá se repetir deve estar compreendido entre o comando inicial *FOR* e o final *NEXT* (seguinte). A sua sintaxe é:

NEXT < **variável de controle** > **NEXT**

Os comandos *FOR* e *NEXT* devem estar sempre associados.

Exemplos:

a) O cálculo da média de 10 pares de valores pode ser obtido através de

```
1Ø  FOR C = 1 TO 1Ø
2Ø        INPUT  X1
3Ø        INPUT  X2
4Ø        PRINT "MEDIA = "; (X1 + X2)/2
5Ø  NEXT   C
```

Na linha 10, o computador estabelece C = 1 e, ao chegar à linha 50, testa a variável *C* para verificar se já alcançou o valor 10 ou não. Se for menor que 10, incrementa a variável *C* em uma unidade e retorna à linha 10. Quando a variável alcança o valor limite 10, o controle não retorna e prossegue à linha seguinte.

b) A exibição dos números pares de 0 a 100 pode ser conseguida através de:

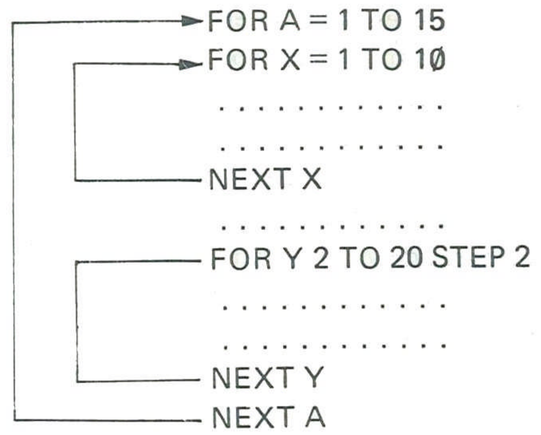
```
1Ø  REM   NUMEROS PARES ATE 1ØØ
2Ø  FOR N = 2 TO 1ØØ STEP 2
3Ø        PRINT N
4Ø  NEXT   N
```

Os valores inicial e final da variável de controle podem variar de — 9999999 até 9999999, incluído o zero.

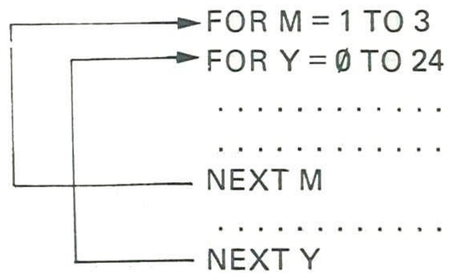
Construções válidas:

```
FOR A  =  Ø TO 15
FOR L  =  5Ø TO 1Ø STEP –1
FOR X  =  A TO B STEP C
FOR L  =  2 * X TO B ** 2
FOR A  =  Ø.ØØ2 TO Ø.1 STEP Ø.Ø2
FOR V  =  INT (RND * 1Ø) TO Ø STEP –1
```


Vários comandos *FOR* podem vir aninhados:



Não é permitido um *FOR* interno saltar para fora de outro:



c) No seguinte programa

```
10  FOR I = 1 TO 5Ø
20     PRINT "M";
30     FOR J = 1 TO 1Ø
40           PRINT "U";
50     NEXT J
60  NEXT I
```

A letra *M* será mostrada 50 vezes, enquanto que a letra *U* 50 × 10 = 500 vezes.

O uso aninhado de vários *FOR* deve receber cuidado especial, pois seus efeitos são multiplicativos. Um engano pode fazer o computador não parar de processar, ou ficar processando várias horas. É aconselhável, sempre que for necessário, o uso de vários *FOR* aninhados, o cálculo prévio do número de iterações, para evitar surpresas desagradáveis.

4.5 SIMULAÇÃO DO COMANDO *WHILE*

O comando *WHILE* (enquanto) executa determinado trecho de programa repetidamente enquanto o resultado de uma expressão não assumir determinado valor. Este comando não existe no BASIC abordado, mas pode ser simulado facilmente. Para exemplificar, consideremos o seguinte problema: calcular a média aritmética de diversos pares de valores (desconhecido o número de pares). O programa deve ler novos pares de valores "enquanto" a média calculada não for nula. Quando a última média calculada for zero, o processo deve ser interrompido. O programa BASIC para este caso será:

```
10  LET MEDIA = 0
20  FOR I = 0 TO 1
30      INPUT A
40      INPUT B
50      PRINT "MEDIA = "; (A + B)/2
60      LET I = (MEDIA = (A + B)/2))
70  NEXT I
```

Como funciona? É fácil de entender, se olharmos com atenção a linha 60. Inicialmente, na linha 20, a variável I assume o valor zero. Chegando à linha 60, e supondo *A* e *B* diferentes de zero, a expressão *MEDIA* = = (*A* + *B*)/2 será *FALSE*, e sendo false, I = 0. Ou seja, a linha 60 fará com que a variável I seja sempre zero, enquanto *A* e *B* forem diferentes de zero. E *I* sendo zero, o *FOR* da linha 20 é reinicializado a cada passada. Quando *A* e *B* forem nulos (A = B = 0), a expressão da linha 60 passará a *TRUE* e, portanto, igual a um, finalizando o *FOR* da linha 20.

4.6 EXERCÍCIOS

9. Escreva um programa que selecione aleatoriamente um número entre 1 e 100, e peça para que esse número seja advinhado. Se o "chute" for muito alto ou muito baixo, o programa deve dizê-lo. Nos acertos, o programa deve "dar os parabéns" e escolher aleatoriamente outro número. Para fazer a escolha aleatória do número, use a sentença

   *LET NUMERO = INT (RND * 100) + 1*

   onde NUMERO resulta sempre num valor inteiro entre 1 e 100.

10. Ao programa anterior inclua um contador de tentativas. Se o acerto for "de cara", o programa deve conferir "menção honrosa". Se o acerto ocorrer em até 5 tentativas, conceito "bom" e, em mais de cinco, conceito "fraco". A cada acerto, o programa deve indicar o número de tentativas efetuadas.

11. Escreva um programa "tutor de matemática". Deve-se escolher aleatoriamente dois valores entre 1 e 100. Chamando de "A" e "B" estes dois valores, o programa deve perguntar quanto é $A \times B$ e $A + B$ e pedir os resultados. O programa deve permitir duas tentativas para o acerto e, caso o acerto não ocorra, fornecer a resposta correta.

12. Uma empresa vende peças de automóvel. Faça um programa que emita de maneira simplificada uma ou mais notas fiscais (NF), mas uma de cada vez. O programa deve perguntar: a) quantos itens (peças diferentes) vão constar na NF; b) nome do item (peça), quantidade vendida e preço unitário; c) data. Deve calcular: a) o custo total da NF; b) o desconto: zero porcento (PC) para compras até Cr$ 200.000,00; 5 PC para compras entre 200.001 e Cr$ 500.000,00; 10 PC para compras superiores a Cr$ 500.000,00. Deve exibir: a) nome da empresa (invente um); b) data; c) relação dos itens, quantidades, preços unitários e parciais; d) preço total sem e com desconto.

13. As raízes X1 e X2 da equação de segundo grau

    $AX^2 + BX + C = 0$

    são dadas por:

    $X1 = -B/2A + SQR\ D$
    $X2 = -B/2A - SQR\ D$
    $D = (B^2 - 4AC) / 4A^2$

    se D for positivo ou zero, as raízes são reais. Consegue-se maior precisão fazendo

    $X1 = -B/2A + SQR\ D$ se $-B/2A >= 0$
    $X1 = -B/2A - SQR\ D$ se $-B/2A < 0$

    e para qualquer caso

    $X2 = C/X1.A$

    se $D < 0$, as raízes são complexas. Elabore um programa que calcule as raízes, avisando quando as mesmas são complexas.

4.7 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
  1 REM >>>PROGRAMA 7<<<
 10 PRINT "A QUINA DA LOTO DEST
A SEMANA"
 20 PRINT "VAI DAR:"
 30 PRINT
 40 PRINT
 50 FOR I=1 TO 5
 60 PRINT INT (RND*100);"   ";
 70 NEXT I
```

```
  1 REM >>>PROGRAMA 8<<<
 10 PRINT "     CARTAO DA LOTEC
A"
 20 PRINT "     ---------------
-"
 30 PRINT
 40 LET B$="
          "
 50 PRINT "    COL.1  COL.2  CO
L.3"
 60 PRINT "    -----------------
---"
 70 PRINT AT 20,0;"INDIQUE AS C
HANCES"
 80 PAUSE 60
 90 PRINT AT 20,0;B$
100 PRINT AT 20,0;"EM PORCENTAG
EM, DE CADA UMA"
110 PAUSE 60
120 PRINT AT 20,0;B$
130 PRINT AT 20,0;"DAS 3 COLUNA
S, OK ?"
140 PAUSE 60
150 PRINT AT 20,0;B$
160 FOR I=1 TO 13
165 PRINT AT 20,0;"JOGO ";I
166 PAUSE 60
167 PRINT AT 20,0;B$
170 PRINT AT 20,0;"COLUNA 1 ?"
180 INPUT C1
190 PRINT AT 20,7;"2"
200 INPUT C2
210 PRINT AT 20,7;"3"
```



```
220 INPUT C3
230 PRINT AT 20,0;B$
231 LET E1=0
232 LET E2=0
233 LET E3=0
235 FOR J=1 TO 3
240 LET N=RND
250 LET E1=E1+(N<=C1/100)
260 LET E2=E2+(N<=C2/100)
270 LET E3=E3+(N<=C3/100)
280 NEXT J
290 IF E1=E2 AND E1=E3 THEN PRI
NT AT I+5,0;I;TAB 7;"X";TAB 14;"
X";TAB 21;"X"
300 IF E1>E2 AND E1>E3 THEN PRI
NT AT I+5,0;I;TAB 7;"X"
310 IF E2>E1 AND E2>E3 THEN PRI
NT AT I+5,0;I;TAB 14;"X"
320 IF E3>E1 AND E3>E2 THEN PRI
NT AT I+5,0;I;TAB 21;"X"
330 IF E1=E2 AND E1>E3 THEN PRI
NT AT I+5,0;I;TAB 7;"X";TAB 14;"
X"
340 IF E3=E2 AND E2>E1 THEN PRI
NT AT I+5,0;I;TAB 14;"X";TAB 21;
"X"
350 IF E3=E1 AND E1>E2 THEN PRI
NT AT I+5,0;I;TAB 7;"X";TAB 21;"
X"
360 NEXT I
370 PRINT AT 21,0;"BOA SORTE...
"
```

```
  1 REM >>>PROGRAMA 9<<<
 10 REM "NOITE ESTRELAR"
 20 FOR I=1 TO 10
 30 FOR J=1 TO 32
 40 PRINT CHR$ 128;
 50 NEXT J
 60 NEXT I
 70 PRINT AT RND*9,RND*31;CHR$
151
 80 FOR K=1 TO 3
 90 PRINT AT RND*9,RND*31;CHR$
128
100 NEXT K
110 PRINT AT 1,1;CHR$ 128
120 PRINT AT 1,1;CHR$ 151
130 GOTO 70
```

```
  1 REM >>>PROGRAMA 10<<<
 10 GOTO 200
 50 PRINT AT 10,8;H;":";
 60 IF M<10 THEN PRINT "0";
 70 PRINT M;":";
 80 IF S<10 THEN PRINT "0";
 90 PRINT S
100 LET S=S+1
110 LET M=M+(S=60)
120 IF S=60 THEN LET S=0
130 LET H=H+(M=60)
140 IF M=60 THEN LET M=0
150 IF H=13 THEN LET H=1
190 GOTO 50
200 PRINT ,,,,
220 PRINT TAB 8;"RELOGIO DIGITA
L",,,
240 PRINT "QUE HORAS SAO ?",,
260 PRINT "HORAS ? ";
270 INPUT H
280 PRINT H
290 PRINT "MINUTOS ? ";
300 INPUT M
310 PRINT M
320 PRINT "E SEGUNDOS ? ";
330 INPUT S
340 PRINT S
345 CLS
350 GOTO 50
```


5 STRING

5.1 DECLARAÇÃO STRING

Para facilitar o tratamento de um conjunto (cadeia) de caracteres, a BASIC dispõe de uma variável específica, chamada *variável string.* Esta variável permite o armazenamento e a manipulação de cadeias de caracteres.

Um *string* é considerado como um dado cujo componente é um conjunto de caracteres que podem ser alfabéticos, numéricos, gráficos ou outros.

Uma variável string tem dois atributos: comprimento e conteúdo. O comprimento corresponde ao número de caracteres. Sua forma geral é:

< uma letra > $ = " < conj. caracteres > "

O computador diferencia a variável STRING de uma variável comum pela presença do símbolo *$* (dólar). A definição de uma variável STRING está restringida pelo uso de somente uma letra; pode-se, assim, usar no máximo 26 variáveis STRING por programa.

É possível o uso simultâneo de uma variável simples e STRING de mesmo nome: *A* e *A$.*

Exemplos:

```
PRINT "MEDIA ARITMETICA = "; (A + B)/2
PRINT "CUSTO FINAL: " ; C; "CRUZEIROS"
LET N$ = "ALFREDO GUIMARAES"
LET D$ = "19/12/83"
IF Z = 6 THEN PRINT "OS INVASORES VENCERAM"
PRINT "P/CONTINUAR . . . . DIGITE < NEW LINE > "
INPUT H$
PRINT T$ ; T
```

É possível a soma de variáveis STRING, recebendo, no caso, o nome de *concatenação* ou *junção.* Se

```
A$ = "João", e          B$ = "Silva"
LET C$ = A$ + B$
```

Resulta em

C$ = "Joao Silva"

Não é possível a subtração, multiplicação, divisão ou exponenciação de variáveis STRING; somente a adição.

Associada à variável STRING, a BASIC dispõe de três funções muito úteis:

LEN, VAL, STR$

LEN

A função *LEN* aplicada a uma variável STRING resulta num valor que representa o comprimento da variável, ou seja, o número de caracteres.

Exemplos:
se A$ = "INVASORES DO ESPAÇO": LEN A$ = 19
LEN "MEDIA"
FOR I = 1 TO LEN A$
LET J$ = J$ + ",00"

VAL

A função *VAL* permite que uma variável STRING formada por números e/ou expressões algébricas seja transformada e tratada como número.

Exemplo:

se A = 9, e B$ = 1Ø, então

VAL "32" → 32
VAL "1.5 + SQR A" → 4.5
VAL "5" + VAL "2Ø" → 25
VAL B$ → 1Ø

Se a função *VAL* fizer parte de uma expressão maior, ela deve vir sempre no início da expressão. Assim,

LET X = 1Ø + VAL "K"

deve ser alterado para

LET VAL "K" + 1Ø

Nas especificações de coordenadas (associada a PRINT, PLOT, UNPLOT), a função *VAL* somente pode ser utilizada para especificar a primeira coordenada, como nos exemplos:

PRINT AT VAL "X", Y
PLOT VAL "Y", H

STR$

A função *STR$* tem função contrária da função VAL. Se aplicada sobre um número, o transforma num STRING. Assim,

STR$ 3.5 → "3.5"

Todas estas funções podem ser combinadas entre si, como no exemplo seguinte:

LET M = VAL (STR$ LEN "345" + "– 2")

resultando em

M = VAL (STR$ 3 + "– 2")
M = VAL ("3" + "– 2")
M = VAL ("3–2")
M = 3–2
M = 1

É possível o uso de aspas " numa variável STRING, desde que se use a aspa dupla " " **(não são duas aspas simples juntas).** Assim, é possível,

PRINT "TITULO: ""ANALISE FINANCEIRA"" "

resultando em

título: "ANALISE FINANCEIRA"

5.2 SUBSTRING

Entende-se como *substring* parte dos caracteres que formam uma STRING. Assim, se *ABCDEFGH* forma uma variável STRING, *ABC, DE* e *G* são exemplos de substring. A forma geral é:

" < **EXPRESSAO** > " (< **INICIO** > **TO** < **FIM** >)
STRING

Exemplos:
"PEDRO AMERICO" (2 TO 5) = "EDRO"
"3.468,78" (3 TO 4) = "46"

Algumas particularidades:

a) Na omissão do número que indica o início da substring, o computador assume *INÍCIO = 1;*

"ABCDE" (TO 3) = "ABC"

b) Na omissão do número que indica o final da substring, o computador assume como sendo até o último caractere;

"ABCDE" (3 TO) = "CDE"

c) Uma substring composta de somente um caracter pode ser indicada de duas maneiras:

"ABCDE" (3 TO 3) = "ABCDE" (3) = "C"

d) A indicação de todos os caracteres pode ser feita de três maneiras diferentes:

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

e) A indicação de um número final maior que o número de caracteres provoca um erro tipo-3;

"ABCDE" (2 TO 6) → ERRO

f) A indicação de um número inicial maior que o final resulta numa STRING vazia;

"ABCDE" (3 TO 2) = ""

Outras possibilidades no uso de substring:

g) se A$ = "MUITO BEM",
PRINT A$(TO 5) → MUITO

h) se B$ = A$(7 TO),
PRINT B$ → BEM

i) se C$(2 TO 3) = "ABCDE",
PRINT C$ → BC

j) se A$(6) = " – ",
PRINT A$ → MUITO – BEM

k) "ABC" + "OHR"(2 TO 3) = "ABCHR"

l) ("XYZ" + "MMM")(3 TO 4) = "ZM"

m) U$(2 * A TO C + 5)

n)
```
10 LET A$ = "VASCO DA GAMA"
20 FOR I = 1 TO LEN A$
30 PRINT A$(I);
40 NEXT I
```

5.3 EXERCÍCIOS

14) Faça um programa que pergunte o nome e telefone de 5 pessoas. Após a entrada dos dados, deve listar os nomes e os números dos telefones.

15) Altere o programa anterior para que pergunte, de saída, o número de pessoas e depois execute conforme o exercício anterior.

16) Altere o programa **15** para que: a) após a listagem, limpe a tela; b) peça o nome de uma pessoa e indique o número do seu telefone.

17) Faça um programa que guarde o nome e endereço dos amigos, parentes ou outras pessoas, aos quais deseja mandar cartão de Natal no final do ano.

18) Faça um programa que guarde o nome de pessoas e as datas de seu aniversário. Dado um intervalo de dias, ou um mês, o programa deve listar as datas e os nomes das pessoas que aniversariam.

5.4 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
 1 REM >>>PROGRAMA 11<<<
10 LET A$=" -"
20 FOR I=0 TO 30
30 PRINT AT 10,I;A$
40 NEXT I
```

```
 1 REM >>>PROGRAMA 12<<<
10 LET A$="<-"
20 FOR J=30 TO 0 STEP -1
30 PRINT AT 10,J;A$
40 NEXT J
```

```
 1 REM >>>PROGRAMA 13<<<
 10 LET B$="
"
 20 LET A$="NAO ENTENDI O CODIG
O"
 30 FOR I=1 TO 5
 40 FOR J=1 TO 5
 50 PRINT AT 21,0;A$
 60 NEXT J
 70 PRINT AT 21,0;B$
 80 NEXT I
```

```
 1 REM >>>PROGRAMA 14<<<
10 LET M$="+"
20 FOR J=0 TO 31
30 PRINT AT 0,J;M$;AT 21,J;M$
40 NEXT J
50 FOR I=1 TO 20
60 PRINT AT I,0;M$;AT I,31;M$
70 NEXT I
```

```
  1 REM >>>PROGRAMA 15<<<
 10 REM DIGITE UM OU MAIS CARAC
TERES GRAFICOS
 20 INPUT X$
 30 PRINT X$;
 40 GOTO 30
 50 REM APOS A TELA CHEIA, VOLT
E A DAR "RUN"
```

```
  1 REM >>>PROGRAMA 16<<<
 10 REM MAQUINA DE ESCREVER
 20 REM NAO
 30 REM NAO DIGITE BRANCO
 40 IF INKEY$<>"" THEN GOTO 40
 50 IF INKEY$="" THEN GOTO 50
 60 PRINT INKEY$;
 70 GOTO 40
```

```
  1 REM >>>PROGRAMA 17<<<
 10 REM ARTISTA
 20 REM USE AS TECLAS 5,o,7 E 8

 30 LET N$=CHR$ 136
 40 LET X=10
 50 LET Y=5
 60 PRINT AT X,Y;N$
 70 LET X=X+(INKEY$="6" AND X<2
1)-(INKEY$="7" AND X>0)
 80 LET Y=Y+(INKEY$="8" AND Y<3
1)-(INKEY$="5" AND Y>0)
 90 FOR L=1 TO 10
100 NEXT L
110 GOTO 60
```

```
  1 REM >>>PROGRAMA 18<<<
 10 REM COMPRAS EFETUADAS NO SU
PERMERCADO
 15 LET L=0
 16 DIM Z$(14)
 20 PRINT AT 21,0;"NOME DO ARTI
GO COMPRADO ?"
 30 INPUT F$
 40 PRINT AT 21,0;"QUANTO CUSTO
U ?            "
 50 INPUT Z1
 60 GOSUB 9000
 70 PRINT AT 21,0;"OUTRAS COMPR
AS ? <S/N>    "
```



```
 80 INPUT O$
 90 IF O$="S" THEN GOTO 20
100 IF O$="N" THEN STOP
110 GOTO 80
9000 LET L=L+1
9004 LET Z$="000.000.000,00"
9005 LET Z2=15
9006 LET Z1=INT (Z1*100+.5)
9007 LET Z3=INT (Z1/10)
9008 LET Z4=Z1-Z3*10
9009 LET Z2=Z2-1-(Z$(Z2-1)=".")-
(Z$(Z2-1)=",")
9010 LET Z$(Z2)=CHR$ (Z4+28)
9011 LET Z1=Z3
9012 IF Z1>0 THEN GOTO 9007
9013 IF Z2>12 THEN LET Z$(11)=CH
R$ 28
9014 IF Z2>12 THEN LET Z2=11
9016 PRINT AT L,0;F$;TAB Z2+10;"
CR$ ";Z$(Z2 TO 14)
9017 RETURN
```

6 FORMATAÇÃO E USO DA TELA

6.1 DIVISÃO DA TELA

Para exibição dos resultados, o computador utiliza uma área no vídeo da televisão, chamada de *tela.*

Essa tela compõe-se de *22 linhas* (numeradas de 0 a 21, de cima para baixo) e de *32 colunas* (numeradas de 0 a 31, da esquerda para a direita).

Além dessas **22** linhas disponíveis para a exibição de dados, o computador reserva para si mais duas: a 22 e a 23. A linha 23, chamada de *linha de comunicação,* é reservada para a edição de comandos, instruções e o reporte de mensagens do computador com o usuário. A linha 22 é uma linha em branco e serve tão-somente como separador visual entre as duas áreas citadas.

Se o número de caracteres que compõem uma linha excede as 32 posições disponíveis, novas linhas são utilizadas, enquanto houver necessidade. O mesmo já não acontece com a tela. Por segurança, quando as 22 linhas estiverem completas, a exibição de dados é interrompida, e informada através do código de reportagem número 5 na linha 23. O uso da instrução *CONT* limpa a tela, gerando uma nova.

CONT

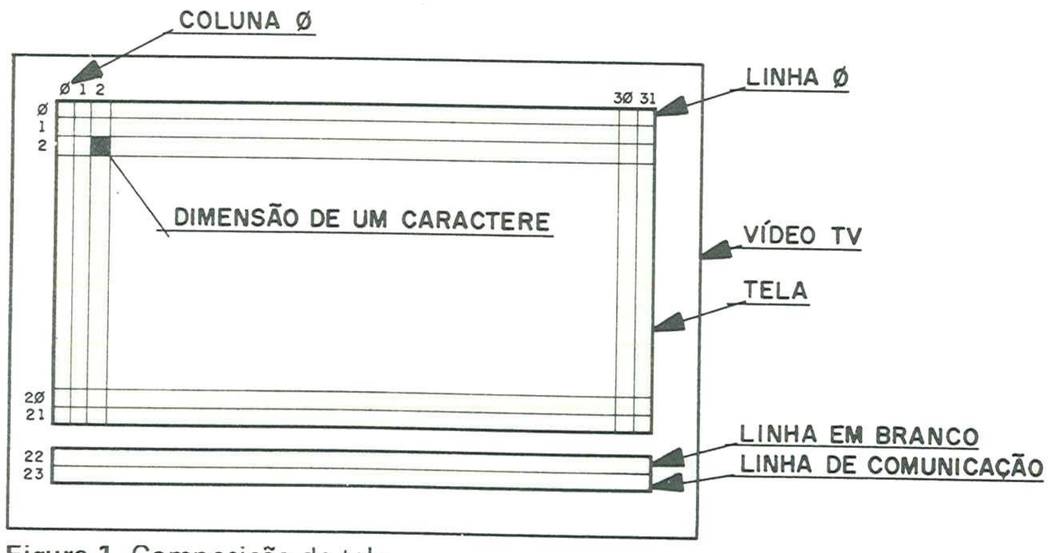


Figura 1. Composição de tela

Além dessas duas áreas, o computador também divide a tela em duas zonas verticais de mesmas dimensões:

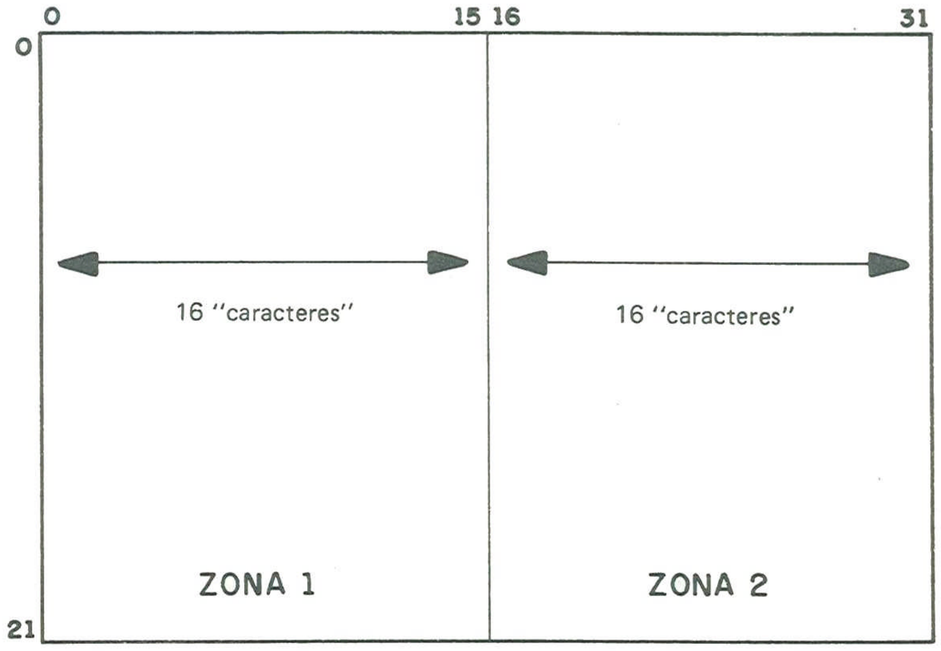


Figura 2. Delimitação das zonas da tela

Como usar determinada linha, coluna ou zona, veremos logo a seguir.

6.2 FORMATO EXPLÍCITO: AT

A forma geral deste comando é:

PRINT AT < NUM. LINHA > , < NUM. COLUNA > ;

PRINT AT

Enquanto *PRINT* indica ao computador que algo deve ser mostrado, *AT < linha > , < coluna >* indica em que posição da tela (número da linha e da coluna). Por exemplo.

PRINT AT 11, 16; "* "

resulta na exibição de um asterisco no meio da tela, na interseção da linha 11 com a coluna 16.

A utilização pura e simples do comando *PRINT* resulta na exibição de algo na linha zero, se a tela estiver limpa, ou na linha imediatamente abaixo da última exibida. O uso de *PRINT AT* altera esta seqüência,

permitindo a exibição de uma informação na linha 21 e, logo após, na linha 3, por exemplo.

É possível o uso de mais de um *AT* por comando.

```
1Ø PRINT  AT 3,1Ø;  "RELATORIO FINAL";
          AT 4,1Ø;  "– – – – – – – – – –";
          AT 6,11;  "DATA: JAN/84"
```

desde que estejam sempre separados por ponto-e-vírgula.
Outros exemplos de uso do *PRINT AT*:

```
1ØØ  PRINT AT C,D; " + "
 8Ø  IF C > Ø THEN PRINT AT 21,Ø; "O INVASOR
     ATERROU"
212  PRINT AT H-2, P; "* "; AT H,P-2 "***";
     AT H-1,P-1; "***"
31Ø  PRINT AT 2,3*V; "■"
94Ø  IF FLAG = 9 THEN PRINT AT 21,Ø;
     "BUM BUM BUUUMMM"
19Ø  PRINT AT ABS(22-A),A; "H"
112  PRINT AT INT (RND*2Ø), INT(RND*31); "*"
 4Ø  PRINT AT 21,Ø; "DIGITE < NEW LINE > P/CONTINUAR"
```

O seguinte exemplo constrói uma moldura na tela:

```
10 FOR L=0 TO 21
20 PRINT AT L,0;CHR$ 136
30 PRINT AT L,31;CHR$ 136
40 NEXT L
50 FOR C=1 TO 30
60 PRINT AT 0,C;CHR$ 136
70 PRINT AT 21,C;CHR$ 136
80 NEXT C
```

O programa número 13 da seção desafio mostra o uso do *PRINT AT* para emissão de mensagem que pisca na tela, para chamar atenção.

O comando *PRINT AT* pode ser utilizado para posicionar o "PRINT" até sobre uma posição já ocupada por um caractere. O novo irá substituir o antigo.

6.3 FORMATO EXPLÍCITO: TAB

Sua forma geral é:
PRINT TAB < NUM. COLUNA > ;

PRINT TAB

O comando *PRINT TAB* move a posição do PRINT para a coluna indicada, permanecendo na mesma linha.

```
PRINT  TAB 1;   "NUMERO"; TAB 12; "CONTEUDO";
       TAB 24;  "PAGINA"
```

resulta na exibição das palavras NUMERO, CONTEUDO e PAGINA, a partir das colunas 1, 12 e 24, respectivamente.

O *PRINT TAB* é muito útil para a tabulação de dados na tela. E possível a indicação de um número superior a 31 (que é o maior valor de coluna). A indicação de *tab 33,* por exemplo, é idêntica a *TAB 1* para a próxima linha.

As especificações *AT* e *TAB* podem vir numa mesma declaração:

```
PRINT AT 1Ø,5; "4."; TAB 11; "REGISTROS ESPECIAIS"
```

resulta na exibição de *4.* nas colunas 5 e 6 da linha 10 e a frase *REGISTROS ESPECIAIS* nas colunas 11 a 29 da mesma linha 10.

6.4 FORMATOS IMPLÍCITOS

Além dos comandos *AT* e *TAB,* o ponto-e-vírgula ; e a vírgula , são passíveis de uso para formatação da tela.

O ponto-e-vírgula, que já foi mostrado anteriormente, aparentemente tem a função de elemento separador, como no exemplo:

```
PRINT AT 1Ø,5; "*"
```

Este delimitador, entretanto, também instrui o computador no sentido de *não mover o cursor na tela* (não mover o PRINT).

No exemplo

```
PRINT "CR$"; VALOR
```

o computador exibe a palavra *CR$* e, ao encontrar o delimitador, toma conhecimento de que deve deixar o PRINT onde parou e colocar o valor de *VALOR* logo a seguir, na mesma linha.

```
PRINT "JOAO"; "MANUEL"; "SILVA"
```

resulta na exibição de

```
JOAOMANUELSILVA
```

Por isto, é salutar sempre incluir um caracter branco entre as palavras e valores delimitados por ponto-e-vírgula. Se

A = 123.584 E B = Ø.ØØØ27

e, para evitar que

```
PRINT  A;B
```

resulte em

123.5840.0027

convém fazer

```
PRINT  A;" ";B
```

ou

```
PRINT  "JOAO";" MANOEL"; " SILVA"
```

ou

```
PRINT  "JOAO";" "; "MANOEL"; " "; "SILVA"
```

O conjunto de instruções

```
10  FOR I = 1 TO 10
20  PRINT "-";
30  NEXT I
```

resultam na exibição de 10 hífens numa mesma linha (----------), já que o ponto-e-vírgula da linha 20 impede que o "PRINT" mude de linha. Se não houvesse o ponto-e-vírgula, o computador colocaria um hífen em cada linha, usando, assim, 10 linhas.

Já o uso da vírgula faz com que o cursor seja movido para a primeira posição da zona seguinte, seja na mesma linha ou nas linhas seguintes.

A instrução

```
PRINT AT 10,1;  "VALOR:",  VALOR
```

fará com que o cursor do PRINT se posicione na linha 10, coluna 1, fique nessa posição (devido ao ;), mostre a palavra *VALOR* (colunas 10 a 14), salte para a coluna 16 da mesma linha (devido a ,) e, a partir dessa posição, exiba o valor numérico.

Se VALOR = 34547.50, tem-se

```
linha 10 → VALOR:          34547.50
             └→ COL. 1       └→ COL. 16
```

A vírgula permite a tabulação implícita nas duas zonas indicadas na figura 3.

É possível o uso de várias vírgulas, como em

```
PRINT AT 5,0;  "VALOR.1" , , , ,  "VALOR.2"
```


resultando em

```
LINHA 5  →      VALOR.1
LINHA 7  →      VALOR.2
                  └──→ COLUNA 0
```

6.5 COMANDO PAUSE

Esta instrução permite paralisar temporariamente a execução de um programa e, com isso, retardar a apresentação dos dados na tela.

Sua forma geral é:

PAUSE < EXPRESSAO ARITMETICA > **PAUSE**

A instrução *PAUSE* deve vir sempre acompanhada de um número, expressão aritmética ou variável que, dividido por 60, corresponde aproximadamente ao tempo, em segundos, de paralisação. Assim,

```
PAUSE 60   → PAUSE 60/60   →   1 SEGUNDO
PAUSE 180  → PAUSE 180/60  →   3 SEGUNDOS
PAUSE 600  → PAUSE 600/60  →  10 SEGUNDOS
```

O valor máximo que a *PAUSE* admite é de 9 minutos, o que corresponde ao valor 32767 para a expressão aritmética. Qualquer valor acima deste limite provoca uma pausa com tempo ilimitado.

É possível interromper uma PAUSE, independente de seu tempo, pelo toque de uma tecla qualquer do computador.

A *PAUSE* também pode ser utilizada para dar "ritmo" aos resultados, como no exemplo:

```
 10 FOR I=360 TO 30 STEP -30
 20 PRINT AT 11,15;CHR$ 128
 30 PAUSE I
 40 PRINT AT 11,15;" "
 50 NEXT I
 60 FOR I=30 TO 0 STEP -1
 70 PRINT AT 11,15;CHR$ 128
 80 PAUSE I
 90 PRINT AT 11,15;" "
100 NEXT I
```

Uma simulação do *PAUSE* pode ser conseguida com uso da instrução *FOR . . . NEXT,* como no exemplo:

```
100  FOR J = 1 TO 50
200  NEXT J
```

já que o computador "leva algum tempo" para a execução do comando *FOR,* com a vantagem de não piscar a tela.

Outro exemplo seria:

```
 50  FOR I = 1 TO 30
 60  PRINT AT 10,0; "SIMULACAO DO PAUSE"
 70  NEXT I
```

A instrução *PAUSE* é insubstituível quando o computador trabalha no modo *FAST* (veja manual do micro ou apêndice A)..

6.6 COMANDO CLS

Este comando "apaga" a tela, retirando todas as informações nela contidas (não altera valores de variáveis), e permitindo a exibição de novos dados.

Sua forma geral é

CLS **CLS**

Após cada comando *CLS,* o cursor do *PRINT* volta a assumir a posição (0,0), ou seja, primeira linha, primeira coluna.

Exemplo:

```
 10 PRINT "EM QUE ANO NASCEU ?"
 20 INPUT A$
 30 CLS
 40 PRINT "EM QUE MES ?"
 50 INPUT M$
 60 CLS
 70 PRINT "E, EM QUE DIA ?"
 80 INPUT D$
 90 CLS
100 PRINT "ENTAO, VOCE NASCEU E
M:",,,D$;"/";M$;"/";A$
```

6.7 COMANDO *SCROLL*

Quando o relatório gerado por um programa exceder as 22 linhas disponíveis na tela, devemos incluir um *PAUSE,* para "segurar" as

informações no tempo necessário para a sua assimilação, incluir um *CLS* para limpar a tela, e, então, liberar o programa para gerar novas informações. Uma alternativa seria a inclusão de um *INPUT* fictício, quando a imagem fica retida enquanto não for pressionado à tecla < *NEW LINE* > . Veja o exemplo:

```
. . . . . .
1500. . . . . .
1510 INPUT Z$
1520 CLS
1530  . . . . . .
. . . . . .
```

Outra alternativa é o uso da instrução *SCROLL,* que significa *rolamento.* Sua forma geral é

SCROLL **SCROLL**

e provoca um rolamento de toda a imagem da tela de uma linha para cima, liberando a última linha (a de baixo, de número 21) para novos dados. Dois *SCROLL* seguidos provocam dois rolamentos, e assim sucessivamente. O uso do comando *SCROLL* dentro de um *FOR NEXT* permite a exibição de dados de maneira ilimitada, pois a cada novo dado a ser exibido rola a imagem para cima. A tela, assim, nunca fica "cheia".

Exemplos:

```
. . . . . .
 40  . . . . . .
 45  FOR I = 1 TO 100
 50  . . . . . .
. . . . . .
110  SCROLL      (OU: IF I > 21 THEN SCROLL)
115  PRINT VALOR, VALOR * VALOR
120  NEXT I
125  . . . . . .
. . . . . .
```

6.8 TRUNCAMENTOS

O resultado de uma expressão aritmética sempre é exibido na tela com todas as casas decimais, até 8 dígitos significativos. Em muitas situações é indesejável, ou por falta de espaço numa linha, por se tratar de valores

monetários, ou por outro motivo, a exposição de todos os dígitos do número. Esta linguagem BASIC não dispõe, a exemplo de outras linguagens, de uma instrução que permita o tratamento e exibição de números com um número prefixado de casas decimais. Este objetivo, entretanto, pode ser alcançado com o uso da função *INT* (veja o item 2.6). Se *NUM* for uma variável cujo valor deve ser exposto com somente três casas decimais, podemos fazer:

```
LET NUM = INT (NUM * 1000) / 1000
```

ou

```
PRINT INT (NUM * 1000) / 1000
```

vejamos como funciona:

SE NUM = 34.748958

```
NUM*1000              → 34748.958
INT(NUM*1000)         → 34748
INT(NUM*1000)/1000    → 34.748
```

Para duas casas decimais, basta substituir 1000 por 100. Genericamente, seria

```
LET NUM = INT(NUM*10**X)/10**X
```

onde *X* representa o número de casas decimais desejado.

Nesta forma, o número é *truncado* após a última casa decimal. Se além disso desejarmos *arredondar* o último dígito com base no primeiro dígito truncado, podemos fazer:

```
LET NUM = INT(NUM * 10 ** X + .5)/10 ** X
```

assim, se NUM = 4346.6584 e para duas casas decimais, teríamos:

```
4346.6584*100              → 434665.84
436.6584*100 + 0.5         → 434666.34
INT(436.6584*100 + .5)     → 434666
INT(436.6584*100 + .5)/100 → 4346.66
```

6.9 EXERCÍCIOS

19) Escreva um programa que calcule e mostre na tela a raiz quadrada, o quadrado e o logaritmo dos números de 1 a 100. O programa deve exibir o cabeçalho e logo abaixo os resultados, como mostrado a seguir:

NÚM.	RAIZ. QUAD.	QUADRADO	LOGARIT.
1	1	1	0
2	1.414	4	0.693
3	1.732	9	1.098

use a instrução SCROLL.

20) Escreva um programa que "desenhe" 8 retângulos na tela (um dentro do outro, e separados por branco). O retângulo externo deve ser formado por zeros, o mais interno, por uns (1), e assim sucessivamente.

21) Altere o programa anterior para que desenhe o retângulo "caractere por caractere", com uma pausa entre um retângulo e outro.

22) Faça um programa que leia um conjunto de valores numéricos e, com base no maior valor lido, reduza-os para que possam ser representados na tela em forma de barras horizontais formadas por *X*. O valor maior deve ter 32 X e o menor 1 X.

6.10 DESAFIO

Analise, procure entender, rode e tente explicar os seguintes programas:

```
  1 REM >>>PROGRAMA 19<<<
 10 LET K=0
 20 PRINT AT 20,5;"TABELA DE NU
MEROS QUADRADOS"
 30 SCROLL
 40 PRINT TAB 6;"NUMERO INICIAL
 ?";
 50 INPUT INICIO
 60 PRINT " - ";INICIO
 70 SCROLL
 80 PRINT TAB 6;"NUMERO FINAL ?
";
 90 INPUT FIM
100 PRINT " - ";FIM
105 FOR L=1 TO 10
110 SCROLL
112 NEXT L
130 PRINT "DIGITE <NEW LINE> P/
CONTINUAR"
135 INPUT W$
140 FOR L=1 TO 22
145 SCROLL
```

```
148 NEXT L
149 PRINT AT 0,0;
150 LET K=0
160 FOR X=INICIO TO FIM
170 LET K=K+1
180 IF K>21 THEN SCROLL
190 PRINT TAB 5;X;" :QUADRADO -
> ";X*X
200 NEXT X
```

```
  1 REM >>>PROGRAMA 20<<<
 10 LET C=31
 20 LET D=20
 30 LET A=INT (D*RND)
 40 LET B=INT (C*RND)
 50 PRINT AT A,B;"X"
 60 PRINT AT A,D-A;"0"
 70 PRINT AT D-A,B;"*"
 80 PRINT AT D-A,C-B;"+"
 90 RUN
```

```
  1 REM >>>PROGRAMA 21<<<
  5 PRINT "USE AS TECLAS <Z> E
<M>"
 10 LET K=2
 20 LET A=10
 30 LET B=A
 40 LET C=A+A
 50 LET D=10
 60 PRINT AT C,D;CHR$ 128;CHR$
136;CHR$ 136;CHR$ 136;CHR$ 136;C
HR$ 128
 80 PRINT AT A,B;CHR$ 136
 90 SCROLL
100 IF INKEY$="M" THEN LET B=B+
1
110 IF INKEY$="Z" THEN LET B=B-
1
120 PRINT AT A,B;CHR$ 187
130 IF D>17 OR D<7 THEN LET K=-
K
140 LET D=D+RND*K
150 GOTO 60
```


7 SUB-ROTINAS

7.1

COMANDOS GOSUB E RETURN

Uma maneira de otimizar um programa quando um conjunto de comandos (compreendendo um trecho do programa) repete-se duas ou mais vezes é pelo uso de sub-rotinas. O uso da sub-rotina facilita a programação, diminui o "tamanho" do programa e economiza a memória do computador.

Uma sub-rotina nada mais é do que certo trecho do programa que é acessado tantas vezes quanto necessário. Escreve-se somente uma vez e usa-se várias vezes.

A sub-rotina pode estar localizada em qualquer parte do programa, no início, no meio ou no final. A posição mais usual, e a que mais facilita a programação, é a final.

Para indicar ao computador que saia do processamento seqüêncial e se dirija ao trecho do programa que compõe a sub-rotina, usa-se o comando

GOSUB < N > **GOSUB**

onde N representa o número da linha do programa onde começa o trecho compondo a sub-rotina. O *N* também pode ser uma expressão algébrica.

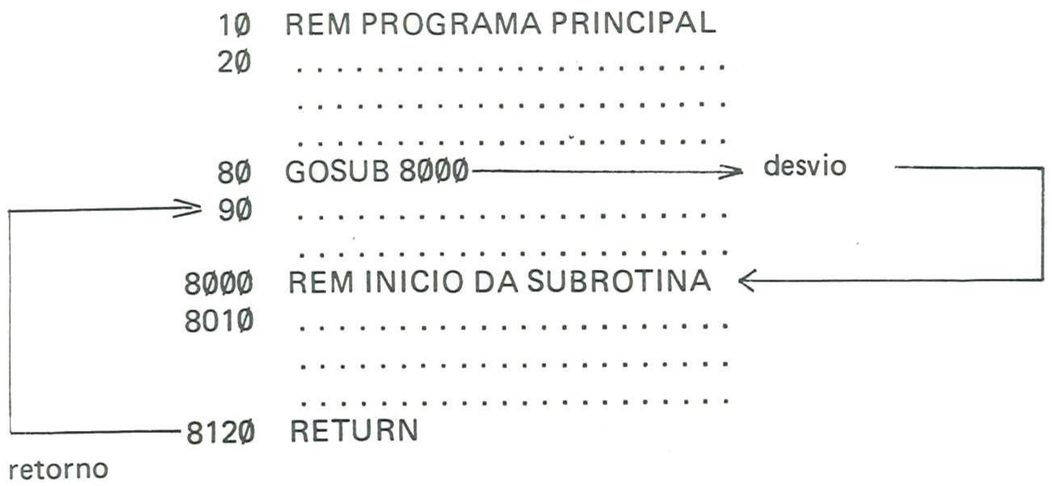
O final do trecho repetitivo ou sub-rotina é indicado ao computador através da instrução

RETURN **RETURN**

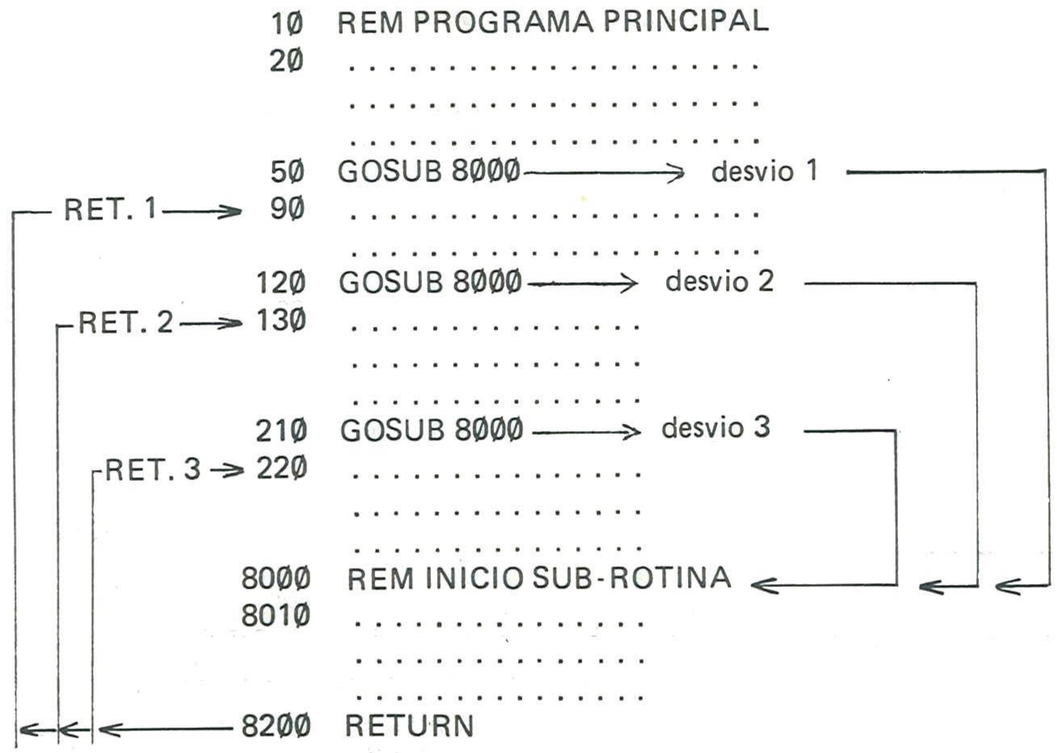
Este comando, na realidade, é uma instrução ao computador para retornar à linha seguinte ao *GOSUB* que o desviou a esta sub-rotina.

Não existe limitação quanto ao número de sub-rotinas num programa, nem no número de vezes que uma sub-rotina possa ser acessada. Quando o computador encontra um comando *GOSUB,* ele "guarda" o número da linha, para retornar à linha seguinte, ao encontrar a primeira instrução *RETURN*.

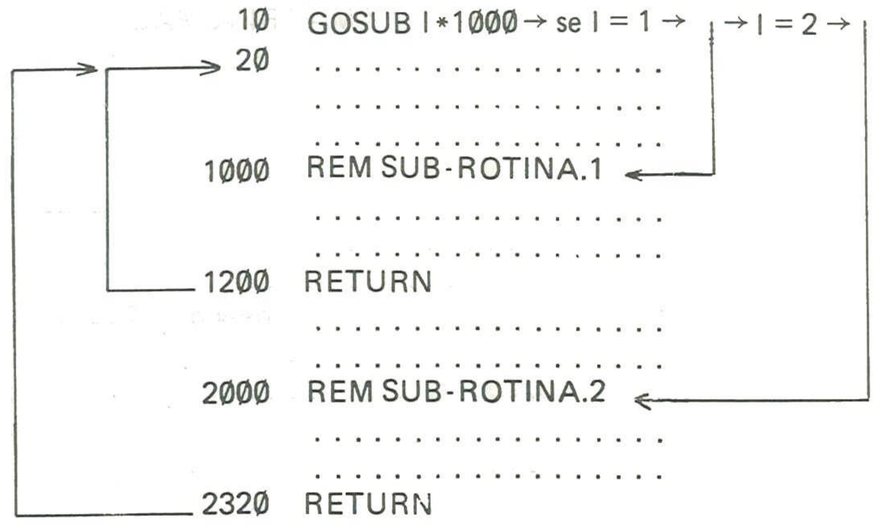
Representação gráfica de uma sub-rotina:



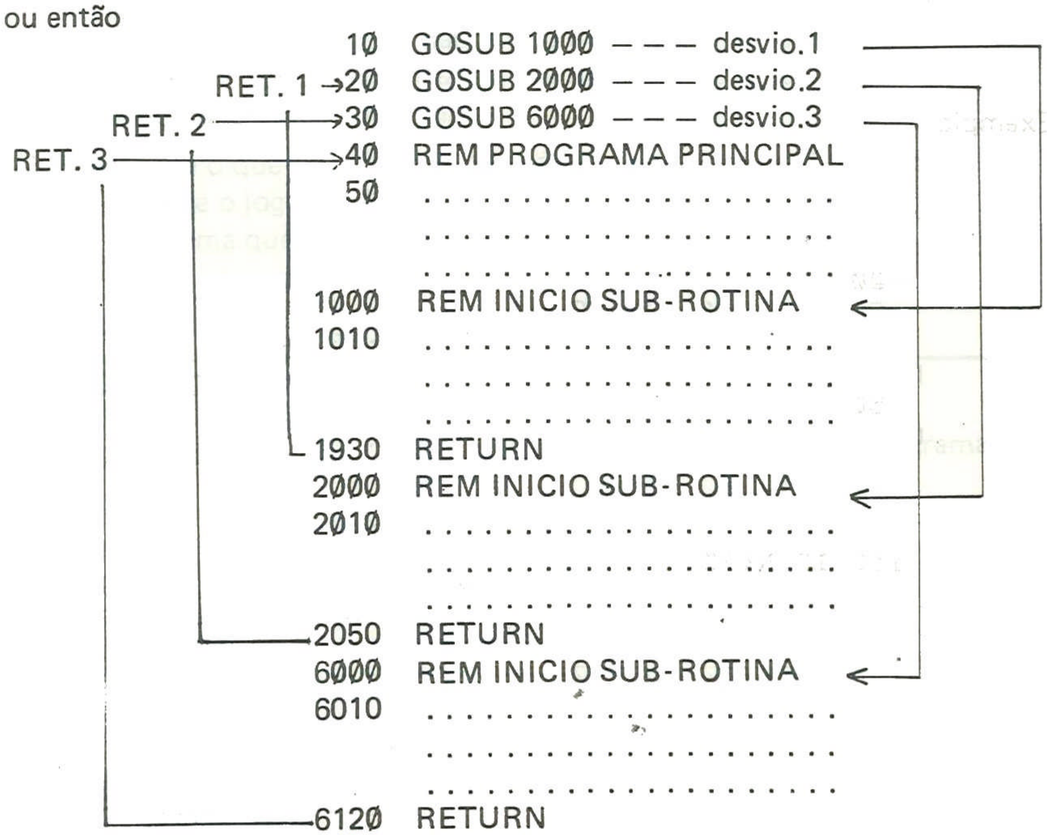
Outras possibilidades no uso de sub-rotinas:



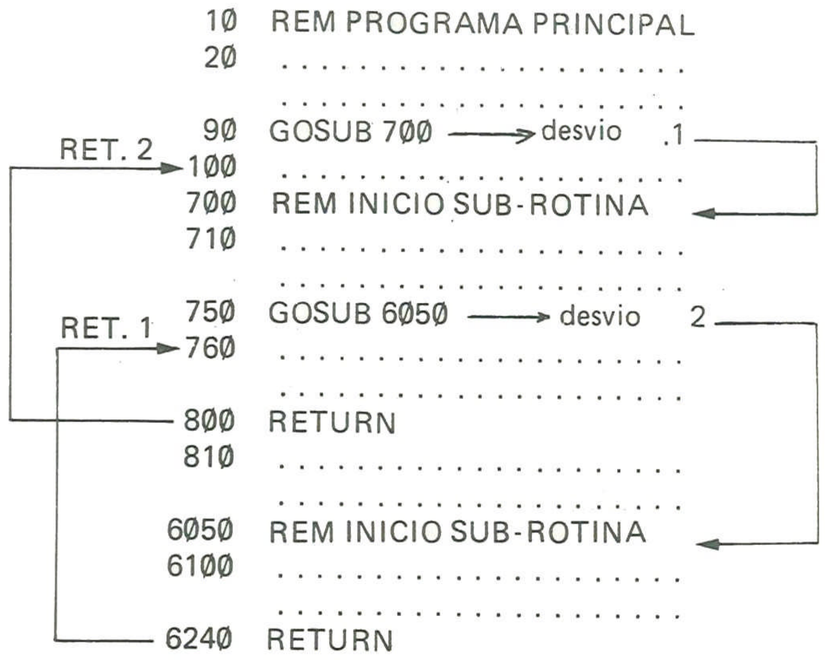
ou então



ou então



ou ainda



Exemplo:

```
   5 REM >>>PROGRAMA PRINCIPAL<<
<
  10 LET N=0
  20 GOSUB 100
  30 FOR N=1 TO 100
  40 PRINT N;TAB 7;N*N;TAB 17;SQ
R N
  50 IF N/18=INT (N/18) THEN GOS
UB 100
  60 NEXT N
  65 STOP
 100 REM >>>INICIO SUBROTINA<<<
 110 IF N<>0 THEN PRINT "
                        "
 120 IF N<>0 THEN PAUSE 300
 130 CLS
 140 PRINT "NUM QUADRADO  RAIZ Q
UADRADA"
 150 PRINT "-------------------
-------"
 160 RETURN
 170 REM >>>FIM SUBROTINA<<<
```

Este programa calcula o quadrado e a raiz quadrada dos números de 1 a 100. A cada 18 valores (uma tela), o programa transfere o

processamento à sub-rotina 100 (vide linha 50), que faz o "fechamento" da tela, introduz uma pausa de 10 segundos, limpa a tela e monta o cabeçalho da próxima tela.

7.2 EXERCÍCIOS

23) Elabore um programa que simule o lançamento de um dado. Crie uma sub-rotina que gere um número aleatório entre 1 e 6 (vide item 2.6). O programa deve perguntar quantos lançamentos devem ser simulados, e mostrar o resultado ocorrido.

24) Altere o programa anterior para simular o lançamento simultâneo de 5 dados. O programa deve mostrar os resultados dos 5 dados, numa só vez.

25) Altere o programa anterior, para que o programa pergunte se é desejado repetir o lançamento de um ou mais dados. Se for indicado, por exemplo, os dados 2, 3 e 5, gerar três novos lançamentos que devem substituir os três resultados anteriores, deixando inalterados os demais.

26) Aproveite o que foi feito nos programas anteriores e elabore um que simule o jogo do "general", faltando somente montar a parte do programa que faça a contagem dos pontos.

7.3 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
   1 REM >>>PROGRAMA 22<<<
  10 GOSUB 5000
  15 LET T$="-------------------
-------"
  20 PRINT "APRENDA A DIZER FRAS
ES DE EFEITO"
  30 PRINT AT 19,0;"DIGITE UM NU
MERO ENTRE 111 E 888"
  60 PRINT "DIGITE <0> P/DESCONT
INUAR"
  70 INPUT N
  75 IF N=0 THEN STOP
  80 IF N<111 OR N>888 THEN GOTO
 70
  90 LET A=INT (N/100)
```

```
 100 LET B=1
 110 GOSUB 2000
 120 LET A=INT ((N-((INT (N/100)
)*100))/10)
 130 LET B=2
 140 GOSUB 2000
 150 LET A=N-(INT (N/10)*10)
 160 LET B=3
 170 GOSUB 2000
 180 CLS
 185 PRINT T$
 190 PRINT F$( TO 26);TAB 5;F$(2
7 TO )
 192 PRINT T$
 195 LET F$=""
 200 GOTO 30
2000 IF B=1 THEN LET F$=F$+U$(A*
14 TO A*14+13)
2010 IF B=2 THEN LET F$=F$+U$(A*
14+112 TO A*14+125)
2020 IF B=3 THEN LET F$=F$+U$(A*
14+224 TO A*14+237)
2030 RETURN
5000 LET F$=""
5010 LET U$=".............PROGRA
MACAO...ESTRATEGIA....MOBILIDADE
....PLANIFICACAO..DINAMICA......
FLEXIBILIDADE.IMPLEMENTACAO.RETR
OACAO.....PROJETIVA.....FUNCIONA
L.....OPERACIONAL...DIMENSIONAL.
..ESTRUTURAL....GLOBAL........DI
RECIONAL....OPCIONAL......CABALI
STICA...LOGISTICA.....SISTEMATIC
A...INTEGRADA.....EQUILIBRADA...
TOTALIZADA....INSUMIDA......BALA
NCEADA....COORDENADA....COMBINAD
A.....ESTABELIZADA.."
5020 RETURN
```

SUBSTITUIR OS PONTOS POR BRANCOS

```
   1 REM >>>PROGRAMA 23<<<
  20 GOTO 210
  30 LET T=0
  40 IF B-3>=0 THEN LET T=2
  50 LET B=B+(T=2)
  60 IF T=2 THEN GOTO 90
  70 LET A=A-1
  80 LET B=B+13
  90 LET E=INT (365.25*A)+INT (3
0.6*B)+C
 100 RETURN
 110 LET T=0
```



```
 120 LET F=G-INT (G/D)*D
 130 IF F>D/2 THEN LET T=2
 140 IF T=2 THEN LET H=D-F
 150 IF T=2 THEN GOTO 180
 160 IF F=0 OR D/2=F THEN PRINT
"HOJE"
 170 IF F=0 OR D/2=F THEN RETURN

 180 LET H=D/2-F
 190 PRINT "APOS ";H;" DIAS"
 200 RETURN
 210 LET Y$="O PRIMEIRO DIA E "
 220 LET X$=" CICLO"
 230 CLS
 240 PRINT "DIGITE O SEU NOME ";

 242 INPUT N$
 243 PRINT N$
 244 PRINT "DIGITE O ANO DO NASC
IMENTO","(COMO 1964)"
 250 INPUT A
 260 PRINT "DIGITE O MES DO NASC
IMENTO","(COMO 12)"
 270 INPUT B
 280 PRINT "AGORA DIGITE O DIA",
"(COMO 30)"
 290 INPUT C
 300 CLS
 310 GOSUB 30
 320 LET J=E
 330 PRINT "DIGITE, AGORA, PARA
QUE DIAS DESEJAS O BIORITMO"
 340 PRINT "DIA ? ";
 345 INPUT C
 350 PRINT " - MES ? ";
 355 INPUT B
 360 PRINT " - ANO ? ";
 365 INPUT A
 370 LET Z=A
 380 IF A<1900 OR A>1999 THEN PR
INT "1900<=ANO<=1999"
 385 IF A<1900 OR A>1999 THEN GO
TO 365
 390 LET Q=B
 400 GOSUB 800
 420 LET P=C
 425 GOSUB 30
 430 CLS
 440 LET G=E-J
 442 CLS
 443 PRINT "BIORITMO PARA: ";N$
 444 PRINT "PERIODOS POSITIVOS"
```

```
445 PRINT
450 PRINT "DATA: ";P;"/";Q;"/";

452 IF Z-1900<10 THEN PRINT "0"
;
455 PRINT Z-1900
460 PRINT
480 LET D=23
490 PRINT X$;" FISICO"
495 PRINT "------------"
500 PRINT Y$
510 GOSUB 110
520 PRINT
530 LET D=28
540 PRINT X$;" EMOCIONAL"
545 PRINT "---------------"
550 PRINT Y$
560 GOSUB 110
570 PRINT
580 LET D=33
590 PRINT X$;" MENTAL"
595 PRINT "------------"
600 PRINT Y$
610 GOSUB 110
620 PRINT
630 PRINT "OUTRA DATA (S/N)"
640 INPUT U$
650 CLS
660 IF U$="S" THEN GOTO 330
670 IF U$<>"S" AND U$<>"N" THEN
GOTO 630
800 CLS
810 FOR I=1 TO 96
820 PRINT CHR$ 0;CHR$ 0;CHR$ 12
8;
830 NEXT I
840 RETURN
```


8 ARRANJOS E TABELAS

8.1 ARRANJOS NUMÉRICOS

O *arranjo* corresponde a um conjunto ordenado de dados, valores ou informações. Os nomes dos funcionários de uma empresa constituem um arranjo. O conjunto ordenado dos seus salários também é um arranjo. Os elementos de um arranjo devem possuir sempre uma característica comum.

A linguagem BASIC, como a matemática, admite o uso de variáveis com índices, chamadas de *variáveis subscritas,* ou *variáveis indexadas.* Assim, a matemática, para diferenciar quatro valores para a variável *B,* lança mão dos índices, definindo

B_1, B_2, B_3, e B_4
Em BASIC, seria

B(1), (B(2), B(3) e B(4)

onde *B* é o nome do arranjo e 1, 2, 3 e 4, os índices ou subscritos.

Se uma ficha contém 10 notas de matemática de determinado aluno, o conjunto de notas pode ser representado pelo arranjo *M*. A primeira nota seria representada por *M(1),* a segunda por *M(2)* e a última por *M(10).*

A grande vantagem do uso de variáveis subscritas reside no fato de permitir o uso de um único identificador (variável) para representar vários valores de um arranjo (no caso, o identificador *M*).

Cada elemento do conjunto é representado por um subscrito que representa, no caso, um teste de avaliação de matemática.

O arranjo *M* pode ser representado:

ARRANJO	M	6	8	5	...	6
SUBSCR.	Nº TESTE	1	2	3	...	10

Assim como na matemática, os arranjos em BASIC podem ser unidimensionais, bidimensionais, tridimensionais etc. O arranjo unidimensional, que corresponde ao exemplo citado, também é

chamado de *vetor,* onde

M(1) = 6, M(2) = 8, M(10) = 9

Um arranjo bidimensional corresponde a uma tabela de duas dimensões:

		COLUNAS					
		1	2	3	.....		10
LINHAS	1	6	8	5	.....		9
	2	4	5	4	.....		4
	3	6	6	6	.....		7

Para a representação de uma tabela é necessário o uso de dois subscritos para identificar um elemento; o primeiro para indicar o número da linha e o segundo o número da coluna. *(2,3)* indica a segunda linha e a terceira coluna, que na tabela acima corresponde ao elemento 4.

Considerando o exemplo abordado, o primeiro subscrito pode indicar o número seqüêncial dos alunos e o segundo o número do teste de matemática. Assim, se a linha 1 representar o aluno 1, a linha 2, o aluno 2, e assim sucessivamente, é possível condensar todas as notas de matemática de todos os alunos de uma escola.

Assim

M(2,1Ø) = 4

indica que a décima nota do segundo aluno é 4. Observe que os subscritos devem estar separados por vírgula.

Imagine, agora, um arquivo de aço com 8 gavetas, cada uma contendo 500 fichas. Cada ficha relaciona as 10 notas de 50 alunos, e cada gaveta do arquivo, uma matéria diferente. Para representar a nota de uma avaliação de um aluno, numa determinada matéria, necessitamos de três subscritos. Como cada ficha pode conter somente 50 alunos, podemos usar um quarto subscrito para representar o número da ficha (sua posição de ordem dentro da gaveta). Teremos, assim,

ÍNDICE 1: NÚMERO DA GAVETA = NOME DA MATÉRIA
ÍNDICE 2: NÚMERO DA FICHA
ÍNDICE 3: NÚMERO DA LINHA DA FICHA = NOME DO ALUNO
ÍNDICE 4: NÚMERO DA COLUNA DA FICHA = NÚMERO DO TESTE

Se a quarta gaveta corresponder às notas de física, e se chamarmos este arranjo de quatro dimensões de *N,*

N (4,15Ø,2Ø,6) = 1Ø

indica que a ficha número 150 da quarta gaveta relaciona na vigésima linha o nome de um aluno que obteve grau 10 no sexto teste de física.

8.2 DECLARAÇÃO DE ARRANJOS

Para usar um arranjo é necessário primeiro declará-lo, indicando seu nome, sua dimensão e número máximo de elementos. Esta declaração é feita via instrução

DIM < NOME ARRANJO > (A,B,C..) **DIM**

onde *A* representa o número máximo de elementos para um arranjo unidimensional, *A,B* para um bidimensional, e assim sucessivamente.

A declaração *DIM* pode ser especificada em qualquer parte do programa e, ao encontrá-la, o computador reserva uma área de memória para o arranjo e o inicializa (zera todas as variáveis).

Algumas regras no uso da instrução *DIM*:

a) Reserva área de memória para todos os elementos do arranjo;
b) Inicializa as variáveis do arranjo;
c) Cancela qualquer arranjo já existente com o mesmo nome;
d) Provoca erro-4 se não houver espaço suficiente na memória;
e) Provoca erro-3 se foi subdimensionado.

Exemplos:

DIM M(3Ø) → Arranjo numérico unidimensional com no máximo 30 elementos

DIM O(8,3Ø,5Ø) → Arranjo tridimensional, de nome *O* com um máximo global de 8 x 30 x 50 = 12000 elementos

Outras possibilidades:

```
15Ø  IF CHAVE = 1 THEN DIM X(2ØØ)
2ØØ  DIM S(A)
 75  DIM B(A*2 + 3Ø)
```

8.3 ARRANJOS STRING

Da mesma maneira que um arranjo pode ser usado para trabalhar e/ou armazenar valores (números), também pode ser criado um arranjo para armazenar STRINGS (caracteres alfanuméricos).

Para diferenciar o arranjo STRING do numérico, é necessário colocar o

símbolo *$* atrás da letra (única) que representa o nome do arranjo. Exemplos:

A$(1Ø,3Ø)
W$(3Ø,5Ø)
H$(1ØØ,1)

O arranjo STRING tem sempre uma dimensão a mais que o numérico, onde o último subscrito indica o número de caracteres. A seguinte tabela STRING

PEDRO	JOAO	ANTONIO
ANTUNES	PEREIRA	SILVA
MARIA	ISABEL	ELISA

deve ter a dimensão

A$(3, 3, 7)

Se um arranjo STRING for composto por somente um caractere, a segunda dimensão pode ser omitida.

N$(7Ø4,1) = N$(7Ø4)

O arranjo N$(1Ø,3Ø) pode corresponder, por exemplo, a um cadastro de nomes de pessoas, com no máximo 10 nomes de no máximo 30 caracteres cada nome (não esqueça de que o espaço em branco é considerado como caractere).

Na hora de referenciar o arranjo, o segundo subscrito pode ser omitido. Veja o seguinte exemplo:

```
1Ø DIM N$(1Ø,3Ø)
2Ø FOR I = 1 TO 1Ø
3Ø INPUT N$(I)
4Ø NEXT I
```

A cada *INPUT N$,* o computador espera um conjunto de 30 caracteres. Se for digitado um nome com mais de 30 caracteres, somente os 30 primeiros serão lidos.

O conceito de substring, visto em 5.2, também se aplica aqui. Veja o exemplo abaixo:

```
1Ø DIM N$(1Ø,8Ø)
2Ø DIM A$(1Ø,4Ø)
3Ø DIM B$(1Ø,4Ø)
4Ø FOR I = 1 TO 1Ø
5Ø INPUT N$(I)
6Ø LET A$(I) = N$(I, TO 4Ø)
7Ø LET B$(I) = N$(I, 41 TO)
8Ø NEXT I
```


onde os quarenta primeiros caracteres do arranjo N$ representam nomes de pessoas (formando posteriormente o arranjo A$) e os últimos 40 caracteres o seu endereço (formando o arranjo B$).

Um arranjo STRING ocupa menos memória que um arranjo numérico, de mesma dimensão. Assim, em programas extensos, é possível, para economizar memória, o uso de um arranjo STRING para armazenar valores numéricos, desde que se faça a devida conversão via instrução *VAL*.

Exemplo:

```
1Ø  LET  A$ = "125354ØØ212Ø64Ø75ØØ483ØØ24Ø9ØØ"
11  FOR I = 1 TO 3Ø STEP 3
12  LET  A = VAL A$(I TO I + 2)
13   . . . . . .
     . . . . . .
     . . . . . .
30  NEXT I
```

Neste exemplo, a função *VAL* da linha 12 converte o substring *125* (na primeira passada) no valor numérico 125, e o preserva com o nome *A*.

8.4 EXEMPLOS DE APLICAÇÃO

a) O seguinte programa calcula a média aritmética, a amplitude (diferença entre o maior e o menor) de um conjunto amostral de *N* valores, e a diferença entre cada valor e a média aritmética.

```
  5 PRINT "QUANTOS VALORES ";
  6 INPUT N
  7 PRINT N
 10 LET MENOR=999999
 20 LET MAIOR=0
 30 LET SOMA=0
 40 DIM V(N)
 50 FOR I=1 TO N
 55 PRINT I;" ";
 60 INPUT V(I)
 70 LET SOMA=SOMA+V(I)
 80 IF V(I)<MENOR THEN LET MENO
R=V(I)
 90 IF V(I)>MAIOR THEN LET MAIO
R=V(I)
100 NEXT I
105 LET MEDIA=SOMA/N
```

```
 106 CLS
 110 PRINT "MEDIA ARITMETICA = "
;MEDIA
 120 PRINT "AMPLITUDE = ";MAIOR-
MENOR
 130 PRINT "--------------------
-----------"
 135 PAUSE 300
 140 FOR I=1 TO N
 150 IF I>18 THEN SCROLL
 160 PRINT I;TAB 5;"--->";TAB 13
;V(I)-MEDIA
 170 NEXT I
```

b) O programa deste exemplo lê um número não predefinido de valores (mas de máximo 100), ordena-os em ordem não-decrescente, exibindo na tela, após a ordenação, O fim da entrada de dados é indicado ao programa, com a digitação de um valor negativo.

```
  1 REM PROGRAMA CLASSIFICADOR
  2 REM         NUMERICO
 12 REM MAXIMO DE 100 VALORES
 13 REM
 20 DIM A(100)
 30 LET I=0
 40 LET I=I+1
 50 INPUT A(I)
 60 IF I=101 THEN GOTO 90
 70 IF A(I)>=0 THEN GOTO 40
 80 REM >>INICIO ORDENACAO<<
 85 FAST
 90 LET FIM=I-1
100 LET CHAVE=0
110 FOR J=1 TO FIM-1
120 IF A(J)<=A(J+1) THEN GOTO 1
70
130 LET TROCA=A(J+1)
140 LET A(J+1)=A(J)
150 LET A(J)=TROCA
160 LET CHAVE=1
170 NEXT J
180 IF CHAVE=1 THEN GOTO 100
190 REM >>FIM ORDENACAO<<
195 SLOW
200 FOR J=1 TO FIM
210 PRINT A(J);"-";
220 NEXT J
```


A *chave* do programa está na variável *CHAVE*. O trecho do programa compreendido entre as linhas 100 e 180 é repetido enquanto houver um valor desordenado (testado na linha 120). Quando dois valores seqüenciais estão ordenados, o *IF* da linha 120 desvia o processamento para a linha 170, resultando em *CHAVE = 0*. Enquanto houver um valor desordenado, a variável *CHAVE* assume o valor unitário e todo o processo de ordenação é repetido (linha 180).

Uma característica muito útil da BASIC é a possibilidade de ordenação de variáveis STRINGS. Assim, é possível ordenar alfabeticamente um conjunto de nomes, endereços ou telefones. Se N$(9) = "MANOEL" e N$(10) = "ADAO",

N$(10) < N$(9)

c) Se N$(100,30) representa um arranjo STRING de 100 nomes com no máximo 30 caracteres, o programa ordenador pode ser:

```
 10 REM PROGRAMA CLASSIFICADOR
 11 REM         ALFABETICO
 12 REM MAXIMO DE 100 VALORES
 20 REM INDICADOR DE FINAL -> B
RANCO
 25 DIM N$(100,30)
 30 LET I=0
 40 LET I=I+1
 50 INPUT N$(I)
 60 IF I=101 THEN GOTO 90
 70 IF N$(I,1)<>" " THEN GOTO 4
0
 80 REM >>INICIO ORDENACAO<<
 85 FAST
 90 LET FIM=I-1
100 LET CHAVE=0
110 FOR J=1 TO FIM-1
120 IF N$(J)<=N$(J+1) THEN GOTO
170
130 LET T$=N$(J+1)
140 LET N$(J+1)=N$(J)
150 LET N$(J)=T$
160 LET CHAVE=1
170 NEXT J
180 IF CHAVE=1 THEN GOTO 100
190 REM >>FIM ORDENACAO<<
195 SLOW
200 FOR J=1 TO FIM
205 SCROLL
210 PRINT N$(J)
220 NEXT J
```

d) O seguinte programa permite a criação das tabelas abaixo e a pesquisa de determinado nome.

TABELA A	TABELA B	TABELA C
NOME (até 30 caracteres)	ENDERECO (até 40 caracteres)	TELEFONE (até 10 caract.)
DIM N$(50,30)	DIM E$(50,40)	DIM F$(50,10)
JOSE NEI PINOTI	TUIUTI, 345/401	30.4578
MARCO ANTONIO CASTRO	CRISTOVAO COLOMBO, 208	21.3131
PEDRO ARANTES	RAMIRO BARCELOS	34.6781
.....	.....	.....
.....	.....	.....
até 50 nomes		

```
 10 REM ---ENTRADA DE DADOS----
 20 PRINT AT 21,0;"DIGITE <999>
AO FINAL"
 30 DIM N$(50,30)
 40 DIM E$(50,40)
 50 DIM F$(50,10)
 60 FOR I=1 TO 50
 70 PRINT AT 0,0;I;" NOME     "
 80 INPUT N$(I)
 90 IF N$(I, TO 3)="999" THEN G
OTO 230
120 PRINT AT 0,0;I;" ENDERECO
          "
130 INPUT E$(I)
170 PRINT AT 0,0;" TELEFONE
          "
180 INPUT F$(I)
220 NEXT I
230 LET FIM=I-1
240 CLS
250 PRINT "PESQUISA ? <S/N> ";
260 INPUT W$
270 IF W$="N" THEN STOP
280 IF W$<>"S" THEN GOTO 250
290 REM ---INICIO PESQUISA---
300 PRINT "OK"
310 FOR I=1 TO 50
320 NEXT I
500 REM ---PESQUISA DO NOME---
510 CLS
520 PRINT "DIGITE O NOME PROCUR
ADO"
530 INPUT W$
540 PRINT W$
545 FOR I=1 TO 30
546 NEXT I
```



```
550 FAST
555 FOR I=1 TO FIM
560 IF W$<>N$(I, TO LEN W$) THE
N GOTO 730
570 REM ---ENCONTRADO O NOME---
580 CLS
590 SLOW
600 PRINT "NOME:  ";N$(I)
610 PRINT "ENDERECO:",,E$(I)
620 PRINT "TELEFONE:  ";F$(I)
630 PRINT AT 21,0;"CONTINUO A P
ESQUISA ? <S/N>"
640 INPUT Q$
650 IF Q$="S" THEN GOTO 730
660 IF Q$<>"N" THEN GOTO 630
670 PRINT "NOVA PESQUISA ? <S/N>"
680 INPUT Q$
690 IF Q$="N" THEN STOP
700 IF Q$<>"S" THEN GOTO 670
710 CLS
720 GOTO 510
730 NEXT I
735 SLOW
736 CLS
740 PRINT "NOME(OU OUTROS NOMES
) NAO ENCONTRADO"
750 PRINT "LISTAGEM DOS NOMES ?
<S/N>"
760 INPUT W$
770 CLS
780 IF W$="N" THEN GOTO 320
790 IF W$<>"S" THEN GOTO 750
800 FAST
805 FOR I=1 TO FIM
810 SCROLL
820 PRINT N$(I)
830 PAUSE 60
835 NEXT I
840 SLOW
845 SCROLL
846 SCROLL
860 PRINT "P/CONTINUAR DIGITE <
NEW LINE>"
870 INPUT W$
890 GOTO 510
1000 SAVE "CADASTRO"
1010 GOTO 500
```

O trecho do programa compreendido entre as linhas 10 e 310 é usado somente uma vez, para a criação das tabelas. Concluída esta operação, o programa e as tabelas podem ser preservados em fita magnética, através da instrução "Goto 1000".

8.5 EXERCÍCIOS

27) Incremente o programa do exemplo d) do item 8.4 para permitir a inclusão de novos nomes e a exclusão de antigos.

28) Elabore um programa que leia 20 nomes de pessoas de no máximo 25 caracteres. Se for digitado um nome com mais de 25 caracteres, o programa deve indicar o fato, exibindo os caracteres que foram truncados.

29) Faça um programa que cadastre as 5 notas anuais de 12 matérias de uma escola com 200 alunos. Dado o número de um aluno, o programa deve relacionar as 5 notas de cada matéria, calcular a média da matéria e a média final.

8.6 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
  1 REM >>>PROGRAMA 24<<<
  5 GOTO 500
 10 LET L=0
 20 LET C=0
 30 PRINT AT L,C;"."
 40 PRINT AT L,C;" "
 50 IF INKEY$="" THEN GOTO 30
 60 LET A$=INKEY$
100 IF A$<>"W" AND A$<>"K" AND
A$<>"Y" THEN PRINT AT L,C;A$
110 IF A$="K" THEN PRINT AT L,C
;" "
115 IF A$="Y" THEN GOTO 600
120 IF A$="W" THEN GOTO 800
125 IF A$="0" THEN GOTO 900
130 LET C=C+1
140 IF C<32 THEN GOTO 30
150 LET C=0
160 LET C=C+1
165 IF L=20 THEN GOTO 900
170 GOTO 30
500 PRINT "MINI EDITOR DE TEXTO
S"
510 PRINT "K=ESPACO EM BRANCO",
"W=NOVA LINHA",,"Y=RETORNO"
520 PRINT
530 PRINT "O PROGRAMA PRESERVA
```



```
O CONTEUDO","DE ATE 5 TELAS"
540 PRINT
545 PRINT "PARA PRSERVAR UMA TE
LA","DIGITE <0>"
560 LET TELA=1
570 LET B$="K=BRANCO W=N.LINHA
Y=RETORNO"
575 PRINT AT 21,0;B$
580 FOR I=1 TO 500
581 NEXT I
590 FOR I=0 TO 20
592 PRINT AT I,0;"
                  "
594 NEXT I
595 GOTO 10
600 IF C=0 THEN GOTO 700
610 LET C=C-1
620 GOTO 30
700 LET L=L-1
710 LET C=31
720 GOTO 30
800 LET L=L+1
810 LET C=0
820 GOTO 30
900 PRINT AT 21,0;"
                  "
910 GOSUB TELA*10+1490
915 FAST
920 FOR Z=0 TO 21
925 FOR Q=1 TO 32
930 GOSUB TELA*10+1990
935 NEXT Q
940 NEXT Z
945 SLOW
950 CLS
955 PRINT "TELA PRESERVADA"
956 PRINT
957 PRINT "DIGITE <C> P/CONTINU
AR","       <P> P/PARAR","
<R> P/RECUPERAR TELAS"
958 INPUT I$
959 IF I$="C" THEN GOTO 970
960 IF I$="P" THEN STOP
961 IF I$="R" THEN GOTO 8000
962 GOTO 958
970 CLS
975 PRINT AT 21,0;B$
980 LET TELA=TELA+1
985 GOTO 10
1500 DIM C$(704)
1505 RETURN
1510 DIM D$(704)
```

```
1515 RETURN
1520 DIM E$(704)
1525 RETURN
1530 DIM F$(704)
1535 RETURN
1540 DIM G$(704)
1545 RETURN
1550 STOP
2000 LET C$(Q+32*Z)=CHR$ PEEK ((
PEEK 16396+256*PEEK 16397)+Q+33*
Z)
2005 RETURN
2010 LET D$(Q+32*Z)=CHR$ PEEK ((
PEEK 16396+256*PEEK 16397)+Q+33*
Z)
2015 RETURN
2020 LET E$(Q+32*Z)=CHR$ PEEK ((
PEEK 16396+256*PEEK 16397)+Q+33*
Z)
2025 RETURN
2030 LET F$(Q+32*Z)=CHR$ PEEK ((
PEEK 16396+256*PEEK 16397)+Q+33*
Z)
2035 RETURN
2040 LET G$(Q+32*Z)=CHR$ PEEK ((
PEEK 16396+256*PEEK 16397)+Q+33*
Z)
2045 RETURN
8000 CLS
8010 FOR I=1 TO TELA
8020 PRINT AT 0,10;"TELA - ";I
8025 FOR J=1 TO 30
8026 NEXT J
8030 CLS
8035 GOSUB I*100+8400
8040 PRINT AT 21,0;"DIGITE <NEW
LINE> P/CONTINUAR"
8045 INPUT O$
8050 NEXT I
8055 STOP
8500 PRINT C$
8510 RETURN
8600 PRINT D$
8610 RETURN
8700 PRINT E$
8710 RETURN
8800 PRINT F$
8810 RETURN
8900 PRINT G$
8910 RETURN
```


9 GRÁFICOS

9.1 SÍMBOLOS GRÁFICOS

A linguagem BASIC da linha SINCLAIR dispõe de um conjunto de 22 símbolos gráficos:

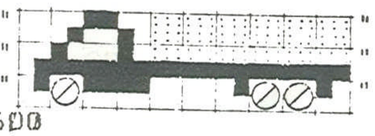
SIMB	TECLA P/OBTENÇÃO	CÓDIGO	SIMB	TECLA P/OBTENÇÃO	CÓDIGO
▗	G + SHIFT + 1	1	▟	G + SHIFT + R	132
▖	G + SHIFT + 2	2	▚	G + SHIFT + T	6
▄	G + SHIFT + 3	135	▞	G + SHIFT + Y	134
▙	G + SHIFT + 4	4	▒	G + SHIFT + A	8
▌	G + SHIFT + 5	5	▀	G + SHIFT + S	10
▃	G + SHIFT + 6	131	▜	G + SHIFT + D	9
▂	G + SHIFT + 7	3	▛	G + SHIFT + F	138
▐	G + SHIFT + 8	133	▝	G + SHIFT + G	137
▜	G + SHIFT + Q	129	▒	G + SHIFT + H	136
▛	G + SHIFT + W	130		SPACE	0
▟	G + SHIFT + E	7	▀	G + SPACE	128

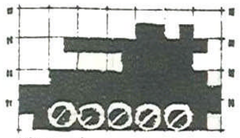
Tabela 4 — Relação dos símbolos gráficos e seus códigos

Estes símbolos são tratados pelo computador como caracteres STRING, tal qual uma letra ou número.

Exemplos:

```
 50 REM >>EXEMPLO 1<<
100 FOR I=1 TO 704
200 PRINT "■";
300 NEXT I
301 PAUSE 600
302 CLS
310 REM
320 REM >>EXEMPLO 2<<
330 REM
500 PRINT "
501 PRINT "
502 PRINT "
503 PAUSE 600
504 CLS
510 REM
520 REM >>EXEMPLO 3<<
530 REM
600 DIM A$(4,7)
610 REM
620 PRINT
621 PRINT
622 PRINT
623 PRINT
630 FOR I=1 TO 4
640 PRINT TAB 7;A$(I)
650 NEXT I
```





9.2 COMANDOS PLOT E UNPLOT

A tela é dividida em 22 linhas e 32 colunas, permitindo a exibição simultânea de até 704 caracteres. A tela é assim formada por 704 pequenos quadrados que, por sua vez, podem ser divididos em quatro partes, chamadas *pixel*. A tela, assim, pode conter até 2.816 pixéis.

Para a definição da posição exata de um pixel (que é um quadradinho preto), a tela é divida em 44 linhas (numeradas de 0 a 43, de *baixo para cima*), por 64 colunas (numeradas de 0 a 63, da esquerda para a direita). Note que a numeração das linhas inicia *embaixo,* e não em cima, como no comando PRINT.

O pixel ocupa o espaço de 1/2 linha por 1/2 coluna.

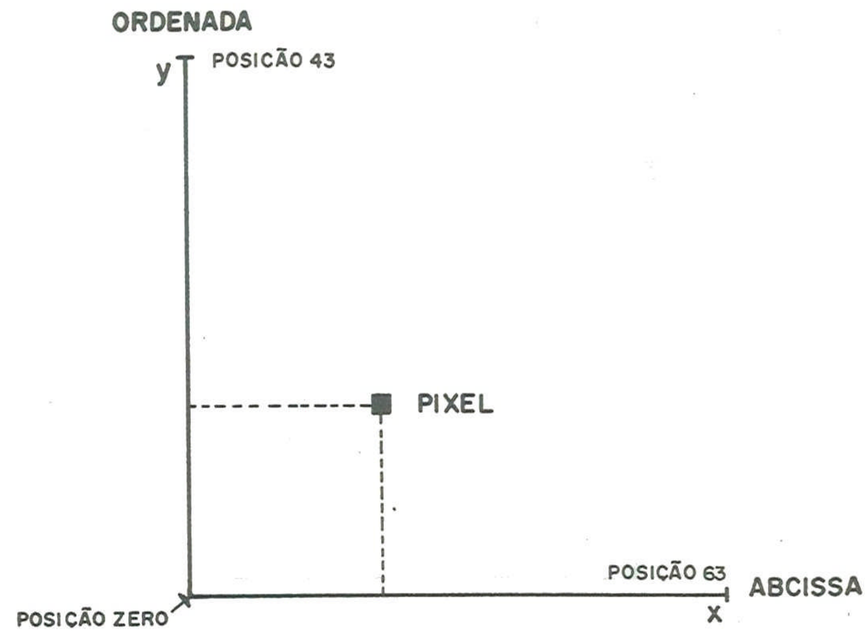


Figura 3 — Posicionamento do pixel

A colocação de um pixel numa determinada posição é feita pela instrução *PLOT,* enquanto que a instrução *UNPLOT* retira (apaga) o pixel.

PLOT
UNPLOT

A colocação e retirada de pixéis baseia-se no sistema de coordenadas cartesianas. Este sistema é composto por dois eixos; o horizontal, chamado abscissa (eixo x), e o vertical, chamado ordenada (eixo y). A forma geral destas instruções é:

PLOT < EXP.ARIT. > , < EXP.ARIT. >
ABSCISSA ORDENADA

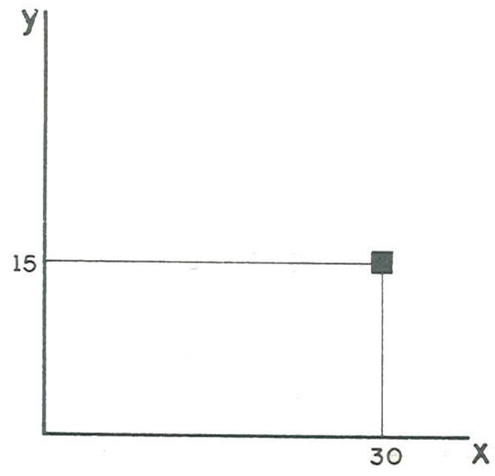
UNPLOT < EXP.ARIT. > , < EXP.ARIT. >
ABSCISSA ORDENADA

A instrução

PLOT 30,15

coloca um pixel na interseção da linha 15 com a coluna 30 (abscissa = 30

e ordenada = 15, ou X = 30 e Y = 15).



Outros exemplos válidos:

PLOT X + 2, Z-2
PLOT I, 63
UNPLOT I, 63
IF A = L THEN PLOT L, 45

O seguinte exemplo constrói uma moldura na tela.

```
10  FOR J = Ø TO 63
20      PLOT Ø, J
30      PLOT 43, J
40  NEXT J
50  FOR J = 1 TO 42
60      PLOT I, Ø
70      PLOT I, 63
80  NEXT J
```

9.3 COMANDOS CODE E CHR$

Todos os caracteres da linguagem BASIC possuem um código numérico, que varia entre 0 e 255. A relação completa destes códigos pode ser encontrada no manual do computador. As tabelas 4 e 6 relacionam os códigos dos caracteres mais usados.

CARACT	CÓDIGO	CARACT	CÓDIGO	CARACT	CÓDIGO	CARACT	CÓDIGO
ESPACO	0	3	31	E	42	D	53
$	13	4	32	F	43	Q	54
+	21	5	33	G	44	R	55
–	22	6	34	H	45	S	56
*	23	7	35	I	46	T	57
/	24	8	36	J	47	U	58
,	26	9	37	K	48	V	59
.	27	A	38	L	49	W	60
0	28	B	39	M	50	X	61
1	29	C	40	N	51	Y	62
2	30	D	41	O	52	Z	63

Tabela 2. Extrato dos caracteres e seus respectivos códigos

A ligação entre um caractere e seu código e vice-versa é feita através das instruções *CODE* e *CHR$*.

CODE

A instrução *CODE* é aplicada a uma variável STRING e fornece o código do primeiro caractere da variável STRING (se a variável tiver mais de um caractere). O retorno é zero, se a variável STRING for vazia (sem caracteres).

Exemplos:

```
CODE "$"      →   13
CODE "ADAO"   →   38
```

Possibilidades:

```
25Ø  IF CODE A$ = 28 THEN GOTO 5ØØ
 8Ø  LET VALOR = CODE B$
```

CHR$

A instrução *CHR$* tem função inversa; se aplicada a um número, fornece o caractere correspondente.

Exemplos:

```
CHR$  13  → $
CHR$  5Ø  → M
CHR$ 128  → ■
```

Esse programa exibe os 256 caracteres disponíveis e seus respectivos códigos:

```
1Ø  FOR I = Ø TO 256 STEP 2
2Ø  SCROLL
3Ø  PRINT I; " = "; CHR$ I, I + 1; " = ";
    CHR$ (I + 1)
```

```
40  FOR J = 1 TO 10
50  NEXT J
60  NEXT I
```

O processamento e a digitação, em certas ocasiões, tornam-se mais rápidos com o uso da instrução *CHR$*, já que

PRINT " " = PRINT CHR$ 134

9.4 FUNÇÃO *INKEY$*

As funções *INKEY$* e *INPUT* têm objetivos idênticos: são utilizadas para fornecer uma informação ao computador (entrada de dados). A diferença mais visível reside no fato de no uso de *INKEY$* não haver necessidade do uso de *NEW LINE*.

Na realidade, a função *INKEY$* lê todo o teclado do computador numa velocidade muito rápida, e verifica se alguma tecla está sendo pressionada. Se uma tecla estiver sendo pressionada, a função *INKEY$* assume o caractere da tecla pressionada. Assim, se no instante que se processar a declaração

```
30  INKEY$
```

a tecla *A* estiver sendo pressionada, o programa assume

INKEY$ = "A"

Uma alternativa seria

```
30  LET A$ = INKEY$
```

onde a variável *A$* assume o caractere da tecla pressionada.

Se nenhuma tecla for pressionada no instante em que uma função *INKEY$* ocorre, o programa assume uma STRING vazia " ".

A função *INKEY$* não espera a teclagem de um caractere, enquanto que a função *INPUT* espera, por necessitar do *NEW LINE*.

A função *INKEY$* somente permite a entrada de um caractere STRING (note que a função tem um $ no seu nome), e de um caractere de cada vez. Já o *INPUT* permite entrar com dados numéricos ou vários caracteres ao mesmo tempo.

Esse programa simula uma máquina de escrever, exibindo na tela a tecla pressionada:

```
10  IF INKEY$ <>  ""  THEN GOTO 10
20  IF INKEY$ =   ""  THEN GOTO 20
30  PRINT INKEY$
40  GOTO 10
```

Enquanto a linha 10 aguarda a retirada da pressão sobre uma tecla, a linha 20 "aguarda" pela pressão sobre uma tecla.

Esta função tem sua utilidade na elaboração de jogos, onde a rapidez da entrada de dados é de fundamental importância.

Vamos considerar um programa que simule o movimento de um submarino. Se a direção do seu movimento for indicada pelas teclas 5, 6, 7 e 8 e o disparo de um suposto torpedo, através da tecla F (fogo), o trecho do programa correspondente poderia ser:

```
  3 LET S$=CHR$ 128
  5 LET B$=" "
 10 LET L=1
 20 LET C=15
 30 PRINT AT L,C;S$
 80 LET LA=L
 90 LET CA=C
100 LET L=L+(INKEY$="6" AND L<2
2)-(INKEY$="7" AND L>=0)
110 LET C=C+(INKEY$="8" AND C>=
0)-(INKEY$="5" AND C<32)
115 IF INKEY$="P" THEN STOP
120 IF INKEY$="F" THEN GOSUB 10
00
130 IF LA<>L OR CA<>C THEN PRIN
T AT LA,CA;B$
140 PRINT AT L,C;S$
150 GOTO 80
999 REM ---SUBROTINA DISPARO---

1000 IF C>20 THEN GOTO 1050
1010 FOR I=C+1 TO 31
1020 PRINT AT L,I;"-"
1025 PRINT AT L,I;B$
1030 NEXT I
1040 RETURN
1050 FOR I=C-1 TO 0 STEP -1
1060 PRINT AT L,I;"-"
1065 PRINT AT L,I;B$
1070 NEXT I
1080 RETURN
```

A variável L representa o número da linha e C o número da coluna. A linha 100 incrementa de uma unidade a variável L, desde que a tecla 6 esteja pressionada "e" desde que o seu valor seja inferior a 22, ou

decrementa de uma unidade desde que a tecla 7 esteja pressionada e L seja positivo. A linha 110 faz a mesma coisa, mas com relação à variável C, que representa as colunas. Se nenhuma tecla for pressionada, o trecho do programa entre as linhas 80 e 150 "roda" indefinidamente.

Com a teclagem de *F*, a condição da linha 120 transfere o processamento para a linha 1000, retornando, após, à linha 130. A teclagem de *P* interrompe o processamento.

9.5 EXERCÍCIOS

30) Considere o problema da trajetória de uma pedra jogada para cima num ângulo *TETA*, em relação à horizontal, e com velocidade inicial *V* (desprezando a resistência do ar). As equações seguintes descrevem as coordenadas da pedra, *X* e *Y*, como funções do tempo *T*.

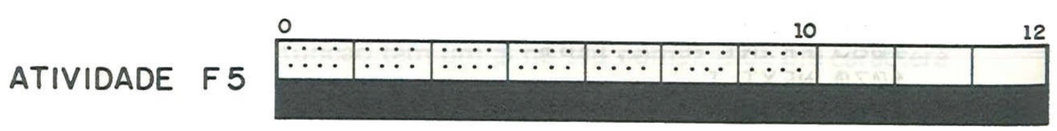
```
X = V * T * COS(TETA)
Y = V * T * SIN(TETA) – 1/2 * G * T**2
G = 9.8      METROS/SEGUNDO**2
```

Elabore um programa que pergunte o ângulo e a velocidade inicial, e trace, via função *PLOT*, a trajetória da pedra. Considere o tempo *T* variando de 0 a 10 segundos, com intervalos de 0.25 seg., a variação do tempo deve ser interrompida quando *Y* (valor da ordenada) for maior que 40 ou menor que zero; e *X* (valor da abscissa) for superior a 60.

31) Escreva um programa que gere, aleatoriamente, 704 símbolos gráficos, e os exiba na tela.

32) Altere o programa anterior, para que o programa, via *INKEY$*, leia um conjunto de caracteres e encha a tela com os mesmos.

33) Um cronograma de barras horizontais pode representar graficamente o tempo planejado e realizado das atividades de um projeto. Por exemplo



a barra pontilhada indica uma duração planejada de 10 dias para a atividade *F5*, enquanto que a barra cheia indica uma duração efetiva de 12 dias.

Elabore um programa que exiba na tela os tempos planejados e realizados de um conjunto de atividades de um projeto.

34) Uma empresa apresentou nos últimos anos o seguinte faturamento, em milhões de cruzeiros,

ANO	1969	1970	1971	1972	X
CR$	1.5	3.2	7.0	5.4	Y

Este comportamento pode ser representado gráfica e algebricamente através de uma reta de regressão linear (de mínimo quadrado). A equação linear é:

```
Y = A + B * X
```

onde

```
B = (SXY – SX*SY/N) / (SX2 – SX**2/N)
A = SY/N – B*SX/N
```

e

```
  N  =  número de pares de valores
 SX  =  somatória dos X
 SY  =  somatória dos Y
SXY  =  somatória dos produtos X*Y
SX2  =  somatória de X**2
```

Elabore um programa que, dado um conjunto de pares de valores *X* e *Y*, calcule a reta de regressão linear, exiba na tela o sistema de eixos cartesianos, os pontos correspondentes aos pares de valores e a reta de regressão linear.

9.6 DESAFIO

Analise, procure entender, "rode" e tente explicar os seguintes programas:

```
   1 REM )))PROGRAMA 25(((
   6 LET T$="DIGITE <NEW LINE>"
  15 PRINT "     ""CORRIDA DE FO
RMULA-1"""
  16 PRINT
  17 PRINT "A CADA INSTANTE, DIG
ITE","O ACRESCIMO DE VELOCIDADE"
,"(+ OU -) E A MARCHA (1 A 5)","
PARA O PROXIMO TRECHO.",,,,,"EX:
```

```
 302= +30 KM/H NA MARCHA 2","
-105= -10 KM/H NA MARCHA 5"
  21 PRINT AT 20,10;T$
  22 INPUT G$
  23 CLS
  24 PRINT """ATENCAO"", PARA OS
 LIMITES:"
  25 PRINT "MARCHA V.MAX. ACELER
."
  26 PRINT "  1      90     +50/-5
0"
  27 PRINT "  2     180     +40/-4
0"
  28 PRINT "  3     270     +30/-3
0"
  29 PRINT "  4     360     +20/-2
0"
  30 PRINT "  5     450     +10/-1
0"
  31 PRINT
  33 PRINT "RPM MAX. DO MOTOR =
9000"
  34 PRINT "COMBUSTIVEL (LTS) =
500"
  35 PRINT AT 20,10;T$
  36 INPUT G$
  37 CLS
  38 PRINT "NAS CURVAS A V.MAX.
FICA REDUZI-DA EM 40 P/CENTO"
  39 PRINT
  40 PRINT "VELOCIDADE PROXIMA D
O LIMITE    CAUSA DERRAPAGEM E A
CRESCIMO NO TEMPO. (MULTA)"
  41 PRINT
  42 PRINT "PISTA COM ""OLEO"" V
.MAX. REDUZIDAEM 40 P/CENTO"
  43 PRINT
  44 PRINT "PISTA ""UMIDA"", V.M
AX. REDUZIDA  EM 10 P/CENTO"
  45 PRINT
  46 PRINT "PISTA ""MOLHADA"", V
.MAX. REDUZIDAEM 20 P/CENTO"
  47 PRINT AT 20,10;T$
  48 INPUT G$
  49 CLS
 100 LET Z$="012201200118011601 1
401120110010801060104010203020502
020504050605080510051207120912091
009080906090409021102130213041306
130813101312131412151015081506 1
504150417041906190819101912191320
132212231023082306230423022 3"
```



```
 600 DIM C$(22,32)
 610 LET C$(1)="111111111111111
111SAIDA"
 620 LET C$(2)="1124444444444444
4444444431C.GER"
 630 LET C$(3)="1132444444444444
4444444331PISTA"
 640 LET C$(4)="113311111111111
CHEGADA331-----"
 650 LET C$(5)="113311111111112
444431133"
 660 LET C$(6)="1134444444443313
244331133"
 670 LET C$(7)="1144444444433313
311331133"
 680 LET C$(8)="1111111111133313
311331133"
 690 LET C$(9)="1111111111133313
311331133"
 700 LET C$(10)="112444444445531
3311331133"
 710 LET C$(11)="113244444444451
3311331133".
 720 LET C$(12)="113311111111111
3311331133"
 730 LET C$(13)="113311111111111
3311331133PISTA:"
 740 LET C$(14)="113444444444444
5311344453"
 750 LET C$(15)="114444444444444
4511444445"
 770 LET C$(17)=" VELOC:     K/H
ROTAC:      RPM"
 780 LET C$(18)=" V.MED:     K/H
MARCH:"
 790 LET C$(19)=" TEMPO:     MIN
GASOL:      LTS"
 810 LET C$(21)="DIGITE O ACRESC
IMO DE VELOCIDADE"
 820 LET C$(22)="E A MARCHA P/PR
OXIMO TRECHO."
1000 FOR I=1 TO 22
1100 PRINT C$(I)
1200 NEXT I
1400 REM SORTEIO COND.PISTA
1405 LET VMAX=450
1406 LET D=0
1407 LET POS=0
1408 LET P=0
1409 LET B=10
1410 LET A=INT (RND*6+1)
1411 LET GAS=500
```

ATENÇÃO:
SUBSTITUIR OS CARACTERES 1,2,3,4 e 5 DAS LINHAS 610 a 750 POR:

1→ BRANCO
2→ 
3→ 
4→ 
5→ 

```
1420 IF A<=4 THEN PRINT AT 4,26;
"SECA"
1430 IF A>4 AND A<=7 THEN PRINT
AT 4,26;"UMIDA"
1433 IF A>7 THEN PRINT AT 4,25;"
MOLHADA"
1435 IF A>4 AND A<=7 THEN LET VM
AX=VMAX*.9
1436 LET MA1=0
1440 IF A>7 THEN LET VMAX=VMAX*.
8
1444 LET O=0
1445 LET POS=1
1446 PRINT AT 1,22;CHR$ 136
1447 LET AVE=0
1448 LET VELOC=AVE
1450 LET ST=POS
1465 LET MA1=0
1469 REM INPUT VELOC+MARCHA
1470 INPUT VEMA
1475 LET POS=POS+1
1476 IF POS=P THEN LET VMAX=VMAX
*.6
1478 IF POS=11 OR POS=13 OR POS=
18 OR POS=20 OR POS=25 OR POS=27
 OR POS=33 OR POS=38 OR POS=40 O
R POS=44 OR POS=46 THEN GOSUB 54
00
1480 LET VE=INT (VEMA/10)+(VEMA(
0)
1490 LET MA=ABS (VEMA)-INT (ABS
(VEMA)/10)*10
1491 LET G=ABS (VE)
1492 IF MA=1 AND G>50 THEN GOSUB
 6000
1493 IF MA=2 AND G>40 THEN GOSUB
 6000
1494 IF MA=3 AND G>30 THEN GOSUB
 6000
1495 IF MA=4 AND G>20 THEN GOSUB
 6000
1496 IF MA=5 AND G>10 THEN GOSUB
 6000
1500 IF MA=0 OR MA>5 THEN GOTO 6
00
1510 IF MA1=0 AND MA>1 THEN GOTO
 1525
1520 GOTO 1630
1525 FOR J=1 TO 5
1530 PRINT AT 7,25;"COMO EH"
1540 PRINT AT 8,25;"SAIR"
1550 PRINT AT 9,25;"EM ";MA;" ?"
```



```
1570 FOR I=7 TO 9
1580 PRINT AT I,25;"
1590 NEXT I
1600 NEXT J
1605 LET POS=POS-1
1610 GOTO 1465
1630 LET AVE=AVE+VE
1631 LET MA1=MA
1632 LET O=O+AVE
1635 LET VELOC=VELOC+VE
1636 IF VELOC<0 THEN LET VELOC=0

1637 IF VELOC>VMAX THEN GOTO 530
0
1638 IF B<=5 AND VELOC>VMAX*.6 T
HEN GOSUB 5450
1639 IF D=1 AND VELOC>VMAX*.8 TH
EN GOSUB 5450
1640 LET VEME=O/(POS-1)
1642 LET MA1=1
1650 LET ROTA=VELOC/MA*100
1660 IF ROTA>9000 THEN GOTO   50

1670 LET TEMPO=(POS-1)/VELOC*10
1680 LET ST=ST+TEMPO
1700 IF VE>=0 THEN LET GAS=GAS-R
OTA/250
1710 IF GAS<=0 THEN GOTO 5100
1740 LET I=POS*4-3
1745 LET LI=INT ((VAL Z$(I TO I+
3))/100)
1748 LET CO=VAL Z$(I TO I+3)-INT
 ((VAL Z$(I TO I+3))/100)*100
1760 PRINT AT LI,CO;CHR$ 136
1768 PRINT AT 16,7;"   ";AT 16,7
;VELOC
1775 IF POS>2 AND VELOC<80 THEN
GOSUB 5000
1780 PRINT AT 17,7;"   ";AT 17,7
;INT (VEME)
1790 PRINT AT 18,7;"   ";AT 18,7
;INT (ST+.5)
1800 PRINT AT 16,22;"    ";AT 16
,22;INT (ROTA)
1810 PRINT AT 17,22;MA
1820 PRINT AT 18,22;"   ";AT 18,
22;INT (GAS)
1821 IF POS=52 THEN GOTO 8000
1822 IF POS=P THEN LET VMAX=VMAX
/.6
1825 IF POS=10 OR POS=12 OR POS=
17 OR POS=19 OR POS=24 OR POS=26
 OR POS=32 OR POS=37 OR POS=39 O
```

```
R POS=43 OR POS=45 THEN GOSUB 52
00
1826 IF D=1 THEN LET VMAX=VMAX/.
3
1827 IF D=1 THEN LET D=0
1840 REM SORTEIO COND/PISTA
1850 LET B=INT (RND*100)
1860 IF B<=5 THEN PRINT AT 14,26
;"OLEO"
1865 IF B<=5 THEN LET P=POS+2
1866 IF B>5 THEN LET P=0
1870 IF B>5 THEN PRINT AT 14,26;
"-OK-"
1880 GOTO 1470
5000 FOR E=1 TO 6
5010 PRINT AT 8,26;"TARTA";AT 9,
26;"RUGA"
5040 PRINT AT 8,26;"     ";AT 9,
26;"    "
5050 NEXT E
5060 RETURN
5100 PRINT AT 7,25;"ACABOU";AT 8
,25;"   A  ";AT 9,25;"GASOL.";AT
 10,25;" O FIM"
5110 STOP
5150 PRINT AT 7,25;"FUNDIU";AT 8
,25;"  O   ";AT 9,25;"MOTOR ";AT
 10,25;">> FIM"
5151 FOR E=1 TO 5
5152 FOR W=1 TO 3
5153 PRINT AT 16,22;INT (ROTA)
5154 NEXT W
5155 PRINT AT 16,22;"     "
5156 NEXT E
5160 STOP
5200 FOR E=1 TO 6
5210 PRINT AT 8,25;"CUIDA";AT 9,
27;"DO"
5240 PRINT AT 8,25;"     ";AT 9,
25;"     "
5250 NEXT E
5260 RETURN
5300 FOR E=1 TO 6
5310 PRINT AT 7,25;"EXCESSO";AT
8,25;"  DE   ";AT 9,25;"VELOC. "

5340 PRINT AT 7,25;"       ";AT
8,25;"       ";AT 9,25;"       "

5350 NEXT E
5355 PRINT AT 7,25;"E O FIM"
5356 PRINT AT 8,25;"TE VEJO"
```



```
5357 PRINT AT 9,25;"FUNERAL"
5360 STOP
5400 LET VMAX=VMAX*.3
5410 LET D=1
5420 RETURN
5450 LET ST=ST+.01
5460 FOR E=1 TO 6
5470 PRINT AT 7,25;"DERRA";AT 8,
25;"PAGEM"
5500 PRINT AT 7,25;"     ";AT 8,
25;"     "
5530 NEXT E
5540 RETURN
6000 FOR E=1 TO 6
6010 PRINT AT 7,25;"INFORMA";AT
8,25;"  CAO  ";AT 9,25;"ERRADA "

6040 PRINT AT 7,25;"       ";AT
8,25;"       ";AT 9,25;"       "

6050 NEXT E
6055 LET POS=POS-1
6060 GOTO 1470
8000 LET ES=VEME*GAS/ST
8001 FOR E=1 TO 50
8002 NEXT E
8020 CLS
8030 PRINT "P A R A B E N S"
8040 PRINT
8050 PRINT "COMPLETOU O CIRCUITO
"
8070 PRINT AT 10,0;"TEU ESCORE F
OI DE:"
8080 PRINT
8090 PRINT "     ";INT (ES);" PO
NTOS"
8100 STOP
9000 SAVE "F1"
9010 RUN


   1 REM >>>PROGRAMA 26<<<
   5 LET G=16398
   6 LET H=G+1
  10 FOR I=0 TO 31
  11 PRINT AT 0,I;CHR$ 128;AT 21
,I;CHR$ 128
  20 NEXT I
  30 FOR A=1 TO 20
  35 PRINT AT 2+RND*18,1+RND*27;
"  "
  36 PRINT AT 2+RND*18,1+RND*27;
"  "
```

```
  37 PRINT AT 2+RND*18,1+RND*27;
" "
  42 PRINT AT A,0;CHR$ 128
  43 PRINT AT 3+RND*15,2+RND*22;
CHR$ 128;CHR$ 0;CHR$ 128
  44 PRINT AT 2+RND*18,1+RND*27;
" "
  45 PRINT AT 2+RND*18,2+RND*24;
CHR$ 128
  46 PRINT AT 3+RND*15,2+RND*27;
CHR$ 136
  47 PRINT AT 2+RND*18,2+RND*24;
CHR$ 137;CHR$ 138
  50 PRINT AT A,31;CHR$ 128
  57 PRINT AT 2+RND*16,2+RND*26;
" "
  60 NEXT A
  61 FOR Z=1 TO 13
  62 PRINT AT 20,30;CHR$ 0
  63 PRINT AT 20,30;CHR$ 151
  64 PRINT AT 20,30;CHR$ 151
  65 PRINT AT 20,30;CHR$ 0
  66 PRINT AT 20,30;CHR$ 151
  67 PRINT AT 20,30;CHR$ 0
  68 NEXT Z
  70 LET A=INT (RND*8+1)
  75 LET Q=0
  80 LET B=INT (RND*15+1)
  85 PRINT AT 20,30;CHR$ 0
  90 LET E=A
  95 LET Q=Q+1
 100 LET F=B
 101 IF A=20 AND B=30 THEN GOTO
2000
 105 LET T=0
 106 IF RND>.2478 THEN GOTO 120
 110 LET Y=INT (RND*7)+1
 111 IF Y=1 THEN GOTO 120
 112 IF Y=6 THEN GOTO 169
 113 IF Y=3 THEN GOTO 200
 114 IF Y=4 THEN GOTO 250
 115 IF Y=5 THEN GOTO 290
 116 IF Y=2 THEN GOTO 154
 117 IF Y=7 THEN GOTO 330
 120 PRINT AT A+1,B;
 130 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 140 IF T=1 THEN LET A=A+1
 150 IF T=1 THEN GOTO 1000
 152 IF RND>.2 THEN GOTO 169
 154 IF A=0 OR B=30 THEN GOTO 16
9
```



```
 155 PRINT AT A-1,B+1;
 156 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 157 IF T=1 THEN LET B=B+1
 158 IF T=1 THEN LET A=A-1
 159 IF T=1 THEN GOTO 1000
 165 IF RND<.2 THEN GOTO 110
 169 PRINT AT A,B+1;
 170 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 180 IF T=1 THEN LET B=B+1
 190 IF T=1 THEN GOTO 1000
 195 IF RND<.6 THEN GOTO 290
 200 PRINT AT A+1,B+1;
 210 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 220 IF T=1 THEN LET A=A+1
 230 IF T=1 THEN LET B=B+1
 240 IF T=1 THEN GOTO 1000
 245 IF RND<.1 THEN GOTO 110
 250 PRINT AT A-1,B;
 260 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 270 IF T=1 AND A>0 THEN LET A=A
-1
 280 IF T=1 THEN GOTO 1000
 290 PRINT AT A,B-1;
 300 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 310 IF T=1 AND B>0 THEN LET B=B
-1
 320 IF T=1 THEN GOTO 1000
 330 IF B=0 OR A=0 THEN GOTO 110

 340 PRINT AT A-1,B-1;
 350 IF PEEK (PEEK G+256*PEEK H)
=0 THEN LET T=1
 360 IF T=1 THEN LET A=A-1
 370 IF T=1 THEN LET B=B-1
 380 IF T=1 THEN GOTO 1000
 390 GOTO 110
1000 PRINT AT E,F;CHR$ 0
1010 PRINT AT A,B;CHR$ 151
1020 GOTO 90
2000 PRINT AT 0,15;CHR$ 146;Q;CH
R$ 147
2010 FOR N=1 TO 50
2011 NEXT N
2030 FOR N=15 TO 20
2040 PRINT AT 0,N;CHR$ 128
2050 NEXT N
2060 GOTO 30
```

```
  1 REM >>>PROGRAMA 27<<<
 50 FAST
100 LET A$=CHR$ 128
110 LET B$=" "
120 FOR X=28 TO 63
130 FOR L=1 TO 8
140 LET V=PEEK (7679+L+8*X)
150 LET P$=""
160 LET D=256
170 FOR K=8 TO 1 STEP -1
180 LET D=D/2
190 LET C$=B$
200 IF V<D THEN GOTO 230
210 LET C$=A$
220 LET V=V-D
230 LET P$=P$+C$
240 NEXT K
250 PRINT P$
260 NEXT L
270 PAUSE 60
290 CLS
300 NEXT X
```


10 LINGUAGEM DE MÁQUINA

10.1 GENERALIDADES

O computador pode ser comparado a um ser humano que fala BASIC em vez de português. Tem cérebro, chamado *microprocessador,* e memória. Conversamos com ele através do teclado (para enviar informações) e da tela da TV (para receber informações), usando BASIC. Acontece que o microprocessador não entende BASIC. Ele entende uma linguagem mais simples, chamada *linguagem de máquina.*

É possível, entretanto, conversar diretamente com o microprocessador, só que através de um programa *montador,* chamado *assembly* ou *assembler.*

Esta possibilidade aumenta significativamente a velocidade de processamento.

O acesso direto ao microprocessador é feito através das funções *USR, POKE* e *PEEK,* sendo que os valores numéricos são informados num outro sistema, *binário* ou *hexadecimal* (o nosso é o decimal).

O ensino da linguagem de máquina foge aos objetivos desta obra. Para maiores detalhes, consulte a relação bibliográfica.

10.2 BIT, BYTE, SISTEMAS BINÁRIO E HEXADECIMAL

A rápida evolução da eletrônica tornou possível a criação de circuitos extremamente pequenos e capazes de assumir dois *estados,* representando a presença ou não de tensão elétrica. Estes estados são representados respectivamente pelos algarismos **1 e 0** (um e zero). Um computador pode ter "milhões" destes circuitos, todos interligados.

Cada algarismo 0 ou 1 é chamado de *BIT,* e o conjunto de 8 bits é chamado de *BYTE.*

BIT
BYTE

SISTEMA BINÁRIO			SISTEMA DECIMAL
0000	0000	→	0
0000	0001	→	1
0000	0010	→	2
0000	0011	→	3
....	....		...
1111	1111	→	255

Assim, 1 byte (8 bits) pode representar os números inteiros positivos de 0 a 255. Números maiores de 255, mais bytes, devem ser utilizados.

Alguns micros trabalham com 16 bits (2 bytes), podendo, assim, preservar números de 0 a 65535 numa única posição de memória.

Para transformar um número do sistema binário (base 2) para o sistema decimal (base 10), é suficiente fazer:

```
1000  1010   →  (1*2**7) + (0*2**6) + (0*2**5) + (0*2**4)
                (1*2**3) + (0*2**2) + (1*2**1) + (0*2**0)
```

$(1000\ 1010)_2 \rightarrow (138)_{10}$

Os números hexadecimais foram criados para facilitar a representação de números binários, pela substituição de longas seqüências de zeros e uns, pelos símbolos hexadecimais, que reduzem em quatro vezes uma representação binária.

Como existem somente 10 dígitos, e este sistema tem base 16, as letras do alfabeto são utilizadas para completar a simbologia.

SISTEMA HEXADECIMAL		SISTEMA DECIMAL
0	→	0
1	→	1
2	→	2
3	→	3
4	→	4
...		...
9	→	9
A	→	10
B	→	11
...		...
F	→	16
...		...
FF	→	255

10.3 FUNÇÃO *USR*

Ao receber uma instrução *RUN* ou *GOTO N,* o computador passa a se utilizar do interpretador BASIC para "se instruir sobre o que fazer". E, assim, prossegue linha após linha até o final do programa.

Caso encontre uma instrução *USR,* ele "esquece" a BASIC e passa a trabalhar em linguagem de máquina, dirigindo-se ao byte indicado junto ao *USR* (*USR 16514,* por exemplo). E, assim, prossegue até encontrar o código 201. Este código corresponde à instrução *RET* (no programa aparece *TAN*) e indica ao computador que retorne a trabalhar via BASIC, voltando à linha seguinte da que continha a instrução *USR.* **USR** **RET**

Assim, a instrução *USR,* lida em BASIC, faz o computador trabalhar em linguagem de máquina; e a instrução *RET,* lida em linguagem de máquina, o traz de volta ao BASIC.

Exemplo:

```
1  REM E$ RND7 — : 5? C L EN ?74 POKE (TAN
2  . . . . . .
   . . . . . .
5Ø  LET DE = USR 16514
6Ø  . . . . . .
```

A função USR da linha 50 ordena ao computador que se dirija, já em linguagem de máquina, ao byte de número 16514, onde encontra-se o código do primeiro caractere após o *REM* da linha 1 (código do caractere E = 42). A linha 1 funciona, assim, como uma sub-rotina em linguagem de máquina. O "conteúdo" do *REM* de uma linha 1, o computador sempre coloca a partir do byte número 16514.

A memória é divida em duas grandes áreas: a *ROM* (read only memory) contém o programa interpretador BASIC que "traduz" as instruções da BASIC para a linguagem de máquina. Esta região da memória é inacessível (pelo usuário); somente pode ser lida e não perde jamais seu conteúdo (mesmo desligando-se o computador). Logo a seguir vem a *RAM* (random access memory), que pode ser lida, escrita e alterada, e onde ficam armazenados o programa BASIC elaborado por nós, o conteúdo da tela da TV, as variáveis do programa e as rotinas em linguagem de máquina (também por nós elaboradas). Esta parte da memória perde seu conteúdo quando o computador é desligado. **ROM** **RAM**

ROM →	PROGRAMA INTERPRETADOR			
INICIO RAM →	variáveis do sistema	DFILE, CODR RAMTOP, ETC	← 16384	
		10 FOR I = 0 . . .	← 16509	
	programa	20 PRINT 30		
DFILE →	arquivos da tela	RESULTADOS: – – – – – – – MEDIA: 5.45	← PEEK 16396 + 256*PEEK 16397	
VARS →	variáveis	LET B = 15 LET A$ = "$" DIM C$ (45,2)	← PEEK 16400 + 256*PEEK 16401	
ELINE →	área de trabalho + linha digitada	100 LET A = . . .	← PEEK 16404 + 256*PEEK 16405	
PILFUN →	pilhas para cálculos	LET B = (V-3)	← PEEK 16410 + 256*PEEK 16411	
PILFIM →	área de memória		← PEEK 16412 + 256*PEEK 16413	
	pilha p/trabalho do microprocessador	Z80		
SP →	pilha p/endereçamento de sub-rotina	50 GOSUB 200 90 RETURN	← PEEK 16386 + 256*PEEK 16387	
RAMTOP →	área que pode ser reservada p/rotinas em linguagem de máquina	USR	← PEEK 16388 + 256*PEEK 16389	

Figura 4 — Distribuição da memória

10.4 FUNÇÕES *POKE* E *PEEK*

A função *POKE* serve para armazenar o código binário correspondente a um valor decimal num determinado "endereço" (byte). Sua forma geral é:

POKE < Nº DO BYTE >, < VALOR DECIMAL > **POKE**

Exemplo:

POKE 16522, 126

coloca o número decimal 126 no byte número 16522.

A função *PEEK* faz o contrário. Traz o conteúdo binário de determinada posição da memória para a forma decimal:

PEEK < Nº DO BYTE > **PEEK**

Exemplo:

PEEK 16522

traz ao programa o conteúdo do byte número 16522.

Possibilidades:

```
1Ø  PEEK (Z-125)
3Ø  LET B = PEEK (16396) + PEEK (16397)*256 – 1
8Ø  PRINT I, CHR$ (PEEK I)
6Ø  POKE X + 3, 213
47  POKE 16732, CODE ("B")
```

O maior valor decimal passível de ser colocado numa posição de memória é 255. Para colocar um número superior, devemos "quebrá-lo" em duas ou mais partes, usando sempre a base 256 (16**2).

Se o número for menor que 256, não precisa ser dividido. Se for entre 256 e 65536 (256**2), o mesmo deve ser "quebrado" em dois; e se for maior que 65536 e menor que 16777216 (256**3), deve ser "quebrado" em três, e assim sucessivamente.

Exemplo:

O número 522, por ser maior que 256**1 e menor que 256**2, deve ser colocado em dois bytes da memória:

```
522 – 256 = 266   (primeira subtração)
266 – 256 = 1Ø    (segunda subtração, e
                   resto 10)
```

Se os bytes forem os de números 30000 e 30001, as instruções serão:

PCKE 30000, 1Ø
POKE 30001, 2

O retorno deste valor seria possível através de:

PRINT PEEK 3ØØØØ + 256 * PEEK 3ØØØ1

10.5 ALGUMAS APLICAÇÕES SIMPLES

a) Para conhecer o número de bytes ocupados pelo programa:

PRINT PEEK 15396 + 256*PEEK 16397 − 16509

pelo programa, pelas variáveis e a tela:

PRINT PEEK 16404 + 256*PEEK 16405 − 16384

ainda disponível:

```
PRINT  (PEEK 16386 + 256*PEEK 16387)  −
       (PEEK 16412 + 256*PEEK 16413)  + 87
```

b) Para a introdução de uma senha (password) num programa BASIC, é suficiente incluir as seguintes linhas

```
  1Ø PRINT "VOCE TEM 1Ø SEGUNDOS
P/INFORMAR A SENHA:"
  11 POKE 16437,2
  12 INPUT S$
  13 IF S$<>"SENHA" THEN GOTO 12

  14 POKE 16437,255
  15 PRINT
  16 PRINT "OK... SENHA CORRETA"

  17 STOP
9998 SAVE "NOME DO PROGRAMA"
9999 RUN
```

c) Esta sub-rotina preserva o conteúdo de uma tela (que sempre se encontra na *RAM* a partir do byte número 16550), num arranjo

string. Para a reprodução da tela, basta pedir a exibição do conteúdo do arranjo.

```
9000 FAST
9001 DIM Z$(704)
9002 FOR Q=Ø TO 21
9003 FOR W=1 TO 32
9004 LET Z$(W+32*Q)=CHR$ PEEK ((
PEEK 16396+256*PEEK 16397)+W+33*
Q)
9005 NEXT W
9006 NEXT Q
9007 SLOW
9008 RETURN
```

Deve-se ter o cuidado de "chamar" a sub-rotina antes de um *CLS.* Para armazenar várias telas, basta definir novos arranjos *STRING.* Outra vantagem desta sub-rotina é a possibilidade de se arquivar várias telas em fita K7. O comando *SA VE* preserva o programa e suas variáveis, que incluem as telas. Para se recuperar a tela, basta digitar *PRINT Z$* (p/exemplo acima).

d) Este programa lê os bytes da ROM

```
 100 LET N=1
 110 PRINT "ENDERECO","BYTE"
 120 PRINT
 130 LET X=15*(N-1)
 140 FOR B=X TO 15*N
 150 PRINT B,PEEK B
 160 NEXT B
 170 PRINT
 180 PRINT "DIGITE <NEW LINE> P/
CONTINUAR"
 190 INPUT C$
 200 CLS
 210 LET N=N+1
 220 IF B>8190 THEN STOP
 230 GOTO 110
```

e) O computador usa as linhas 0 a 21 para compor a tela e reserva para si as linhas 22 e 23, para entrada de dados e códigos de reportagem.

Esta reserva, que é permanente e já vem "embutida" no computador, é feita via instrução

POKE 16418, 2

onde o byte 16418 corresponde a uma variável chamada *DF-SZ*

e o valor numérico 2, ao código de reserva.

Para abrir esta reserva e acessar as duas últimas linhas, basta fazer

POKE 16418,Ø

como no exemplo

```
1Ø  POKE 16418,Ø
2Ø  PRINT AT 22,Ø; "TESTE"; AT 23,Ø; "TESTE"
3Ø  POKE 16418, 2
```

Este "macete" só pode ser utilizado para efeitos de tela e, antes do próximo *INPUT*, a reserva deve ser restabelecida (linha 30).

f) A linguagem BASIC de alguns microcomputadores possui um conjunto de comandos *DATA, READ, RESTORE* que inexistem na BASIC da linha SINCLAIR. Este conjunto de comandos permite a leitura de dados dentro do próprio programa e sem interrupção do processamento. Esta possibilidade, entretanto, também existe na linha SINCLAIR. Uma das opções é a já indicada no item 8.3, o uso de uma variável string para "armazenar" os dados. Existe uma segunda opção, que veremos agora.

Olhando a Fig. 7 (item 10.3), podemos ver que o programa tem início no byte número 16509 da memória. Para a instrução *REM,* o micro utiliza 2 bytes para registrar o número da linha, 2 bytes para o tamanho da linha, 1 byte para a instrução *REM*, mais tantos bytes quanto o número de caracteres digitados após o *REM,* e mais um byte para indicar o final da linha (NEW LINE). Como a instrução *REM* é ignorada no processamento, mas utiliza memória, podemos usar esta instrução para "guardar" números (dados). Tantos quanto quisermos. Esta instrução *REM* deve estar na primeira linha do programa, e, por isso, começa na posição de memória 16514 (16509 + 5 bytes). Veja o seguinte exemplo:

```
  10 REM 10,20,35,76,3456
  20 LET BYTE=16514
  30 FOR I=1 TO 5
  40 GOSUB 500
  50 PRINT I;") SQR(";VAL Z$;")=
";SQR (VAL Z$)
  60 NEXT I
  70 STOP
 500 LET Z$=""
 510 LET Z$=Z$+CHR$ PEEK BYTE
 520 LET BYTE=BYTE+1
 530 IF PEEK BYTE<>26 AND PEEK B
YTE<>118 THEN GOTO 510
 540 LET BYTE=BYTE+(6 AND PEEK B
YTE=118)+(1 AND PEEK BYTE=26)
 550 RETURN
```


APÊNDICE – A

RELAÇÃO DOS COMANDOS E FUNÇÕES DO INTERPRETADOR BASIC

ABS (X)	→	Fornece o valor absoluto de X
ACS (X)	→	Fornece o arco co-seno de X, em radianos
AND	→	Operador lógico (e)
ASN (X)	→	Fornece o arco seno de X, em radianos
ATN (X)	→	Fornece o arco tangente de X, em radianos
BREAK	→	Detém a execução do programa (aplicação externa)
CHR$ (X)	→	Fornece o caractere correspondente ao código X
CLEAR	→	"Apaga", zera todas as variáveis existentes na memória (não "apaga" o programa)
CLS	→	Limpa a tela do vídeo
CODE (A$)	→	Fornece o código do primeiro caractere do arranjo string A$
CONT	→	Continua a execução do programa, após uma interrupção (aplicação externa)
COPY	→	Transfere para a impressora o conteúdo de uma tela
COS (X)	→	Fornece o co-seno de X, em radianos
DIM	→	Reserva área na memória para um arranjo uni ou multidimensional, numérico ou alfanumérico
EXP (X)	→	Eleva o número "e" (e = 2,71828) ao expoente X
FAST	→	Processamento em velocidade mais rápida (somente exibe dados na tela, quando encontrar PAUSE, INPUT ou STOP)
FOR . . TO . . STEP	→	Define um "loop" de programa
GOSUB N	→	Transfere o processamento a uma sub-rotina que tem início na linha N
GOTO N	→	Transfere o processamento para a linha N (aplicação interna), ou inicia o processamento na linha N (aplicação externa)
IF . . . THEN	→	Testa uma condição, se for verdadeira toma uma ação (se . . . então)
INKEY$	→	Entrada de um caractere sem interrupção do processamento
INPUT	→	Entrada de dados via teclado

INT (X) → Fornece o valor inteiro e truncado de X

LEN (A$) → Fornece o "comprimento" (número de caracteres) da variável STRING A$

LET → Atribui um valor a uma variável

LIST → Exibe na tela as 22 primeiras linhas do programa

LIST N → Exibe na tela 22 linhas do programa, a partir da linha N

LLIST → Lista o programa na impressora (imprime)

LN (X) → Fornece o logaritmo natural de X

LOAD "A$" → Procura o programa A$ numa fita K7, e o transfere para a memória do computador

LPRINT → Imprime determinada informação (na impressora)

NEW → Elimina ("apaga") o programa da memória

NEXT → Última instrução do "LOOP" iniciado em FOR

NOT → Operador lógico (não)

OR → Operador lógico (ou)

PAUSE X → Retém a imagem na tela da TV por X/60 segundos

PEEK (X) → Fornece o conteúdo do byte X (da memória)

PI → Número PI, PI = 3,141592653

PLOT X, Y → Exibe um pixel (quadradinho preto) nas coordenadas X, Y

POKE X, Y → Preserva o valor Y no byte X (da memória)

PRINT → Exibe informações na tela da TV

PRINT AT X, Y; → Exibe uma informação a partir da interseção da linha X com a coluna Y

PRINT TAB X; → Exibe uma informação a partir da coluna X (se mantém na linha anterior)

RAND → Define a "semente" (início) de uma série de números aleatórios

REM → Define uma linha de interesse exclusivo do usuário, (desconsiderado pelo programa no processamento); também utilizado para definição de instruções em linguagem de máquina

RETURN → Último comando de uma sub-rotina iniciada num GOSUB, e transfere o processamento para a linha seguinte ao GOSUB

RND → Gerador de números aleatórios entre 0,0000000 e 0,99999999

RUN → Executa o programa, após "apagar" todas as variáveis porventura existentes na memória

SAVE A$ → Preserva com o nome A$, em fita K7, o programa residente na memória do computador

SCROLL → "Rola" a imagem da tela uma linha para cima

SGN (X) → Retorna o valor – 1, 0, ou 1 se o sinal de X for negativo, nulo ou positivo

SIN (X) → Fornece o seno de X, em radianos

SLOW → Velocidade de processamento mais lenta, exibindo continuamente as informações na tela

SQR (X) → Fornece a raiz quadrada de X

STOP → Detém a execução do programa

STR$ (X) → Converte o número X num caractere string X

TAN (X) → Fornece a tangente de X, em radianos

UNPLOT X, Y → Retira o pixel das coordenadas X, Y

USR → Transfere o processamento a uma sub-rotina em linguagem de máquina

VAL (X) → Converte o caractere string "X" em valor numérico X

APÊNDICE – B

CÓDIGOS DE REPORTAGEM

a) Indicação de erro:

Código	Significado	Corretivo Possível
0	Execução Bem-sucedida	–
1	*NEXT* sem o *FOR* correspondente	*Incluir o FOR* faltante, ou retirar o *NEXT* excedente
2	Nome de variável não existente (variável indefinida)	Definir a variável em linha anterior
3	Subscrito negativo ou superior ao valor informado no *DIM*	Dar *PRINT* no subscrito p/conhecer seu valor, e aumentar o limite no *DIM*, se necessário
4	A memória disponível é insuficiente p/continuar o processamento	Otimizar o programa; dar *CLEAR*, se possível; deletar as linhas já executadas
5	Faltando espaço na tela para exibir todas as informações	*CONT* cria a área necessária; modificar o programa; o uso do *SCROLL* elimina o problema
6	A expressão aritmética da linha indicada resultou num valor superior a 10**38	Executar a expressão passo a passo (via *PRINT*), p/verificar onde está o erro
7	*RETURN*, sem o *GOSUB* correspondente	Incluir o *GOSUB* faltante, ou retirar o *RETURN* excedente
8	*INPUT* inaceitável	Liste a linha indicada p/verificar o que está sendo pedido
9	Processamento interrompido devido a um comando *STOP*	
A	Argumento inválido p/uma função existente na linha indicada	Dê *PRINT* na função existente na linha indicada p/conformar de *PRINT* na função; verifique a sintaxe correta

B	O arredondamento de um número real, via *INT*, resultou num valor inválido	Verificar a função de arredondamento
C	Função *VAL* com argumento inválido	
D	Programa interrompido por *BREAK;* ou *STOP* mal usado	
F	Especificação de uma STRING vazia numa operação *SAVE*	

b) Continuação após interrupção por erro

A aplicação do comando *CONT* vem a ser o mesmo que o uso de *GOTO M* onde M é o número da linha exibido juntamente com o código de erro; com exceção do código 9, quando deve ser *GOTO M + 1.*

c) Cursores de edição:

Símbolo	Interpretação
>	Cursor do programa; indica a linha onde se encontra
K	Esperando um comando ou instrução BASIC
L	Os caracteres digitados a seguir são de exclusiva responsabilidade do usuário
S	Indicador de erro de sintaxe (construção inválida da linha)
G	Trabalhando em modo gráfico ou vídeo inverso

APÊNDICE – C

GLOSSÁRIO DE ALGUNS TERMOS TÉCNICOS MAIS USADOS

Arquivo:
Conjunto de informações (armazenadas em fita ou disco) tratadas como uma só unidade; dividem-se em registros que correspondem à menor unidade endereçável; os registros, por sua vez, podem dividir-se em campos.

Arquivo Aleatório ou Randômico:

Permite ir direto à informação desejada.

Arquivo Indexado:
É a combinação do arquivo aleatório com o seqüencial. É aleatório quanto ao acesso, seqüencial quanto à organização.

Arquivo Seqüencial:
O acesso a uma informação só é possível passando por todas as anteriores.

Banco de Dados:
Arquivo de dados (informações) devidamente estruturados e logicamente interligados.

Backup:
Cópia de reserva do programa (do software) para ser usado em caso de perda ou dano no original.

Buffer:
Dispositivo de armazenamento temporário de dados; normalmente utilizado entre o computador e uma impressora.

Byte:
Menor unidade de informação endereçável na memória; compreende 8 bits.

Chip:
Circuito integrado formado por uma pastilha de silício (geralmente com menos de 1/2 cm^2 de área), contendo milhares de componentes microscópicos interligados.

Compilador:
Programa que converte (traduz) uma linguagem de alto nível (BASIC, por exemplo) para a linguagem de máquina.

CPS:
Caracteres por segundo; mede a velocidade de impressão de impressoras seriais.

CPU:
Unidade central de processamento: responsável pelo processamento, decodificação e execução dos comandos/instruções.

Cursor:
Símbolo, indicando na tela a posição de entrada do próximo caractere.

Densidade de Gravação:
Quantidade de informação armazenada por unidade de comprimento (centímetro ou polegada) de uma fita magnética ou uma trilha de um disquete. A densidade pode ser simples ou dupla: no primeiro caso, é possível armazenar cerca de 170 KB em cada face de um disquete de 5 1/4''; no segundo caso, cerca de 350 KB.

Diretório:
Índice de arquivo que define localização, tamanho, nome e tipo das informações contidas no disco.

Disco:
Peça circular sobre cuja superfície magnetizada são armazenadas informações.
O disco pode ser flexível ou rígido. O disco flexível (disquete ou "Floppy Disk") é produzido com material flexível e protegido por envelope plástico. Permite a gravação em uma ou ambas as faces.
A quantidade de informações armazenadas pode variar de 80 KB a 1 MB, dependendo do tamanho, face, densidade e características de gravação.
O disco rígido é maior, com mais capacidade; pode ser fixo (Winchester) ou removível.

Disk Drive:
Também chamado de controlador de disquete. Unidade mecânica de comando do disco.

EAROM:
Corresponde à "Electrical Alterable Read Only Memory"; uma ROM que pode ser eletricamente apagada e reprogramada.

EEPROM:

"Electrically Erasable and Programmable ROM", memória semicondutora, não-volátil, onde não há perda de informação na ausência da fonte de alimentação.

EPROM:
Corresponde à "Erasable Programmable Read Only Memory"; uma ROM especial que pode ser apagada com luz ultravioleta e reprogramada.

Firmware:
Conjunto de programas (Software) permanentemente gravados na memória, usualmente na ROM.

Hard Copy:
Exibição de informações na impressora.

Hardware:
Conjunto dos circuitos eletrônicos do computador.

Hexadecimal:
Sistema de numeração de base 16. Utiliza os algarismos 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Interface Serial:
Os dados são transferidos à impressora por seqüência de bits, primeiramente para uma placa e depois para a impressora. Permite o uso de modens. Transmissão mais lenta.

Interface Paralela:
Os dados são transmitidos simultaneamente; distância máxima de 5 metros e de acordo com o cabo.

Impressão Matricial:
Também conhecida por serial; a impressão é caractere a caractere, agulhado, e usualmente numa matriz 7 x 9. Se for direcional, imprime só da esquerda para a direita; se for bidirecional, em ambas as direções.

Impressão Linear:

A impressão é feita linha por linha.

IPS:
Polegadas por segundo. Mede a velocidade da fita magnética.

KB:
Kilobyte: 1.024 bytes de memória.

LOOP:
Ciclo ou laço: conjunto de instruções que deve ser executado várias vezes. Pode ser controlado quanto ao número de repetições.

Memória Virtual:
Técnica que combina memória principal com uma auxiliar, permitindo o uso como uma só grande memória.

Modem:
Modulador/demodulador. Equipamento responsável pela conversão do sinal digital do computador em sinal analógico para transmissão de dados a longa distância, via telefone e vice-versa.

Multiplexador:
Dispositivo que permite que vários sinais sejam tratados por um único canal de comunicação.

Nybble:
Dígito hexadecimal; conjunto de 4 bits.

OEM:
"Original Equipment Manufacturer"; fabricante de equipamento original. Sistema de comercialização, onde o fabricante de um periférico (disco, impressora etc.) só vende aos fabricantes de sistemas, e não ao usuário final.

Palavra:
Na linha Sinclair, uma unidade de informação é composta por 1 byte. (oito bits).
Outros computadores podem apresentar configurações diferentes.

Programa-Fonte:
Código-fonte ou fonte: programa original, escrito em linguagem de alto nível. É utilizado pelo compilador para gerar o programa-objeto que será executado pelo microprocessador.

Programa-objeto:
Código-objeto ou objeto: programa que pode ser executado diretamente pelo processador; resulta da compilação de um programa-fonte.

RAM:
"Random Acess Memory", memória de acesso direto. Parte da memória do computador, onde ficam armazenados o programa, as variáveis, a tela. De livre acesso ao usuário.

ROM:
"Read Only Memory", memória somente de leitura. Parte da memória do computador, onde estão residentes o interpretador ou compilador. Memória não-volátil (não se perde com a ausência de energia). O usuário não tem acesso a esta parte da memória do computador.

RS232-C:
Interface para vídeos, impressoras e modens.

Rotina:
Conjunto completo de instrução que executa determinada tarefa dentro de um programa.

Soft Copy:
Exibição de informações na tela do vídeo.

Software:

Conjunto de programas e aplicativos, sem os quais o computador não teria utilidade.

String:
Seqüência de letras, números ou outros caracteres.

Subrotina:
Trecho de programa que é escrito uma só vez e utilizado várias vezes.

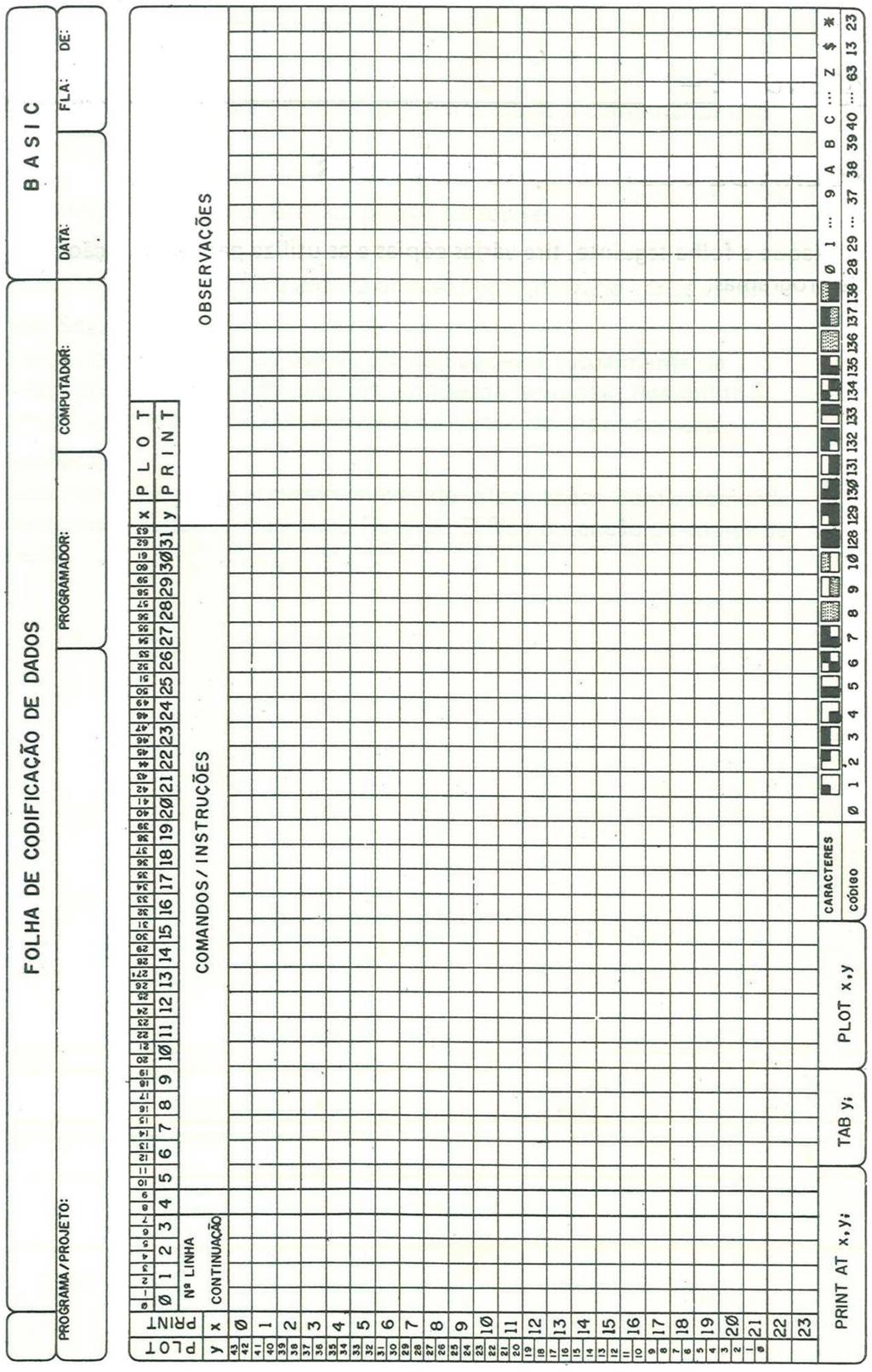
Time Sharing:
Distribuição do tempo de uso de um computador (usualmente de grande porte) entre vários usuários, através de terminais. Possibilita a comunicação entre usuários a software e banco de dados comuns.

Winchester:
Disco rígido fixo para armazenamento de informações, com velocidade e capacidade superiores ao disco flexível. Difícil obtenção de cópias de reserva.

APÊNDICE – D

FOLHA DE CODIFICAÇÃO DE DADOS

Destaque a folha seguinte, tire várias cópias e as utilize para elaboração de programas.



BIBLIOGRAFIA

Bits — Editora Técnica Eletrônica Ltda. Ano 1, nº 2, dezembro de 1983

Hartnell, Tim — *Getting Acquainted With Your ZX81 Creative Computing Press*

Helwet-Packard — *HP 45, programação aplicada*

Lima, Délio Santos — *45 programas para o TK-82C*

Microdigital Eletrônica Ltda. — *TK-85, programação BASIC. Manual do equipamento.*

Microhobby — *Micromega Publicações e Material Didático. Ano 1, nº 2,* junho 83

Pereira Filho, Jorge da Cunha — *BASIC para Micros Pessoais* — Editora Campus, 2ª ed., Rio de Janeiro, 1984.

Rossini, Flávio — *Linguagem de Máquina para o TK,* vol. 1 — micromega — Editora Moderna

Steinbruch, Marília — *Introdução à Linguagem BASIC* — Editora Nobel

Impressão e Acabamento
IONELI - Indústrias Gráficas Ltda..
Rua Viúva Cláudio, 81 - Tel. 201-2644