CURSO DE INFORMÁTICA

2

CURSO DE INFORMÁTICA

BASIC

EDROS
EDITORA ROSA E SILVA LTDA

Curso de Informática
BASIC

Editor — F. da Silva Neto.
Colaborador especial:
Abel De Mariano Camargo.

Wenceslau Carlos Galvão Filho

Curso de Informática

BASIC

FUNDAMENTAL
NÍVEL I e II

VOLUME 2

EXERCÍCIOS

EDROS — Editora Rosa e Silva Ltda.
São Paulo

O COMPUTADOR NÃO FAZ NADA DO QUE QUEREMOS, A NÃO SER QUE QUEIRAMOS O QUE ELE FAZ.

APLICAÇÃO DA LINGUAGEM BASIC E EXERCÍCIOS NO COMPUTADOR CP 500 DA PROLÓGICA

—x—

Obs.:. Todos os erros aqui cometidos podem ser facilmente corrigidos com o uso do próprio computador ,através do processador.

MÓDULO 1

I. INTRODUÇÃO

O Microcomputador PROLÓGICA CP-500 é composto de quatro partes (veja no desenho a seguir).

1. A **unidade central,** que realiza todas as operações do computador.

2. **Teclado,** parecido com o teclado de uma máquina de escrever, utilizado para nós entrarmos com as instruções no computador para que ele realize alguma operação.

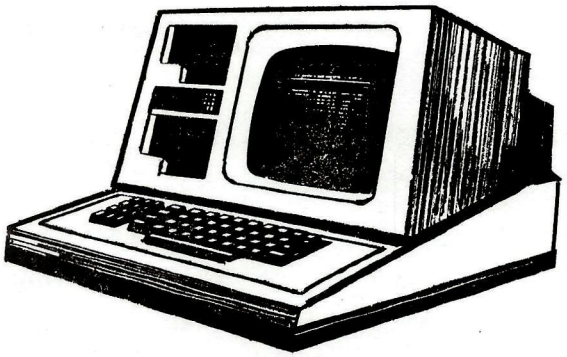
3. **Vídeo** ou **tela,** que servirá para mostrar o que nós estamos digitando e para o computador enviar-nos as mensagens ou respostas das instruções.

4. Para que os dados e instruções (programas) possam ser utilizados novamente, o computador tem um **gravador cassete,** ao que nós já conhecemos. Alguns computadores utilizam unidades de leitura de discos.

Unidade Central

Disketes

Vídeo



Teclado

II. LIGANDO O COMPUTADOR

Ligue o computador no botão que existe atrás, no lado direito do computador.

Aparecerá a seguinte mensagem:

BASIC S/N?

APERTE S

O computador perguntará novamente:

CASS?

Pressione B

Finalmente aparecerá na tela:

MEM USADA?

Simplesmente aperte ENTER.

Vamos ver o que aconteceu. O computador mostrou na tela:

PROLÓGICA BASIC 1981
READY
> —

A palavra READY indica que o computador está pronto para receber instruções (modo de comando). Ela aparecerá sempre ao fim de um programa.

Abaixo de READY aparecem dois sinais:

O sinal (>) indica que o computador está esperando para alguma instrução e o sinal (□) é o CURSOR que indica a posição onde iremos escrever alguma instrução. Toda instrução deve ser enviada ao computador quando estes sinais aparecerem indicando o começo de uma linha.

III. COMUNICAÇÃO

Escreva a palavra **Prológica.** Veja que quando você está escrevendo as letras vão aparecendo na tela na posição do cursor.

Procure no teclado a tecla (seta para a esquerda).

Aperte uma vez e veja o que acontece. Aperte mais algumas vezes.

Explique o que aconteceu:

Então responda como devemos corrigir uma linha errada ou alguma letra errada.

Escreva a palavra **Prológica** novamente na mesma linha e aperte a tecla ENTER.

O que aconteceu? Apareceu a mensagem:

?SN ERRO

Essa mensagem indica que o computador não entendeu o que você escreveu.

O computador só entende algumas poucas palavras como instruções para executar alguma coisa e a palavra **Prológica** não significa nada para ele.

O que significa a tecla ENTER? Ela serve para indicar ao computador que já acabamos de escrever uma linha de instrução e que ele deverá executar esta instrução.

Esta tecla deverá ser pressionada a cada final de linha de instrução.

Vamos aprender o primeiro comando que o computador reconhece:

Escreva no começo de uma linha:

CLS

e aperte ENTER. O que aconteceu? Explique:

Portanto já aprendemos a primeira palavra que o computador reconhece (CLS), e ela serve para limpar a tela e colocar o cursor na primeira posição da tela, ou seja, em cima no lado esquerdo.

A essas palavras que o computador reconhece nós chamamos de **comandos ou instruções.** E elas fazem parte da linguagem do computador, a linguagem BASIC.

Na linguagem BASIC o Comando CLS significa **"limpar a tela e posicionar o cursor na posição mais acima e a esquerda da tela"** que foi o que o computador fez quando apertamos ENTER depois do comando CLS.

IV. TECLADO

Já vimos para que serve a tecla ENTER. Mas no teclado existem outras teclas além das teclas de letras e números.

Repare que as teclas de números tem alguns sinais especiais em cima delas.

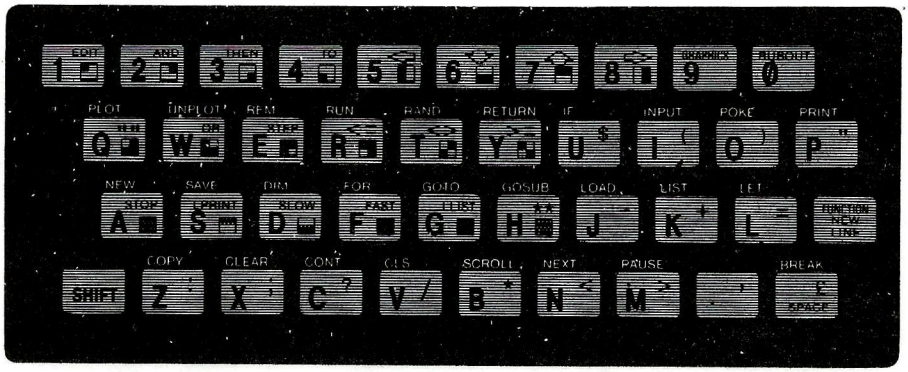
Primeiro devemos apertar a tecla SHIFT (qualquer uma das duas). Sem soltar a tecla SHIFT devemos apertar a tecla que contém o sinal desejado.

A tecla SHIFT será usada toda vez que quisermos escrever um sinal que está na parte de cima da tecla.

Repare que existe um teclado numérico (à direita).

Estas teclas são as mesmas que estão na parte de cima do teclado e servem para facilitar a entrada de números ,pois parecem um teclado de máquina de calcular.

A tecla CLEAR executa a mesma função do comando CLS, ou seja, limpa a tela.


EDIT
1
ANS
2
WHEN
3
TO
4
5
6
7
8
9
0
PLOT
Q
UNPLOT
W
REM
E
RUN
R
RAND
T
RETURN
Y
IF
U
INPUT
I
POKE
O
PRINT
P
NEW
A
SAVE
S
DIM
D
FOR
F
GOTO
G
GOSUB
H
LOAD
J
LIST
K
LET
L
COPY
Z
CLEAR
X
CONT
C
CLS
V
SCROLL
B
NEXT
N
PAUSE
M
BREAK
SHIFT
SPACE

V. INSTRUÇÃO PRINT

Vamos aprender mais um comando BASIC.

Escreva (sempre após o sinal >):

PRINT 2

e aperte ENTER. O que aconteceu?

O computador enviou a mensagem 2 na linha abaixo daquela que você escreveu e a palavra READY e o sinal > — também foram mostrados na tela.

A instrução PRINT significa **mostre na tela** (saída de dados, via tela), e portanto o computador fez aparecer na tela o número 2, pois estava após a instrução PRINT.

O resto da mensagem nós já conhecemos. Significa que o computador já acabou de realizar a instrução e está pronto para mais uma instrução ou comando.

Exemplo: PRINT 3 + 4

(não se esqueça do ENTER)

O computador conhece as quatro operações aritméticas que nós conhecemos:

ADIÇÃO	+
SUBTRAÇÃO	−
MULTIPLICAÇÃO	*
DIVISÃO	/

Já vimos como utilizar o comando PRINT para imprimir resultados numéricos na tela. Como poderemos fazer para que ele mostre letras ao invés de números?

Tente a seguinte instrução:

PRINT "COMPUTADOR PROLÓGICA"

(Não se esqueça das aspas)

A instrução PRINT também é utilizada para imprimir mensagens na tela, sem executar nenhuma operação.

Para isso devemos utilizar as aspas (") para indicar ao computador o que deverá ser impresso.

Compare o resultado desses dois comandos:

1 — PRINT 3 + 2 * 3

2 — PRINT "3 + 2 3"

Efeito:

1 — Nesta caso será listado 9. (numérico)

2 — E aqui 3 + 2 * 3 (string)

Quando quisermos que o computador imprima uma mensagem devemos colocar o conteúdo desta mensagem entre aspas indicando o começo e o fim dessa mensagem.

Caso você coloque a mensagem entre aspas o computador poderá indicar um erro ou poderá mostrar o número 0 (zero). Mais tarde você entenderá porque.

Faça algumas tentativas com o comando PRINT, não esquecendo-se das aspas.

A instrução PRINT é um comando direto, ou seja, quando nós escrevemos alguma instrução PRINT e apertamos ENTER, o computador executará automaticamente.

Quando temos após o comando PRINT uma mensagem entre aspas, nós chamamos esta mensagem de **cadeia de caracteres** ou **string.** Uma **string** pode incluir além de letras e números todos os sinais especiais, menos o sinal de aspas (") ,porque este sinal é utilizado para determinar o começo e o fim da **string.**

Como poderíamo instruir o computador para que imprimisse ao mesmo tempo uma **string** e um número ou duas **strings?**

Para isso separamos os ítens a serem impressos com vírgulas (,) e ponto e vírgula (;).

Exemplo: PRINT "A SOMA DE 2 É IGUAL A"; 2 + 3

ponto e vírgula

PRINT 5 * 4 , "COMPUTADORES"

vírgula

Tente os comandos acima e veja a diferença entre os dois resultados.

Então explicando:

A **vírgula** (,) é utilizada para separar os ítens a serem impressos no comando PRINT, deixando um espaço entre os ítens enquanto o **ponto e vírgula** (;) une os dois ítens.

Tente esta instrução:

```
PRINT "JOÃO" ; "E" ; "PEDRO"
```

O resultado não foi muito bom.

```
PRINT 'JOÃO" , "E" , "PEDRO"
```

Agora tente desta forma:

```
PRINT  "JOÃO"  " ; "E" ; "  PEDRO"
```

(deixe estes espaços em branco)

Estes sinais também podem separar números.

EXEMPLO:

```
PRINT 2, 3, 4, 5

PRINT 2; 3; 4; 5

PRINT 2. 3, 4. 5, 6
```

Tente o seguinte comando:

```
PRINT '.........'; " TEM "; ..... *12;"
MESES DE VIDA"
```

(escreva seu nome) (escreva sua idade)

MÓDULO 1 — EXERCÍCIOS

1) Complete:

   a) o que significa o sinal > — que aparece na tela?

   b) Para que serve a tecla > — ?

   c) Para que serve a tecla ENTER?

   d) Para que serve o comando CLS ?

2) Complete a linha pontilhada indicando o resultado do Computador:

   a) PRINT 2+2+2 .....
   b) PRINT 4*3 .....
   c) PRINT 4*5−9 .....
   d) PRINT 3/2 .....
   e) PRINT 5+4*2 .....
   f) PRINT (5+4)*2 .....
   g) PRINT 10*25 .....
   h) PRINT (1024+3066)/5−18 .....
   i) PRINT 2.25 * 3.5 .....
   j) PRINT 1.5+2.75+3.25−0.5 .....
   k) PRINT (5/9)*(3+2)−(1/2) .....

3) Faça os seguintes exercícios utilizando o comando PRINT

a) Se temos 270 cestas e em cada cesta cabem 322 laranjas, quantas laranjas eu posso transportar?

Resposta:.....

b) Qual a área de um triângulo de base 120 e altura 25? (Área = (Base × Altura/2).

Resposta:.....

c) Qual a soma de uma triângulo de base 32 e altura 44 e um quadrado de lado 25? (Utilize apenas uma instrução PRINT).

Resposta:.....

4) Verifique se estes comandos estão corretos. Se não estiverem corrija-os ao lado.

a) PRINT 2*3, 3*4, 8*9

b) PRINT 84*1135 = 20 * 4767

c) PRINT 3*4;" = ";2:6

d) PRINT "PROLÓGICA "CP;500

e) PRINT "FEIRA DE" "INFORMÁTICA"

f) PRINT −3*−4

g) PRINT (2* (3+4)−2

h) PRINT PRINT

i) PRINT CLS

j) CLS

l) PRINT 2.5*3,4.2*2

m) PRINT 2;" + ";3;" = ";2+3

1.°)

A—) SIGNIFICA QUE VOCÊ JÁ PODE COMEÇAR A DIGITAR.

B—) SINAL DE MENOR.

C—) EXECUTAR OS COMANDOS

D—) LIMPAR A TELA.

2.°)

A—) 6

B—) 12

C—) 11

D—) 1.5

E—) 13

F—) 18

G—) 250

H—) 800

I—) 7875

J—) 7

K—) 2.27778

3.°)

A—) 86940

B—) 1500

C—) 1280

4.°)

A—) ESTÁ CORRETO.

B—) ESTÁ CORRETO.

C—) PRINT 3*4, "="; 2*6

D—) PRINT "PROLOGICA CP"; 500

E—) ESTÁ CORRETO.

F—) ESTÁ CORRETO.

G—) PRINT (2* (3+4) − 2)

H—) PRINT : PRINT... É NECESSÁRIO DOIS PONTOS ENTRE UM COMANDO E OUTRO.

I—) PRINT : CLS... IDEM AO ITEM H.

J—) ESTÁ CORRETO.

L—) ESTÁ CORRETO.

M—) ESTÁ CORRETO.

MÓDULO 2 — TEORIA

I. VARIÁVEIS ALFANUMÉRICAS (STRINGS)

O computador tem uma memória que permite em certas áreas guardar valores. Essas áreas pareceriam caixas onde podemos dar nomes a essas caixas. Essas caixas com nomes são as **variáveis** e podem armazenar informações.

Nós podemos dar nomes a essas variáveis ou caixas:

A$

B$

C1$

Para que nessas caixas possam estar gravada uma STRING (um nome, números, o usinais especiais) é necessário que os nomes das caixas, as variáveis sigam o seguinte modo:

Devem começar por uma letra ,e devem terminar por $, e não podem ter sinais especiais.

Exemplo de nomes de variáveis STRING possíveis:

A$ B1$ AB$ Z9$

Contra-exemplos:

1A$ (começa por número)
$B (começa com sinal especial)
A+$ (contém sinais especiais)
AB (não possue $)

Vamos imaginar duas caixas dentro do computador com os nomes A$ e B$.

Podemos agora imaginar que na caixa A$ colocaremos a STRING "COMPUTADOR" e na caixa B$, "PROLÓGICA".

A$ | COMPUTADOR |

B$ | PROLÓGICA |

Como poderemos colocar essas STRINGS nas variáveis?

Deveremos utilizar o comando LET e o sinal = da seguinte forma:

LET A$ = "COMPUTADOR"
LET B$ = "PROLÓGICA"

A instrução LET no primeiro comando significa **faça a variável A$ conter a palavra "COMPUTADOR".**

Não se esqueça que quando estamos usando STRINGS devemos colocá-la entre aspas para o computador saber exatamente qual é a STRING.

Execute os comandos acima. O que aconteceu após o comando?

Nada. Isto porque não existe nenhuma instrução que peça ao computador uma resposta, mas apenas que guarde a palavra na variável.

Mas como poderemos saber que o computador guardou a palavra "COMPUTADOR" na variável A$?

Experimente o seguinte comando:

PRINT A$

Qual foi a resposta do computador? .....
(complete).

Quando fizemos a instrução PRINT A$, o computador entendeu como:

MOSTRE O VALOR DA VARIÁVEL A$

Portanto podemos guardar os valores das variáveis com a instrução LET e o sinal de = , e podemos ver o conteúdo das variáveis através do comando PRINT.

Existem algumas observações importantes:

* **O comando LET é opcional, não precisa ser escrito, podendo o comando só conter o sinal de = :**

EX.: A$ = "COMPUTADOR"

* **O nome da variável deve sempre vir antes do valor que estamos atribuindo a ela.**

EX.: C1$="SÃO PAULO"

"COMPUTADOR"=A$ (ERRADO)

Se N$="NOME" considere o seguinte comando:

Z$=N$

Qual o valor da variável Z$? (responda aqui)

Neste caso igualamos o valor de duas variáveis, ou seja a primeira conterá o valor da segunda.

O que acontecerá se fizermos:

Z$="RIO DE JANEIRO"

O valor da variável Z$ passará a ser RIO DE JANEIRO e o valor antigo será eliminado.

Uma variável permanecerá com o seu valor até que você faça-a valer outro ou quando o computador for desligado.

Existe outra maneira de eliminar o valor das variáveis, mas de **todas elas.** É com o comando NEW.

Escrevendo NEW (e apertando ENTER, lógico), toda a memória do computador será esquecida.

II. VARIÁVEIS NUMÉRICAS

Da mesma forma que podemos ter variáveis STRING, temos **variáveis numéricas,** que possibilitam armazenar números e executar operações com essas variáveis.

Essas variáveis numéricas também têm nomes (pense nas caixas que vimos nas variáveis STRING) e devem seguir essas regras:

> **Devem começar por uma letra, não devem possuir sinais especiais.**

Portanto são iguais as variáveis STRING mas sem o sinal de $.

Exemplos de **variáveis numéricas** possíveis:

A B1 BC FF

Exemplos da variáveis numéricas erradas:

1A (começa por número)
C$ (possui $)
G+ (possui sinais especiais)

Da mesma forma que para variáveis STRING utilizamos o comando LET para dar o valor a variável.

Lembre-se que o comando LET é opcional:

LET B1=23.5 ou B1=23.5

LET A =120 ou A=120

E também da mesma forma podemos verificar o valor da variável através do comando PRINT:

PRINT A

PRINT B1

PRINT A,B1

Tente esses comando

Também podemos passar o valor de uma variável para outra:

C=A1 (neste caso o valor da varáivel C será igual ao valor de A1).

E também da mesma forma podemos alterar o valor de uma variável somente atribuindo a ela novo valor.

Tente os comandos abaixo:

```
D1=100
PRINT D1
D1=2345
PRINT D1
```

Lembre-se que o comando NEW serve para cancelar o valor das variáveis. (Todas as variáveis numéricas e STRING).

Podemos também executar operações com as variáveis:

A1=2 (A variável A1 vale 2)

B=3*A1 (A variável B vale 3vezes o valor da variável A1)

D=A1*B

(Complete: A variável D vale .....

Verifique o valor das variáveis A1, B, D utilizando apenas um comando PRINT.

Resposta: .....

Tente esse comando e verifique o valor da variável A1:

A1=2*A1

Neste comando o valor da variável A1 foi multiplicado por 2 e foi colocado na mesma variável A1.

III. PROGRAMAS

Já vimos como o computador realiza operações simples e como mostra os resultados. Como poderemos fazer para que o computador guarde essas instruções na memória para podermos realizar diversas operações sem precisar escrever novamente o comando?

Isto se faz através de um **programa.** Um programa é uma seqüência de instruções numeradas e quando executadas pelo computador realizam operações.

Exemplo de um programa:

```
10 CLS
20 PRINT "COMPUTADOR"
30 END
```

Vamos ver algumas características de um programa:

A) As linhas são numeradas indicando a ordem de execução das instruções. No exemplo acima a primeira linha a ser executada é a linha 10, depois a 20 e por final a 30.

B) Os números das linhas indicam que a instrução deve ser guardada na memória, nas "caixinhas" com este número.

10	CLS
20	PRINT "COMPUTADOR"
30	END

C) Dentro de cada caixinha será guardado um comando Basic igual ao que já conhecemos e outros que iremos aprender.

D) Esse comando que foi guardado na memória do computador só será executado quando nós mandarmos através de outro comando BASIC.

E) O comando NEW também vai apagar todas as instruções que estão guardadas na memória do computador.

Vamos tentar com o exemplo acima. Antes de escrever as instruções, limpe toda a memória do computador com o comando NEW.

Escreva os comandos como estão descritos, não se esquecendo de apertar ENTER após cada linha para que o computador guarde na memória.

Veja que quando uma linha começa com um número, após apertarmos ENTER nada acontece, ou seja

o comando não é executado. Isso ocorre porque o computador guarda o comando na memória (na caixinha número 10 por exemplo) para depois ser executado.

Depois que você entrar com as linhas 10, 20 e 30 (não se esqueça do ENTER), escreva o seguinte comando:

LIST (ENTER)

Responda aqui o que aconteceu:

Então para que serve o comando LIST? Ele serve para verificar a memória de programa do computador, ou seja **quais as instruções que foram guardadas nessa memória.**

Porque as linhas são numeradas de 10 em 10?

Não existe obrigatoriedade para que as linhas sejam numeradas de 10 em 10, pois elas podem ser numeradas com qualquer **número inteiro** (1, 3, 9, 10, 54, 100, 98765 . . .). Elas são numeradas assim porque poderemos incluir uma instrução sem precisar renumerar as linhas.

Exemplo:

Escreva: 15 PRINT "ESTA LINHA FOI INCLUIDA"

Agora liste o programa novamente (comando LIST).

Responda aqui o que aconteceu:

Quantas instruções poderemos incluir entre as linhas 20 e 30?

Para executar o programa que está guardado na memória necessitamos do comando

RUN

Preste atenção que tanto o comando RUN como NEW como LIST são comandos diretos, sem número de linha antes do comando.

Responda o que o computador executou após o comando RUN:

Quando você apertou ENTER após o comando RUN, o computador foi executar as instruções do programa.

10 CLS (O computador limpa a tela)

15 PRINT "ESTA LINHA FOI INCLUIDA" (Imprime ESTA LINHA FOI INCLUIDA)

20 PRINT "COMPUTADOR" (Imprime COMPUTADOR)

30 END (Este comando é necessário para terminar).

O programa indica que o computador deve parar e a mensagem READY aparece na tela.

Da mesma forma que nas variáveis, podemos alterar o conteúdo de uma linha colocando outra instrução em lugar da instrução original.

Experimente com:

15 PRINT "ESTA LINHA FOI SUBSTITUIDA"

Agora liste o programa para verificar.

E como podemos fazer para eliminar uma linha? Devemos colocar uma instrução em branco no lugar, ou seja somente escrevendo o número da linha e apertando ENTER.

Experimente apagar a linha 15.

Vamos ver algumas coisas importantes sobre programas:

A) Se você escrever um comando errado (Por exemplo 10 PRINTT "MICRO", o computador vai guardar a instrução errada. Na hora que você for executar vai aparecer a seguinte mensagem de erro:

?SN ERRO em 10

O número no final indica a linha onde está o erro (no exemplo linha 10).

B) Para corrigir uma instrução, você deve escrever a linha inteira novamente.

Vamos ver alguns exemplos de programas.

Exemplo 1

```
10 CLS
20 A=100
30 B=200
40 C=A*B
50 PRINT A; "MULTIPLICADO POR"; B; "=";C
60 END
```

Exemplo 2

```
10 CLS
20 A$="                "      (coloque seu nome)
30 PRINT "MEU NOME É";A$
40 END
```

Exemplo 3

```
10 CLS
20 A$="                "      (coloque seu nome)
30 B= . . .                   (coloque sua idade)
40 PRINT A$;"TEM";B;"ANOS"
50 END
```

Lembre-se:

1) **Sempre use NEW antes de começar um novo programa.**
2) **Sempre termine um programa com END.**

MÓDULO 2 — EXERCÍCIOS

1) Se:

A$ = "MICRO"

B$ = "COMPUTADOR"

C$ = "PROLÓGICA"

D$ = "CP500"

Qual o efeito das instruções:

PRINT A$;B$;C$;D$ .....

PRINT D$;" ";C$ .....

PRINT A$;A$;A$ .....

PRINT B$;"S";"PROLÓGICA" .....

2) Com os mesmos valores acima quais as instruções necessárias para que seja listado:

MICRO PROLÓGICA COMPUTADOR

PROLÓGICA MICROCOMPUTADOR

3 Verifique os comandos abaixo e indique o erro, se houver:

A=234.5

B=2, 3, 5

C="CORCOVADO"

D$=1234

524=Z9

A1=B2=9

PRINT B, C2, Z$, H

PRINT A1=2

4) Coloque os valores nas variáveis e responda sem usar o computador as respostas dos comandos (antes de começar o exercício limpe a memória do computador com NEW).

```
A1 = 25

B2 = 2.5

C1 = 2

PRINT 2*A1,C1*B2

PRINT "A ÁREA DE UM QUADRADO DE LADO';
      B2;"É";B2*B2

PRINT A1*C1/B2/(C1*B2)
```

5) Ainda com os mesmos valores das variáveis qual vai ser o valor da variável Z após todos esses co-comandos?

```
Z=A1*C1

Z=C1*Z

Z=Z/10 + 10

Z=Z + Z + Z
```

Responda: O valor da variável Z é .....

6) Complete:

A) Para que serve o comando NEW?

B) Para que serve o comando END?

C) Para que serve o comando LIST?

D) Para que serve o comando RUN?

E) Porque numeramos as linhas de 10 em 10?

F) Escreva os seguintes comandos:

```
NEW
1O A=10
PRINT A
```

Explique o porque deste resultado.

GABARITO DOS MÓDULOS

1 — 2 — 3 — 4 — 5

1—)
```
10 CLS
20 PRINT TAB (29) "MICRO"
30 PRINT TAB (27) "COMPUTADOR"
40 PRINT TAB (27) "PROLÓGICA"
50 PRINT TAB (28) "CP—" 500
60 END
```

2—)
```
10 CLS
20 INPUT "DIGITE QUALQUER VALOR..." ;NR
30 PRINT USING "$$                    "; NR
40 END
```

3—)
```
10 CLS
20 INPUTE "DIGITE O SEU SALÁRIO BRUTO";SB
30 IAPAS=SB*0.08
40 SLL=SB-IAPAS
50 PRINT TAB (3) "SALÁRIO BRUTO"; TAB (25)
   "IAPAS"; TAB (36) "SALÁRIO LIQ."
60 PRINT USING "$$                    "; SB;
   IAPAS; SLL
70 END
```

1.°) PÁGINA DOS EXERCÍCIOS:

EXEMPLO 1: R= 100 MULTIPLICADO POR 200 = 20000.

EXEMPLO 2: R= MEU NOME É (IMPRIMIRÁ SEU NOME).

EXEMPLO 3: R= (SEU NOME) TEM (SUA IDADE) ANOS.

2.°) PÁGINA DOS EXERCÍCIOS: N.° 1

1—) MICROCOMPUTADORPROLOGICACP500

2—) CP500 PROLÓGICA

3—) COMPUTADORPROLOGICA

2.°) PÁGINA DOS EXERCÍCIOS: N.° 2

1—) PRINT A$, C$, B$?

2—) PRINT C$; D$; A$; B$

2.°) PÁGINA DOS EXERCÍCIOS: N.° 3

A1=234.5 ESTÁ CERTO.

B=2, 3, 5 ESTÁ ERRADO. B=23.5.

C='CORCOVADO" ESTÁ ERRADO.

C$"CORCOVADO"

D$1234 ESTÁ ERRADO. D=1234

524=Z9 ESTÁ ERRADO. Z9=524

A1=B2=9 ESTÁ ERRADO. A1 NÃO É = B2 QUE NÃO É = 9

PRINT A1=2 ESTÁ ERRADO. A1 NÃO É =A2

PRINT B, C2, Z$, H ESTÁ ERRADO. PRINT B, C$, Z9 À VARIÁVEL H NÃO TEM VALOR.

4.° EXERCÍCIO — 3.ª PÁGINA.

1.) PRINT 2*A1,C1*B2=50 5

2.°) A ÁREA DE UM QUADRADO DE LADO 2.5 = 6.25

5.° EXERCÍCIO — 3.ª PÁGINA.

1.°) O VALOR DA VARIÁVEL Z É = 60

6.° EXERCÍCIO — 3.ª PÁGINA.

A—) LIMPAR A MEMÓRIA.

B—) TERMINAR UM PROGRAMA .

C—) LISTAR UM PROGRAMA.

D—) EXECUTAR UM PROGRAMA.

E—) PARA TERMOS NOVE (9) LINHAS PARA INSERIR OUTRAS INFORMAÇÕES.

F—) PORQUE TODO NÚMERO DIGITADO ANTES DE TUDO SERÁ ASSUMIDO COMO NÚMERO DE LINHA; ENTÃO A VARIÁVEL **A** ESTÁ CONTIDA EM UM PROGRAMA, PARA SABER O SEU VALOR TEREMOS QUE INSERIR MAIS UMA LINHA COM O COMANDO **PRINT A** E APÓ USARMOS O **RUN.**

MÓDULO 3 — TEORIA

1. COMANDO GOTO

Já vimos como determinar variáveis e como escrever programas.

Já conhecemos os seguintes comandos:

PRINT

LET (opcional)

CLS

NEW

LIST

RUN

END

Um programa executa as instruções de acordo com a ordem dos números das linhas.

No Basic existe o comando GOTO que **desvia a ordem de execução do programa.**

Repare neste programa:

```
10 CLS

20 PRINT "COMPUTADOR"

30 GOTO 20
```

As duas primeiras instruções nós já conhecemos. Na linha 30, o comando GOTO 20 fará com que o programa volte para a linha 20 e execute novamente esta linha. E qual será a seqüência deste programa? Ele executará a linha 20 depois a 30, depois a 20 ,30, 20, 30, 20.....

Este programa não necessita da instrução END porque não tem fim. Se executarmos este programa ele só parará quando apertarmos a tecla BREAK.

Faça este teste.

A mensagem que o computador mostrará após apertarmos BREAK indica que o programa foi interrompido e que o computador está disponível para novos comandos.

Tente este outro programa (não se esqueça de NEW).

```
10 CLS

20 PRINT "LINHA 20"

30 PRINT "LINHA 30"

40 GOTO 60

50 PRINT "LINHA 50"

60 PRINT "LINHA 60"

70 END
```

Execute este programa e explique o que aconteceu (porque a linha 50 não foi executada?)

Portanto o comando GOTO é utilizado não só para retornar mas também para avançar a execução normal de um programa.

Escreva novamente o primeiro programa modificando a linha 20 para:

| ponto e vírgula |

```
20 PRINT "COMPUTADOR";
```

| deixe este espaço |

Veja o que aconteceu. O ponto e vírgula colocado no final da instrução fez com que a palavra COMPUTADOR fosse escrita uma ao lado da outra.

Tente responder sem utilizar o computador:

O que aconteceria se ao invés de ponto e vírgula (;) tivéssemos vírgula (,)?

Como poderíamos mostrar os números em seqüência, utilizando o comando GOTO (1, 2, 3,...)?

Veja:

```
10 CLS
20 A = 0
30 PRINT A
40 A = A + 1
50 GOTO 30
```

Execute este programa e veja o que acontece.

Preste atenção nas linhas 30 a 50. Na primeira passagem o número 0 é impresso na tela (instrução 30) e depois é somado 1 a variável A (instrução 40) para o programa voltar a instrução 30, imprimindo 1 (o novo valor da variável A), somando mais 1, voltando a linha 30, ...

Lembre-se que a tecla BREAK **pára** o programa.

Como poderíamos alterar o programa acima para que ele imprima só os números pares?

IMPORTANTE: Se fizermos o desvio com o GOTO para uma linha que não existe no programa o computador apresentará erro.

II. COMANDO INPUT

Vamos supor o seguinte problema:

Desejamos calcular a área de **diversos** quadrados.

Veja este programa:

```
10 CLS
20 A=100
30 PRINT "A ÁREA DE UM QUADRADO DE
   LADO";A;" = " A*A
40 END
```

O inconveniente deste programa é que para cada quadrado teremos que mudar a linha 20, colocando o valor do lado na variável A.

Exemplo:

```
20 A = 120

RUN
```

O computador responderá:

```
A ÁREA DE UM QUADRADO DE LA-
DO 120 = 14400

20 A = 80

RUN
```

O computador responderá:

A ÁREA DE UM QUADRADO DE LADO 80 = 6400 e assim por diante.

Para evitar que alteremos uma linha de programação, existe o comando

INPUT

O comando INPUT precisa de que indiquemos uma variável:

INPUT A (numérico)

INPUT B$ (string)

Execute o seguitne programa (não se esqueça de NEW):

```
10 CLS
20 INPUT A
30 PRINT "O VALOR DA VARIÁVEL A=";A
40 END
```

Digite RUN:

O computador limpou a tela (comando CLS na linha 10) e imprimiu um ponto de interrogação e o cursor apareceu:

Neste momento o computador executou o comando INPUT que significa:

— Pare o programa.

— Imprima um ponto de interrogação e faça aparecer o cursor.

— Espere para que entre com um dado que será introduzido na variável A. (No caso números).

Entre com um número e aperte ENTER.

Nesse momento o computador continua a execução do programa e o número que você escreveu foi colocado na variável A.

```
INPUT A
```

Execute o programa novamente (RUN) e coloque outro valor (não esqueça do ENTER após o valor).

Portanto o comando INPUT serve para que possamos colocar valores nas variáveis **durante** a execução do programa.

É importante notar que, se quisermos colocar um dado alfanumérico (STRING) pelo comando INPUT devemos utilizar uma variável de STRING:

```
INPUT   A$

INPUT   B1$
```

Veja este programa:

```
10 CLS

20 INPUT A$

30 PRINT A$

40 GOTO 20
```

Neste programa ,o computador sempre retorna ao comando INPUT, fazendo aparecer ? — e esperando para um novo valor para A$.

Execute este programa não esquecendo de ENTER para indicar ao computador que você já acabou de entrar com o dado e que ele deve seguir a execução.

(Para pararmos com esse programa utilize BREAK)

Faça uma modificação na linha 20 do programa, substituindo-a por:

```
20 INPUT "QUAL O SEU NOME" ; A$
```

Veja o que acontece se executarmos o programa (RUN).

```
INPUT "QUAL O SEU NOME" ; A$


QUAL O SEU NOME?
```

Este é um comando INPUT que imprime uma mensagem na tela para depois pedir uma entrada de dados (o sinal? faz parte do comando INPUT normal).

Desta forma podemos programar mensagens na tela para nos guiar na entrada de dados através do comando INPUT, evitando assim que ocorram erros como:

```
10 CLS
20 INPUT A
30 PRINT A
40 END
RUN
?JOÃO      (ENTER)
.......... (complete com a mensagem)
```

Neste caso a variável após o INPUT é numérica e o dado introduzido é alfanumérico (STRING)

MÓDULO 3 — EXERCÍCIOS

Fazer um programa que:

1) Calcule a área de um quadrado qualquer.
2) Liste na tela os números pares a partir de zero.
3) Aceite a digitação de um nome e seja listado na linha de baixo.

1.°)
```
10 CLS
20 INPUT A
30 ?A*A
40 END
```

2.°)
```
10 CLS
20 A=0
30 PRINT A
40 A=A + 2
50 GOTO 30
```

3.°)
```
10 CLS
20 INPUT A$
30 PRINT A$
40 END
```

MÓDULO 4 — TEORIA

I. IF ... THEN

Suponha que o computador tenha que determinar a "relação" entre duas variáveis numéricas (ou expressões), por exemplo entre **A** e **B.** O que pode ocorrer?

1 — A pode ser maio rque B.

2 — A pode ser menor que B.

3 — A pode ser igual a B.

Para isso usamos o comando IF ... THEN (que significa SE ... ENTÃO) que faz um teste de comparação entre variáveis.

Veja o exemplo:

•
•
•

30 IF A > 3 THEN PRINT "A É MAIOR QUE B"

Esta instrução significa:

| SE A FOR MAIOR QUE B ENTÃO
| IMPRIMA "A É MAIOR QUE B".

O comando IF . . . THEN é dividido em duas partes.

Na primeira ,testamos uma condição:

IF A > B

IF C = D

IF A < = X

IF A$ = "SIM"

IF D > 2

Essa condição deve vir logo após o IF e deve conter um teste lógico que poderá usar os seguintes sinais:

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

O comando IF deve testar se esta condição é verdadeira ou falsa.

Exemplo:

IF A > 6 SE A = 3 É FALSA
SE A = 8 É VERDADEIRA
SE A = 6 É FALSA

IF A < = B SE A=4 E B=5 ENTÃO É VERDADEIRA
SE A=4 E B=4 ENTÃO É.....
SE A=7 E B=10 ENTÃO É.....

Portanto este teste deve dizer se a expressão é verdadeira ou falsa.

Se a expressão for falsa, o resto do comando é ignorado e o programa continua normalmente.

Se a expressão for verdadeira então o programa vai executar o resto do comando, ou seja a instrução que vier após o THEN, continuando depois normalmente na seqüência do programa.

FORMA GENÉRICA DO IF ... THEN

Instrução 1 IF VERDADEIRO THEN instrução (s)

FALSO DEPOIS

Instrução 2 CONTINUAÇÃO

Veja este programa:

```
10 CLS
20 PRINT "ENTRE COM UM NÚMERO E EU
   DIREI SE ELE É"
30 PRINT "POSITIVO, NEGATIVO OU ZERO"
40 INPUT "QUAL O NÚMERO";X
50 IF X > O THEN PRINT "É POSITIVO"
60 IF X < O THEN PRINT "É NEGATIVO"
70 IF X = 0 THEN PRINT "É ZERO"
```

Execute este programa e veja os resultados para os números 5, — 8 e zero (um de cada vez).

Portanto o comando tem a seguinte forma geral:

IF	comparação	THEN	comando
	A > B		PRINT "...."
	A$="SIM"		GOTO 90
	B > 10		LET B=20

Em resumo

SE A COMPARAÇÃO APÓS O IF FOR VERDADEIRA ENTÃO EXECUTE O COMANDO APÓS O THEN.

SE A COMPARAÇÃO FOR FALSA ENTÃO CONTINUE O PROGRAMA SEM EXECUTAR A INSTRUÇÃO APÓS O THEN.

Veja este programa:

```
10 CLS

20 INPUT "ENTRE COM SEU NOME, SUA
   IDADE' A$,B

30 INPUT "ENTRE COM O NOME DE SEU
   COLEGA, E A IDADE DELE" C$,D

40 IF B  D THEN GOTO 80

50 IF B  D THEN GOTO 100

60 PRINT A$;"TEM A MESMA IDADE QUE"
   ;C$

70 END

80 PRINT A$; "É MAIS VELHO QUE";C$

90 END

100 PRINT A$;"É MAIS MOÇO QUE";C$

110 END
```

Vamos entender este programa:

Na linha 10 simplesmente limpa a tela.

Na linha 20 e 30, deverão ser entrados o nome e idade de cada pessoa.

Na linha 40 tem a primeira condição que testa se B é maior que D. Se for maior desvia o programa para a linha 80 que deverá imprimir a mensagem que A$ é mais velho que C$ e continua na seqüência encerrando o programa com END. Se não existisse essa instrução END o programa seguiria em frente e executaria a próxima instrução.

Na linha 50 tem o outro teste. Se a primeira condição (linha 40) for falsa então ele segue em frente e testa se B é menor que D.

Se isto acontecer então vai para a instrução 100 imprimir a mensagem e novamente encontra outro END para encerrar o programa.

Se B não é maior nem menor que D, então o programa passa pelas instruções 40 e 50 e portanto B só pode ser igual D imprimindo então a mensagem da linha 60 e terminando na linha 70.

Repare que o programa tem 3 instruções END, uma para cada desvio que possa ocorrer no programa, para que tenha um final lógico para cada opção.

II. O COMPUTADOR ESCOLHE UM NÚMERO

Tente a seguinte instrução (em comando direto, sem súmero da linha).

PRINT RND(3)

Faça mais duas vezes.

O comando RND ( ) faz o computador escolher um número ao acaso (como se fosse uma roleta).

Se tivermos RND (3), o número poderá ser 1,2 ou 3.

Se tivermos RND (2), o número poderá ser 1 ou 2.

Se tivermos RND (10), o número poderá ser de 1 a 10.

Esses números são chamados RANDÔNICOS ou ALEATÓRIOS e a função RND sorteia este número.

Faça este programa:

```
10 CLS
20 A=RND(10)
30 PRINT A;
40 GOTO 20 z
```

Execute (RUN) e pare com a tecla BREAK.

Veja que todos os números estão entre 1 e 10.

MÓDULO 4 — EXERCÍCIOS

1) Faça um programa que o computador sorteia um número de 1 a 5 e você deve tentar adivinhar este número.

   (Dica: use o comando IF . . . THEN para saber se você acertou o número).

2) Faça outro programa igual ao anterior mas que tenha só 3 chances para adivinhar, senão deverá aparecer uma mensagem "VOCÊ PERDEU".

3) Faça um programa de tabuada. Para um número que deve ser entrado o programa deve calcular a tabuada deste número até 10. **(Dica:** lembre-se de um programa anterior onde havia A=A+1. Você deve testar essa variável para que saiam só dez números na tabuada).

1.º)

```
10 CLS

20 INPUT "ADIVINHE UM NÚMERO DE
   1—5";A

30 B=RND (5)

40 IF A=B THEN PRINT "VOCÊ ACERTOU
   . . .": GOTO 70

50 IF A < > B THEN PRINT "VOCÊ ERROU
   . . .":GOTO 70

60 PRINT "VOCÊ GANHOU":END

70 PRINT "VOCÊ PERDEU"

80 END
```

2.°)

```
10 CLS

20 PRINT "VOCÊ TEM TRÊS CHANCES DE ACERTAR..."

30 FOR I = 1 TO 3

40 INPUT "ADIVINHE UM NÚMERO ENTRE 1—5";A

50 B = RND (5)

60 IF A = B THEN PRINT "VOCÊ ACERTOU...": GOTO 100

70 IF A < > B THEN PRINT "VOCÊ ERROU ..."

80 NEXT

90 PRINT "VOCÊ PERDEU...": END

100 PRINT "PARABÉNS VOCÊ ACERTOU ...": END
```

3.º)

```
10 CLS

20 A = 0

30 INPUT "DIGITE NÚMERO A SER MULTI-
   PLICADO";B

40 PRINT A; "*";B "=" ; B*A

50 A = A + 1

60 IF A > 10 THEN GOTO 80

70 GOTO 40

80 INPUT "QUER MULTIPLICAR MAIS
   (S/N)";R$

90 IF R$ = "S" GOTO 10 ELSE END
```

MÓDULO 5 — TEORIA

I. COMANDO AUTO

Desde o início do curso para escrevermos linhas de programa usamos o seguinte recurso:

READY

> 10 CLS (ENTER), por exemplo, ou seja, digitamos o n.° da linha e depois a instrução seguido de ENTER. Para qualquer linha. Com o comando AUTO, o Basic **gera** o n.° de linha sem que precisemos digitá-lo. Assim, o comando AUTO na sua forma genérica é:

> AUTO n,k (ENTER),

onde **n** é o n.° de "linha-início do texto
e **k** o incremento de **n.**

EXEMPLO:

```
                                10 CLS
                      EFEITO
> AUTO 10,10 ———— 20 PRINT "AUTO"
                                30 END


                                 1 CLS
> AUTO  1,5  EFEITO
                      ———— 6 PRINT "AUTO"
                                11 END
```

A partir do ENTER em cada linha ela passa automaticamente à linha seguinte com o incremento dado.

II. COMANDO FOR . . . NEXT

Formato geral:

```
FOR variável = n TO k

    procedimentos

NEXT variável
```

onde **n** e **k** são **expressões numéricas ou variáveis numéricas.**

Considere a seguinte proporção:

Dado o programa ————

```
 5 A = 1
10 PRINT "PASSAGEM" A
20 A = A + 1
30 IF A < 6 THEN GOTO 10
40 PRINT "ACABOU"
```

Analise o programa.

Se usarmos a instrução FOR . . . NEXT:

```
 5 FOR A=1 TO 5
10 PRINT "PASSA-
   GEM" A
20 NEXT A
30 PRINT "ACABOU"
```

Quer dizer que a(s) instrução (ões) entre FOR ↔ NEXT será executado(s) 5 vezes, ou seja, até que | A=5 | a partir de | A=1 | . Toda vez que passar pelo **NEXT A,** retorna para linha 5 e se a condição (A=5) não acontecer, soma 1 em A e continua o LOOP. Então a finalidade desta instrução é permitir que uma série de instruções seja executada dentro de um LOOP um certo número de vezes.

Variável: é usado como contador.

n : é o valor da **variável.**

k : é o **último** valor do contador.

Outro **exemplo** de aplicação:

```
10 CLS
20 PRINT "ADIVINHE UM NÚMERO, ENTRE
   1 — 10"
30 A=RNØ (10)
40 FOR I = 3 TO 1 STEP —1
50 PRINT "VOCÊ AGORA TEM    "; I ;"
   CHANCE(S)"
60 INPUT "QUAL É O NÚMERO";N
70 IF N=A THEN GOTO 100
80 PRINT "VOCÊ ERROU"
90 NEXT I
100 PRINT "UEEIII !!!!! GANNHHOOOUUU"
110 INPUT "QUER MAIS? (S/N)";R$
•
•
•
```

Perceba que o contador "vai" de 3 **até** 1. Isso porque o "incremento" é negativo. **STEP** especifica o incremento ,se omitido assume + **1.** Neste programa usamos FOR . . . NEXT para fazer um LOOP 3 vezes, ou seja, do 3 até o 1, inclusive. Porque do 3 até 1? Porque na linha 50 o programa indica o n.° de chances que o jogador ainda possue, aproveitando do contador (no caso, regressivo) —— I.

Usando o STEP positivo . . .

•

•

•

```
FOR A=32 TO 256 STEP 2
PRINT CHR$(A);
NEXT A
```

•

•

•

No programa acima em cada passagem pelo NEXT, a condição (32 TO 256) é checada ,isto é, se A=256 fim do LOOP, caso contrário seria **2** em **A** continua o LOOP. Então com este incremento o LOOP é executado 5 vezes.

* Não esqueça! (Aproveitando o programa acima ou mesmo num caso geral) na 1.ª execução do LOOP o contador está com conteúdo igual a expressão (ou variável) numérica à direita do sinal de igual.

Assim:

1.ª vez FOR A=32 TO 256 STEP 2

2

5.ª vez A = 256

MÓDULO 5 — EXERCÍCIOS

1) Fazer um programa que liste todos os números pares entre 2 e 100.

2) Fazer um programa onde o operador adivinhe um número. A quantidade de tentativas deve ser fixada pelo próprio operador. (SUGESTÃO: usando FOR . . . NEXT utilizar a variável que contém o número de tentativas para limite do FOR . . . NEXT).

3) Fazer um programa onde seja listado na tela (10 vezes) a mensagem "EU GOSTO DA PRO-LÓGICA".

4) Idem acima com mensagem "FELIZ NATAL".

1.°)
```
10 CLS
20 FOR A=0 TO100STEP2
30 PRINT A
40 NEXT A
```

2.°)
```
10 CLS
20 INPUT "DIGITE NÚMERO DE CHANCES QUE VOCÊ DESEJA JOGAR:";Y
30 FORI=YTO1 STEP—1
40 PRINT "VOCÊ AGORA TEM ";I;" CHANCE(S)"
50 INPUT "ADIVINHE UM NÚMERO ENTRE 1—10";A
60 B=RND(10)
70 IF A=B THEN PRINT "VOCÊ ACERTOU!": GOTO 100
80 PRINT "VOCÊ ERROU..."
90 NEXT I
100 PRINT "** JOGO TERMINADO ** "
110 INPUT "QUER JOGAR MAIS (S/N)';R$
120 IF R$= "S' THEN GOTO 2O ELSE END.
```

3.°)
```
10 CLS
20 FOR A=1TO10
30 PRINT "EU GOSTO DA PROLÓGICA"
40 NEXT A
```

4.°)
```
10 CLS
20 FORA=1TO10
30 PRINT "FELIZ NATAL"
40 NEXT A
```

MÓDULO 6 — TEORIA

COMANDOS BASIC — CP - 500

Todos os comandos citados são válidos somente dentro do Basic, daí "COMANDOS BASIC".

1) | AUTO n,k |

onde n é o número da linha.
k é o incremento.

Finalidade: Gerar automaticamente o número da linha.

Exemplo: AUTO 10,5 —— efeito —— 10,15,20, 25,. . . .

2) | CLEAR n |

——— onde n especifica que serão reservados n bytes para armazenagem de string e todas as variáveis são zeradas.

Exemplo: CLEAR 2000

3) | CONT |

——— continuar a execução do programa após um Break, Stop ou End.

4) | DELETE n—k |

——— onde n ↔ é o intervalo de linhas que serão eliminadas.

Eexmplo: DELETE 10—50

5) | LIS n—k |

——— **opções:** List n (a)

List n—k (b)

List n— (c)

a) Faz com que seja exibida a linha n.

b) Lista da linha n até linha k.

c) Lista da linha n até o fim do programa.

Exemplo: List 10—100

6) | NEW |

——— deleta o programa corrente na memória e apaga todas as variáveis.

7) | STOP |

——— finaliza a execução do programa e retorna ao nível de comando.

8) | RUN n |

——— **opções:** Run n (a)
Run (b)

a) Executa o programa em memória a partir da linha n.

b) Executa o programa do início.

9) | EDIT n |

——— onde n é o número de linha.

No modo de edição é possível editar partes de uma linha sem re-teclar a linha inteira. Ao entrar no modo edição, o BASIC emite o número da linha a ser editada, um espaço e espera pelo sub-comando do modo edição.

TeclaENTER: Grava as modificações feitas na linha corrente do programa e volta ao modo de comando.

n Espaço: No modo de edição, a cada toque na tecla espaçadora é listado um caracter. Digitando **n** e em seguida a tecla espaçadora, é listado **n** caracteres da linha.

n ——— : Desloca o cursor para esquerda por n espaços.

SHIFT : Sai do modo de inclusão.

L : Lista toda a linha do modo edit.

I : Insere caracteres.

A : Cancela alterações na linha editada.

nD : Deleta **n** caracteres.

nC : Substitui **n** caracteres.

MÓDULO 7 — TEORIA

INSTRUÇÃO PRINT

A instrução Print tem vários formatos.

Neste item veremos:

PRINT @
PRINT TAB
PRINT USING

PRINT @ EXP, ITEN(S) : Posiciona o cursor especificando onde a impressão deve começar. Pode ser usado apenas para localizar uma posição.

EXP: Equivale a uma expressão onde a técnica usada é:

64 **n da linha + coluna desejada**

pois o vídeo possui:

(0—63 64 (0—15) 16

Por exemplo: Para posicionar o cursor na **linha 5 coluna 10** —— PRINT @ 64 5+10,"";

Outro exemplo:

```
10 CLS
20 PRINT @ 64  2+10, "PROGRAMA-
   TESTE";
30 PRINT @ 64  5+10," ";
40 INPUT "EXECUTA(S/N)";R$
50 IF R$="S" THEN 100
60 IF R$="N" THEN 200
 •
 •
 •
 •
 •
```

PRINT TAB (EXP1)

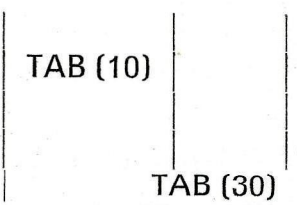
Finalidade: Mover o cursor para direita numa posição especificada na linha.

EXPI: Identifica a localização desejada.

Exemplo:

```
10 PRINT TAB (10) "TABULA 10"; TAB (30)
"TABULA 30"
```

Em cada item **TAB** a tabulação é feita a parda posição inicial (TAB) (1), ou seja, na linha acima o efeito é:



PRINT USING

Finalidade: Usado para formatar impressões de valores string ou numéricos. Para tal devem ser definidos os especificadores de campo (máscaras), que podem ser:

: —— Especifica cada digito do valor numérico.

O número de sinais que se utiliza, determina o tamanho do campo.

: —— **Ponto Decimal:** Pode ser usado em qualquer lugar do campo especificado por .

: —— Os valores digitados em português apresentam uma **vírgula-decimal.** Já os americanos editam um **ponto-decimal.**

Então o que seria para nós o ponto de milhar etc., nesta versão será uma vírgula. OBSERVE:

20.000,00 20,000.00

Há uma inversão de ponto para vírgula e vice-versa.
Uma vírgula colocada em qualquer posição entre o primeiro digito e o ponto decmial mostrará na tela uma vírgula à esquerda de todo terceiro digito.

Exemplo: PRINT USING " ; . ";
50000 50,000.00
READY
>

* * : Todas as posições em desuso serão preenchidas com asteriscos.

$: Será impresso à frente do número um cifrão.

$$: Fará o cifrão ser flutuante no campo, assumindo a posição anterior ao número.

**$: Posições em desuso com asterisco e a primeira posição anterior ao número com cifrão.

%Espaços%: Espaço para saídas strings. O total de posições é:

espaços + 2, equivalente os dois sinais.

MÓDULO 8 — EXERCÍCIOS

1) Fazer um programa que: (usar TAB em todos)
2) Dados um valor qualquer, este seja exibido formatado na tela. (usar ,$ e vírgula).
3) Calcule o valor do Iapas, dado o salári odo indivíduo .(Salário também com máscara de edição).

MÓDULO 9 — TEORIA

VARIÁVEIS NUMÉRICAS

1) **Dados Numéricos**

Podem ser classificados como:

— Numéricos Inteiros (a)
— Simples Precisão (b)
— Dupla Precisão (c)

Para os tipos citados temos as variáveis correspondentes:

— Variáveis Inteiras
— Variáveis Simples Precisão
— Variáveis Dupla Precisão

Quando se fala em dados numéricos é importante considerar sua **eficiência** e **precisão.**

a) **Números Inteiros:** Devem estar no intervalo de — 32.768 à + 32767. **Ocupação:** 2 bytes

Símbolo: %
Característica: Velocidade

e eficiência mas faixa numérica limitada.

Exemplos: 1) A% = 1000
2) PR% = — 50
3) R% = A% + PR%

b) **Variáveis de Simples Precisão:**

Podem incluir até 7 digitos significativos e podem representar valores em notação ceintífica com expoentes até ± 38.

Ocupação: 4 bytes

Símbolo: ! (pode ser omitido)

Característica: Faixa numérica completa

Exemplos: 1) A! = 54213.345

2) B = 1.774E6

3) C = 6.024E − 23

OBS.:

Quando usado em número decimal, o símbolo E representa **"precisão simples vezes 10 elevado a . . ."**

Então, 6.024E − 23 representa o valor precisão simples: 6.024×10^{-23}.

c) **Variáveis de Dupla Precisão:**

Envolvem números com até 17 digitos significativos e podem representar valores na mesma faixa usada para números de precisão simples.

Ocupação: 8 bytes

Símbolo:

Característica: Precisão máxima mas moroso em cálculos.

Exemplos: 1) A = 1010234578

2) B = −8.7777651010

3) C = 8.00100708D12

OBS.:

"**D**" nos números representa dupla precisão em notação científica.

Exemplos de variáveis numéricas

1) A! = 3000

2) A% = 3000

3) A= = 3000

Repare que apesar de todas as variáveis possuirem **A** como nome, os símbolos usados são diferentes. Logo, 1), 2) e 3) representam 3 variáveis distintas.

A! —— S.P

A% —— INT

A —— D.P

RESUMO

Variável	Símbolo	Ocupação	Número de Digitos
Inteira	%	2 bytes	até 5 no interválo − 32.768 à + 32.767
Simples Precisão	!	4 bytes	7 significativos
Dupla Precisão		8 bytes	17 significativos

2) DEFINT, DEFSNG e DEFDBL

a) **Defint Letral — Letra 2**

Define que todas as variáveis começando com a **"letra 1"** até **"letra 2"** serão variáveis **INT**eiras.

Exemplo: DEFINT A—C (A, B, C, são inteiras)

b) Mesma noção para DEF**SNG,** só que **SNG** irá designar variáveis **simples precisão.**

c) Idem, para **dupla precisão.**

OBS.:

Mesmo que definido DEFINT/SNG/DBL as variáveis terão o tipo a qual sejam declaradas nas linhas do programa. Ou seja:

```
10 CLS
20 DEFINT A – Z
30 INPUT A
40 B    = A * A
50 PRINT B
60 END
```

Na **linha 20** são definidas de **A** à **Z** como variáveis inteiras. Mas na **linha 40** tem o B , ou seja, é assumido **B** como variável de dupla precisão apesar da **linha 20.**

MÓDULO 10 — TEORIA

I — Instruções de **função-string**

1) **ASC (string)**

Fornece código ASCII para o **primeiro caracter** do string especificado. O argumento string deve ser colocado entre parênteses.

Exemplo:

1) PRINT ASC ("P")
2) P$ = "PROLÓGICA": PRINT ASC (P$)

As instruções 1) e 2) tem o mesmo efeito.

2) **CHR$ (EXP)**

Fornece um string de um caracter.

EXP pode ser qualquer número de 0 à 255.

Exemplo:

```
10 CLS
20 FOR T% = 1TO255
30 PRINT T% TAB (10) CHR$ (T%)
40 NEXT T%
50 END
```

Este programa irá listar os strings correspondentes a cada código.

3) **LEFT$ (string, n)**

Fornece os **n** primeiros caracteres de um string.

Exemplo:

```
10 A$ = "PROLÓGICA"

20 B$ = LEFT$ (A$,3)

30 PRINT B$
```

efeito

```
> RUN < ENTER
PRO
READY
>
```

Ou seja, será listado (instrução 30) os 3 primeiros caracteres de A$.

4) **MID$ (string, pos. inicial, quantidade)**

Fornece **"quantidade"** caracteres a partir da **"posição inicial". MID** vem de **meio.** Portanto dado uma string, podemos acessar u minter-valo de caracteres.

Exemplo:

```
10 AS = "PROLÓGICA"
20 BS = MID$ (A$, 4, 3)
30 PRINT B$
```


efeito

```
> RUN
  LOG
  READY
>
```

Observe que a partir (pos. inicial) do 4.° caracter (P) tomaremos 3 caracteres.

```
P R O L Ó G I C A
1 2 3 4 5 6 7 8 9
```

5) **RIGHT$ (string, n)**

Fornece os **n últimos** caracteres de uma string.

Exemplo:

```
10 A$ = "PROLÓGICA"
20 B$ = RIGHT$ (A$, 4)
30 PRINT B$
```


```
> RUN
  GICA
  READY
>
```

6) **STR$ (EXP)**

Converte uma expressão ou constante numérica em string.

Exemplo:

```
10 A = 58.75
20 B$ = STR$ (A)
30 PRINT B$
```

Em B$ temos 58.75 na forma de string.

7) **STRING$ (n, "caracter")**

Fornece um string formado por **n caracteres.**

Exemplo:

```
10 PRINT STRING$ (10, " * ")
   RUN
   * * * * * * * * * *
```

8) **VAL (string)**

Realiza a função inversa do STR$;

Fornece o número representado pelos caracteres em um argumento string.

Exemplo:

```
10 B$ = "30"
20 A = VAL (B$)
30 PRINT A
```



```
> RUN
  30
  READY
  >
```

9) **LEN (string)**

Fornece o número de caracteres de uma string.

Exemplo:

```
10 A$ = "NOME"
20 PRINT LEN (A$)
> RUN
 4
```

II — DEFSTR

Consiste na mesma idéia que DEFINT/SNG/DEL mas relacionando **strings.**

Exemplo:

```
10 DEFSTR A – O
20 INPUT "DIGITE O NOME"; A
30 INPUT "DIGITE ENDEREÇO"; B
 •
 •
 •
 •
 •
 •
```

As variáveis **A** e **B** são **STRINGS.**

MÓDULO 11

DATA / READ

DATA. Armazena dados tanto Strings como numéricos para se ter acesso aos dados usamos a instrução **READ** que associa o dado a uma variável correspondente.

Portanto READ/DATA se completam na função.

Dados strings devem ser separados por vírgulas.

Em cada instrução READ é lido em dado da lista (DATA). Na 1.ª leitura lê-se o 1.° dado; na 2.ª leitura o 2.° dado e assim por diante. Um READ se "esgota" quando não existem mais dados para serem acessados no DATA. Quando for necessário uma leitura dos dados deve-se usar a instrução **RESTORE** que reinicia o acesso a partir do 1.° ítem de dado (DATA).

```
10 DATA "HOJE", "E", "DIA", 20
20 READ A$, B$
30 PRINT A$, B$
40 READ C$, D$
50 PRINT C$, D$                HOJE "É"
60 RESTORE                     DIA 20
70 READ E$, E$                 HOJE "É"
80 PRINT E$, F$                READY
> RUN                          >
```

COMANDO DIM

Usado para dimensionar uma tabela (MATRIZ) deve ser especificado o n.° de elementos por dimensão da matriz.

EXEMPLOS:

(1) DIM A$ (5) define uma matriz unidimensional onde a variável (A$) "OCORRE" S vezes. Então

```
1       A$ (1)   10 CLS
                 20 PRINT "TESTE DIM"
2       A$ (2)   30 DIM A$ (5)
                 40 PORT% = 1 TO5
3       A$ (3)   50 INPUT "DIGITE NOME" A$(T%)
4       A$ (4)   60 NEXT T%
                 70 FOR B% = 1 TO 5
5       A$ (5)   80 PRINT A$ (B%)
                 90 NEXT B%
```

As matrizes podem assumir a quantidade de dimensões compatível com a área de memória disponível.

2 — **DIM A (4,3)** — Define uma matriz bidimensional. Aqui se verifica uma **dupla-ocorrência** (4 — 3) isto é, o a ocorre 3 vezes em cada uma das 4 ocorrências iniciais.

3 — **ON VARIÁVEL GOTO LINHA 1, LINHA 2, LINHA N**

Saltar para um dos vários números de linhas especificadas, dependendo do valor da variável.

EXEMPLO:

```
10 CLS
•
•
•
100 ON V% GOTO 150, 250, 350
•
•
•
150 PRINT "ROTINA — 1"
•
•
•
250 PRINT "ROTINA — 2"
•
•
•
350 PRINT "ROTINA — 3"
•
•
•
400 GOTO 10
```

4 — GOSUB — NÚMERO DA LINHA

O comando GOSUB é parecido com o GOTO. Esse comando serve para desvio do programa para subrotina especificada pelo número da linha e permanece nesta subrotina até que seja encontrada a instrução RETURN; Portanto no final de toda subrotina deve ter a instrução RETURN.

EXEMPLO:

```
05 CLS
10 DATA "TESTE", "OK"
20 PRINT "TESTE DE SUBROTINA"
30 GOSUB 60
40 PRINT A$, B$
50 END
60 READ A$, B$
70 RETURN
```