ARSONVAL FLEURY PEREIRA

LIVROS ÉRICA EDITORA LTDA.

ARSONVAL FLEURY PEREIRA

Engenheiro de Eletrônica, formado pelo I.T.A. - Instituto Tecnológico de Aeronáutica, em São José dos Campos, no ano de 1968. Fez Pós-Graduação em Administração de Empresas pela Fundação Getúlio Vargas, na área de Concentração em Marketing e Finanças.

Participou de cursos de aperfeiçoamento na área de Microcomputadores no exterior, tais como na BOING HIDROAIR (E.U.A.), na BURROUGHS (França) e, no Brasil pela SIEMENS DO BRASIL e PHILCO (área de Técnicas Digitais).

Trabalhou na VASP, no setor de Engenharia Aviônica; na TELEFUNKEN na área de Telecomunicações; na NORTRON ELETRÔNICA, no setor de Projetos de equipamentos aplicados na área de Energia Nuclear, Proteção Catódica e Testes de Ultrasom e Dosimetria de Irradiações e, na BURROUGHS ELETRÔNICA LTDA, como gerente de Engenharia de Produtos.

No magistério, é professor titular da cadeira de Técnicas Digitais, na Escola de Engenharia da FAAP e, atualmente, presta serviços de consultoria na área de microcomputadores a diversas empresas, entre elas ARNO S/A.

OUTROS LANÇAMENTOS

Teoria e Desenvolvimento de Projetos de Circuitos Eletrônicos

Diodos, Transistores de Junção, FET, MOS, UJT, LDR, NTC, PTC, SCR, Transformadores, Amplificadores Operacionais e suas aplicações em Projetos de Fontes de Alimentação, Amplificadores, Osciladores, Osciladores de Relaxação e outras.
7.a Edição, 580 páginas.
Autores: **Engos. Cipelli / Sandrini**

Teoria e Processo de Desenvolvimento em Eletrônica

Estudo e Associação de Bipolos Passivos e Ativos, Circuitos Ressonantes, Aparelhos de Medidas, Válvulas, Semicondutores, Leis de Kirchhoff, Teoremas de Thevenin e Norton, Osciloscópio e outros. 2.a Edição, 259 páginas.
Autor: **Eng.o Sidnei David**

Microprocessadores 8080 e 8085 - Hardware - Vol. 1

Memórias RAM, ROM, PROM e EPROM, o 8224, 8228, 8080, 8085, 8255 e 8253, suas aplicações e montagem de um microprocessador. 3.a Edição, 138 páginas
Autor: **Eng.o Antonio Carlos José Franceschini Visconti**

Microprocessadores 8080 e 8085 - Software - Vol. 2

Estudo das instruções dos microprocessadores 8080 e 8085, Fluxogramas, iniciação a programação e desenvolvimento de programas com a utilização dos microprocessadores 8080 e 8085. 3.a Edição, 202 páginas.
Autor: **Eng.o Antonio Carlos José Franceschini Visconti**

Elementos de Eletrônica Digital

Iniciação a Eletrônica Digital, Álgebra de Boole, Minimização de Funções Booleanas, Circuitos Contadores, Decodificadores, Multiplex, Demultiplex, Display, Registradores de Deslocamento, Desenvolvimento de Circuitos Lógicos, Circuitos Somadores/Subtratores e outros. 5.a Edição, 504 páginas.
Autores: **Capuano / Idoeta.**

ARSONVAL.F PEREIRA — Computadores Pessoais para

ARSONVAL FLEURY PEREIRA

LIVROS ÉRICA EDITORA LTDA.

ARSONVAL FLEURY PEREIRA

Engenheiro de Eletrônica, formado pelo I.T.A. - Instituto Tecnológico de Aeronáutica, em São José dos Campos, no ano de 1968. Fez Pós-Graduação em Administração de Empresas pela Fundação Getúlio Vargas, na área de Concentração em Marketing e Finanças.

Participou de cursos de aperfeiçoamento na área de Microcomputadores no exterior, tais como na BOING HIDROAIR (E.U.A.), na BURROUGHS (França) e, no Brasil pela SIEMENS DO BRASIL e PHILCO (área de Técnicas Digitais).

Trabalhou na VASP, no setor de Engenharia Aviônica; na TELEFUNKEN na área de Telecomunicações; na NORTRON ELETRÔNICA, no setor de Projetos de equipamentos aplicados na área de Energia Nuclear, Proteção Catódica e Testes de Ultrasom e Dosimetria de Irradiações e, na BURROUGHS ELETRÔNICA LTDA, como gerente de Engenharia de Produtos.

No magistério, é professor titular da cadeira de Técnicas Digitais, na Escola de Engenharia da FAAP e, atualmente, presta serviços de consultoria na área de microcomputadores a diversas empresas, entre elas ARNO S/A.

BASIC para Computadores Pessoais

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

P489b

Pereira, Arsonval Fleury, 1943-
BASIC para computadores pessoais / Arsonval Fleury Pereira. -- São Paulo : Érica, 1983.

Bibliografia.

1. BASIC (Linguagem de programação para computadores) 2. BASIC (Linguagem de programação para computadores) - Problemas, exercícios etc. 3. Microcomputadores I. Título.

17. CDD-651.8
18. -001.6424
18. -001.642
17. -651.8076
18. -001.6424076

83-1190

Índices para catálogo sistemático:

1. BASIC : Linguagem de programação : Computadores : Processamento de dados 651.8 (17.) 001.6424 (18.)
2. Computadores pessoais : BASIC : Linguagem de programação : Processamento de dados 651.8 (17.) 001.6424 (18.)
3. Exercícios : BASIC : Linguagem de programação : Computadores : Processamento de dados 651.8076 (17.) 001.6424076 (18.)
4. Microcomputadores : Programação : Processamento de dados 651.8 (17.) 001.642 (18.)

ARSONVAL FLEURY PEREIRA

BASIC para Computadores Pessoais

1983

LIVROS ÉRICA EDITORA LTDA.

Revisão:
GUARACIABA MICHELETTI
Composição dos Textos:
ROSELI BARBARESCO DE OLIVEIRA

A meus Pais,
a minha esposa,
a meus filhos.

LIVROS ÉRICA EDITORA LTDA.

Rua Jarinu, 594 - Tatuapé - São Paulo
Fone: 294-8686 - C.G.C. 50.268.838/0001-39
Caixa Postal 15.617

APRESENTAÇÃO

Este livro tem como objetivo expor a linguagem *BASIC* àqueles que estão iniciando seus passos no mundo da computação ou aos que, já havendo trilhado outros caminhos, se aproximam pela primeira vez das fronteiras da linguagem *BASIC*.

Na década de 60, um grupo de trabalho foi criado no Dartmouth College, sob a responsabilidade dos Professores John G. Kemeny e Thomas E. Kurtz, com a finalidade de desenvolver uma linguagem que atendesse ao objetivo de ser de uso geral, suficientemente simples e fácil de ser entendida para que pudesse ser utilizada por pessoas com pouco ou nenhum conhecimento de programação. O resultado foi uma linguagem denominada "Beginer's All-Purpose Symbolic Instruction Code"- *BASIC* a qual atende plenamente aos propósitos pré-estabelecidos.

Por sua real facilidade de ser aprendida e de ser usada sem apresentar limitações quanto ao campo de aplicação, a linguagem *BASIC* foi ganhando adeptos no mundo da informática, o que fez com que fosse adotada pelos fabricantes de computadores pessoais.

Esta adoção, aliada ao grande número de computadores pessoais vendidos, fez com que o número de pessoas envolvidas em programas *BASIC* se multiplicasse rapidamente, ocorrendo como resultado final um enriquecimento substancial da qualidade e quantidade de programas desenvolvidos, além da ampliação das áreas de aplicações.

A linguagem *BASIC* será aqui apresentada, de uma maneira "simples e fácil de ser entendida, para que possa ser compreendida por pessoas com pouco ou nenhum conhecimento de programação".

Arsonval Fleury Pereira.

SUMÁRIO

CAPÍTULO 1

PRIMEIROS PASSOS

BOAS VINDAS

Antes de tudo, minhas boas vindas ao mundo da computação pessoal. Um mundo onde errar é aceito como um estágio do aprendizado e não como uma demostração de incapacidade de aprender. Adote este princípio como verdadeiro e verá que as coisas serão bem mais simples do que você talvez esteja julgando.

Do mesmo modo que no nosso mundo real existem vários idiomas, várias são as linguagens no mundo da computação. Essas linguagens são formadas por determinadas palavras e por regras próprias de sintaxe. Conhecer uma linguagem de computador é conhecer o significado dessas palavras e o uso correto da sua sintaxe.

Das línguas faladas no mundo da computação pessoal, uma das mais difundidas é a linguagem *BASIC*. Sua popularidade deve-se ao fato de ser uma linguagem simples, clara e de fácil aprendizado sem no entanto ser de uso restrito ou de capacidade limitada. Tal condição faz com que ela seja utilizada tanto quanto por estudantes como por cientistas, por pessoas alheias à informática, como por profissionais da área.

O objetivo desse livro é fornecer os conhecimentos básicos necessários à utilização da linguagem *BASIC*. Nenhum conhecimento anterior específico na área é necessário, sendo imprescindível, entretanto, o acompanhamento e a realização dos exemplos e exercícios práticos.

Gostaríamos de incentivar a leitura deste livro, mesmo para as pessoas que não tenham um computador pessoal. Poderão estas ter um pouco de dificuldade em compreender o aqui exposto, mas tal problema será sanado tão logo tenham condições de colocar a "mão na massa".

Normalmente, entre os não iniciados em computação, existe o receio de que uma utilização errônea do computador possa causar danos ao mesmo. Este receio deve ser abandonado e deve-se adotar uma atitude de "descobrir tentando".

Se, durante a leitura do livro e do acompanhamento de um exemplo ou da realização de um exercício, você for assaltado pela curiosidade de saber "... e se ao invés disso eu fizer ...;" vá em frente, teste sua idéia e se o resultado não for o esperado procure descobrir o "porquê".

Qualquer que seja o computador pessoal que você esteja usando, sempre haverá, além do computador propriamente dito, um teclado, normalmente parecido ao de máquina de escrever, e um vídeo que pode ser um televisor ou um monitor de vídeo. Estes dois elementos é que vão possibilitar sua comunicação com o computador.

O teclado obviamente é de uso exclusivo seu, (ainda não inventaram um computador com braços e mãos capaz de também usar o teclado) enquanto que o vídeo é de uso seu e do computador. Para você, o vídeo substitui o papel da máquina de escrever. Qualquer letra ou número que você pressionar no teclado, aparecerá no vídeo. A posição é indicada pelo cursor o qual é um quadrado ou um traço que fica "piscando" no vídeo. Assim que você pressionar uma tecla, o computador envia para o vídeo o carácter correspondente e muda o cursor para a posição a direita, indicando a posição onde o próximo carácter entrará.

Entretanto, o computador só tomara conhecimento do que você digitou, quando você pressionar a tecla *"RETURN"* (*CR* ou *CARRIAGE RETURN* em alguns teclados). Neste instante, o computador irá "ler" o que foi digitado e sendo alguma coisa que compreenda, ou seja, faça parte do vocabulário da língua *BASIC* ele aceitará, caso contrário, ele se utilizará do vídeo para lhe enviar uma mensagem do tipo *SINTAX ERROR* ou seja, erro de sintax.

MÃOS A OBRA

Para iniciarmos o nosso aprendizado da linguagem *BASIC*, é necessário que estejamos com o computador ligado e com o *BASIC* carregado. Alguns computadores pessoais, uma vez ligados, já estão, automaticamente, com a linguagem *BASIC* disponível; outros necessitam que a mesma seja carregada. Se você estiver em dúvida, consulte o manual de seu computador. Não se

preocupe, utilize o tempo que for necessário. Não irei embora.

Tudo pronto ? Ótimo. Então vamos lá.

SEU PRIMEIRO PROGRAMA EM BASIC

Antes de mais nada, você deve limpar a área da memória do computador reservada para guardar programa. Esta limpeza prévia é necessária sempre que você for iniciar um novo programa para evitar que algum programa, que por acaso já esteja na área da memória, fique misturado com o novo programa.

Nada de água e sabão para limpar a memória do computador. Para isso existe a palavra de comando

NEW

O computador foi programado para que, tão logo você envie a palavra *NEW*, apague da área da memória reservada para guardar programa, qualquer programa lá armazenado e se prepare para receber um novo programa. Esta limpeza feita pelo computador é automática e imediata, independente de qualquer outra ação sua e, além disso, esta limpeza é definitiva. Não existe modo de se recuperar o programa apagado. Após ter enviado o comando *NEW* a memória estará limpa e se você quiser utilizar novamente o mesmo programa, o único recurso é digitá-lo por inteiro outra vez. Portanto tenha certeza de que você quer realmente apagar um programa antes de enviar o comando *NEW*.

Como no nosso caso estamos conscientes de que desejamos limpar a memória; tecle:

NEW

Não se esqueça de pressionar a tecla marcada *RETURN* para que o computador tome conhecimento do *NEW* que você teclou.

Isto feito a memória de programa estará limpa e você poderá iniciar a escrever o seu primeiro programa.

Para isso, tecle:

10 PRINT "MEU PRIMEIRO PROGRAMA"

Não se esqueça de teclar *RETURN* e se, após teclar

RETURN, você receber a mensagem

```
SINTAX ERROR
```

isto significa que o computador não entendeu o que foi enviado. Procure descobrir qual foi o erro que você cometeu. Algumas dicas são: ter teclado a letra "l" no lugar do algarismo "1", a letra "O" no lugar do algarismo "0", ou ter se esquecido das aspas. De qualquer modo não se preocupe, verifique o erro cometido e tecle novamente a linha.

```
10 PRINT "MEU PRIMEIRO PROGRAMA"
```

Se você não recebeu a mensagem de erro, aparentemente nada ocorreu. O cursor passou para a linha seguinte e isto parece ser tudo. O que realmente ocorreu é que a linha que você teclou foi armazenada na área da memória reservada para programa e o computador esta aguardando a sua próxima ação.

Digite agora mais uma linha do seu programa,

```
1000 END
```

Esta linha indica o final do programa. Para algumas versões de *BASIC*, a colocação de *END* no final do programa, é facultativo, mas para fazer com que o programa possa ser utilizado por qualquer computador, você deve incluí-la.

Você tem agora um programa armazenado na memória do computador pronto para ser usado. Do mesmo modo que para limpar a memória do computador foi utilizado um comando, neste caso também deve ser enviada uma palavra de comando para que o computador inicie a execução do programa.

A palavra de comando para iniciar a execução de um programa é

```
RUN
```

Tecle, portanto, *RUN* e não se esqueça de pressionar *RETURN*. Tão logo faça isso, o computador executará o programa ou, como se diz na gíria dos já iniciados em computação, irá "rodar" o programa. Como resultado você terá no seu vídeo:


```
RUN
MEU PRIMEIRO PROGRAMA
```

Meus parabéns. Você acaba de rodar o seu primeiro programa. Realmente não é um programa para sair por aí se gabando, mas ninguém pode dizer que não seja um programa. Se no entanto você receber uma mensagem de erro indicando uma linha do programa, isto significa que na hora de rodar o programa, o computador descobriu um erro anteriormente não detectado. Corrija a linha e torne a rodar o programa até obter sucesso.

Mesmo depois de ter rodado, o programa continua intacto na memória do computador. Verifique por você mesmo. Tecle novamente *RUN* e você terá mais uma vez, na tela, a mensagem

```
MEU PRIMEIRO PROGRAMA
```

COMANDO LIST

O programa armazenado na memória do computador pode ser mostrado na tela do vídeo através da palavra de comando *LIST*. Esta listagem do programa é uma cópia exata do programa armazenado na memória, e não interfere absolutamente em nada no mesmo. Após a listagem o programa permanece na memória exatamente como antes. Portanto, sempre que você quiser ler o programa contido na memória, envie *LIST*.

Uma vez conhecido o comando *LIST*, vamos utilizá-lo.

Tecle *LIST*, pressione *RETURN* e teremos no vídeo:

```
LIST

10 PRINT "MEU PRIMEIRO PROGRAMA"
1000 END
```

PALAVRA DE COMANDO E INSTRUÇÃO

Como você deve ter notado, o computador operou de dois modos distintos:

- *DIRETO* ou *IMEDIATO*: Executou imediatamente uma determinada ação ditada por uma palavra de comando. Por exemplo: limpou a memó

ria com *NEW*, rodou imediatamente o programa todas as vezes que você enviou *RUN* e listou o programa com *LIST*.

- *INDIRETO* ou *PROGRAMADO*: Neste modo de operação, o computador executou instruções que estavam armazenadas em sua memória, obedecendo ao comando *RUN*.

O computador, uma vez ligado e carregado com a linguagem *BASIC*, está no modo direto. Para utilizá-lo no modo programado, você deve escrever um programa em *BASIC* utilizando para tal um conjunto de palavras reservadas denominadas instruções e, através do comando *RUN*, fazê-lo rodar o programa.

Você poderá estar pensando "Como indico para o computador que eu vou escrever um programa?"

A resposta é bem simples. Sempre que você enviar para o computador uma linha numerada, ou seja, iniciada por um número, o computador adotará essa linha como a primeira linha do programa, se ainda não existir um programa; ou, como mais uma linha do programa, caso já exista algum programa na memória.

ANALISANDO O PRIMEIRO PROGRAMA

Liste o programa que está na memória do computador. Se você não enviou outras linhas, deveremos ter na tela:

```
LIST

10 PRINT "MEU PRIMEIRO PROGRAMA"
1000 END
```

(Se esse não for o caso, por favor, limpe a memória com *NEW* e reentre o programa.

NUMERAÇÃO DA LINHA

Para numerar as linhas de seu programa, você pode usar qualquer número inteiro, a partir de 1. O limite máximo depende da versão de *BASIC* de seu computador, podendo ser 99999, 65529, 32767 ou 9999. Vamos escolher o menor dentre esses ou seja 9999, com isto, qualquer que seja a versão de *BASIC* de seu computador, tudo estará bem.

Quando o computador recebe a palavra de comando *RUN*, inicia a execução das instruções constantes do programa na sequência numérica das linhas. Assim, neste caso, ele executa primeiro a instrução da linha 10 e depois a linha de numeração seguinte, ou seja, a linha 1000. O fato de existirem linhas "saltadas" não altera a operação do computador, para ele tanto faz se o número da linha seguinte é consecutivo ou não. A vantagem de se saltar linhas é que novas linhas podem ser acrescentadas ao seu programa sem que seja necessário reescrevê-lo.

ACRESCENTAR LINHAS A UM PROGRAMA NA MEMÓRIA

Para acrescentar novas linhas a um programa na memória do computador basta simplesmente digitar a nova linha e enviá-la (ou seja pressionar *RETURN*).

Fiéis ao lema "Aprender fazendo", vamos acrescentar uma nova linha ao nosso programa. Assim, envie a linha abaixo:

```
20 PRINT "AGORA MODIFICADO"
```

Não se esqueça do RETURN após ter digitado a linha. Este é o último lembrete de que você tem que pressionar a tecla *RETURN* para enviar a linha.

Para verificar como essa nova linha foi colocada na memória do computador, use o comando *LIST* ou seja liste o programa.

```
LIST

10 PRINT "MEU PRIMEIRO PROGRAMA"
20 PRINT "AGORA MODIFICADO"
1000 END
```

O computador automaticamente colocou a linha número 20 na sequência certa, embora ela tenha sido enviada posteriormente. Rode agora o programa e teremos como resposta:

(Lembre-se de que para rodar o programa, em outras palavras, ordenar ao computador que execute o programa armazenado na memória, a palavra de comando é *RUN*).

```
MEU PRIMEIRO PROGRAMA
AGORA MODIFICADO
```

Vamos testar, mais uma vez, a ação automática do computador em ordenar adequadamente as linhas do programa.

Acrescente a linha abaixo:

```
15 PRINT "ESTA"
```

Rodando o programa, teremos:

```
RUN

MEU PRIMEIRO PROGRAMA
ESTA
AGORA MODIFICADO
```

e se for listado, teremos:

```
LIST

10 PRINT "MEU PRIMEIRO PROGRAMA"
15 PRINT "ESTA"
20 PRINT "AGORA MODIFICADO"
1000 END
```

O resultado foi o que esperávamos, o computador colocou a nova linha, de número 15, entre as linhas 10 e 20 de nosso programa.

MODIFICAR UMA LINHA

O que aconteceria se fosse enviada uma linha com o mesmo número de uma já existente ? Neste caso o computador apaga a linha anterior e coloca na memória esta última linha. Assim, para mudar uma linha de um programa *BASIC* é bastante simples. Basta enviar uma nova linha com o mesmo número da linha a ser modificada. Seja como exemplo, o caso de modificar o seu programa de:

```
MEU PRIMEIRO PROGRAMA
ESTA
AGORA MODIFICADO
```


Para

```
MEU PRIMEIRO PROGRAMA
SERA
AGORA MODIFICADO
```

A linha do programa responsável pela palavra ESTÁ é a linha 15, donde para modificá-la, tecle uma nova linha 15

```
15 PRINT "SERA"
```

Listando o programa, temos:

```
LIST

10 PRINT "MEU PRIMEIRO PROGRAMA"
15 PRINT "SERA"
20 PRINT "AGORA MODIFICADO"
1000 END
```

Rodando o programa temos:

```
RUN

MEU PRIMEIRO PROGRAMA
SERA
AGORA MODIFICADO
```

SUPRIMIR UMA LINHA

Além da possibilidade de acrescentar e modificar linhas de seu programa, a linguagem *BASIC* também permite que uma linha do seu programa seja suprimida.

Para retirar uma linha do programa, o procedimento é muito simples. Basta enviar somente o número da linha a ser suprimida. Como exemplo, vamos suprimir a linha 15 de nosso programa. Para isto, envie, simplesmente, o número 15. Ao receber o número 15, o computador irá substituir a linha 15 anterior por uma nova linha 15 "vazia" ou seja sem instruções. Como não existe nada a ser executado pelo computador numa linha vazia, esta linha não será listada e deste modo, "desaparecerá" do programa.

Liste o programa. O resultado será:

```
LIST

10 PRINT "MEU PRIMEIRO PROGRAMA"
20 PRINT "AGORA MODIFICADO"
1000 END
```

Rodando

```
RUN

MEU PRIMEIRO PROGRAMA
AGORA MODIFICADO
```

O resultado foi o esperado: a linha de número 15 foi suprimida do programa.

LISTAGEM PARCIAL

O comando *LIST* pode ser usado para listar somente uma linha ou então uma parte do seu programa. Para listar uma linha, acrescente após o comando, o número da linha, por exemplo:

```
LIST 20

20 PRINT "AGORA MODIFICADO"
```

Para listar uma parte do programa, acrescente após o comando, o número da primeira linha a ser listado, uma vírgula, e o número da última linha a ser listada.

```
LIST 10, 20

10 PRINT "MEU PRIMEIRO PROGRAMA"
20 PRINT "AGORA MODIFICADO"
```

RODANDO O PROGRAMA PARCIALMENTE

Você pode iniciar a execução do programa a partir de qualquer uma das linhas do mesmo, acrescentando após o comando *RUN* o número da linha escolhida.


```
RUN 20

AGORA MODIFICADO
```

Esta opção de uso do comando RUN será bastante útil quando você tiver que corrigir erros em programas extensos. No entanto, diferentemente da listagem parcial, você não pode definir uma linha para que o computador interrompa a execução do programa. O programa será "rodado" a partir da linha escolhida por você até o final do mesmo.

EXERCÍCIOS

EXERCÍCIO 1.1

Considerando que você escreva o programa abaixo, responda as seguintes questões:

a.) O programa listado aparecerá na mesma ordem que foi escrito ? Por quê ?

b.) Qual será a seqüência de execução desse programa ?

c.) Qual será o resultado dessa execução ?

Comprove suas respostas com o resultado obtido praticamente.

```
20 PRINT "DOIS"
40 END
30 PRINT "FEIJAO COM ARROZ"
10 PRINT "UM"
```

EXERCÍCIO 1.2

Mude a instrução END da linha 40 para a linha 90. Comprove o resultado, listando o programa.

EXERCÍCIO 1.3

Acrescente as seguintes linhas ao programa:

```
40 PRINT "TRES"
50 PRINT "QUATRO"
60 PRINT "FAZER O PRATO"
```

Qual o resultado de se rodar o programa agora ?

EXERCÍCIO 1.4

Entre com o novo programa abaixo e rode-o

```
100 PRINT "CINCO"
200 PRINT "SEIS"
300 PRINT "OUTRA VEZ"
400 END
```

Se o resultado não foi o esperado, procure descobrir o "porquê".

Observação:

As respostas a esses exercícios e aos dos demais capítulos estão no apêndice A.4.

CAPÍTULO 2

ESCREVENDO NO VÍDEO

INTRODUÇÃO

No capítulo anterior você ficou sabendo que um programa é uma seqüência de linhas numeradas as quais contêm as instruções a serem executadas. Uma vez conhecedor da estrutura de um programa, o passo seguinte é aprender as instruções da linguagem *BASIC*. Este capítulo e os próximos serão dedicados à apresentação das instruções da linguagem *BASIC*. Em cada capítulo, após a apresentação de cada instrução, seguem-se exemplos de aplicação dos mesmos e o resultado obtido, rodando-se esses exemplos. Você deve limpar a memória do seu computador, entrar com os exemplos, rodá-los e comparar os resultados obtidos com o seu computador, com os aqui apresentados. Nos casos em que a entrada de dados for necessária, entre com os mesmos dados do exemplo, para permitir comparações. Após isso, sinta-se à vontade para entrar com os dados que quiser.

O VÍDEO

O principal e, em alguns casos, o único sistema de saída de dados de um computador pessoal é o VÍDEO. É através dele que todas as mensagens, resultados,gráficos, etc., enfim toda a comunicação do computador ao usuário, é feita.

Embora a comunicação no sentido inverso (usuário-computador) seja feita via teclado, a mesma também é mostrada no VÍDEO, para que o usuário possa visualizá-la e, desse modo, corrigi-la, se necessário.

A INSTRUÇÃO PRINT

Todo o trabalho de envio das informações para o VÍDEO é feito automaticamente pelo computador, inclusive as informações, resultados ou mensagens geradas em um programa.

Para que esse trabalho seja executado pelo computador, o programa deve conter instruções adequadas para o envio dessas informações, o que é feito através da instrução *PRINT*.

O formato da instrução *PRINT*, isto é, o modo que você deve escrevê-la em um programa é:

número da linha *PRINT* (complemento)

número da linha - é o número da linha do programa, onde esta instrução será escrita.

PRINT - é a palavra chave da instrução (IMPRIMIR, em Inglês).

Complemento - é aquilo que você quer que seja enviado para o vídeo.

A operação da instrução *PRINT* variará de acordo com o conteúdo do complemento. Como complemento podemos ter uma frase, um número, um valor numérico, uma operação, uma variável, cada um destes isolados ou agrupados numa mesma instrução. Vamos analisar neste capítulo, a operação da instrução *PRINT* para cada um desses complementos, exceto quando o complemento for uma variável, o que será visto no próximo capítulo.

COMPLEMENTO ENTRE ASPAS — "STRINGS"

Você utilizou no capítulo 1, a instrução *PRINT* seguida de uma ou mais palavras que estavam entre aspas. O conteúdo desse complemento entre aspas é chamado de *"STRING"* e é enviado para o vídeo, exatamente como foi escrito, tenha significado ou não, seja letra, número, espaço, sinal ortográfico, etc.

No nosso já "histórico" primeiro programa, você escreveu na linha 10:

```
10 PRINT "MEU PRIMEIRO PROGRAMA"
```

e quando o programa foi rodado, o computador reconheceu a instrução *PRINT* e "escreveu" no vídeo a frase (*STRING*) que estava entre aspas, resultado no vídeo:


```
MEU PRIMEIRO PROGRAMA.
```

O computador trata uma *"STRING"* como um conjunto de dados "fechado", sem avaliar ou analisar o conteúdo do mesmo e, como conseqüência disso, você pode utilizar numa *"STRING"* palavras ou frases em Português ou em qualquer outra língua, ou até mesmo um conjunto de letras, números e sinais ortográficos, sem qualquer significado, os quais serão escritos exatamente como estão pelo computador, no vídeo. No entanto, você não pode colocar aspas dentro de uma *"STRING"* pois como as aspas indicam o seu início e o seu término, uma aspa colocada no meio de uma *"STRING"* seria identificada como final da mesma, gerando desse modo, um erro de interpretação pelo computador.

Exemplo 2.1

```
10 PRINT "UMA FRASE NORMAL"
20 PRINT "F R A S E  E S P A C E J A D A"
30 PRINT "MISTURA - 1 A * ? ABACATE 1.236"
40 PRINT "           NESTA EXISTEM 11 ESPACOS FRONTAIS"
50 END
```

Rode o exemplo

```
RUN

UMA FRASE NORMAL
F R A S E  E S P A C E J A D A
MISTURA - 1 A * ? ABACATE 1.236
          NESTA EXISTEM 11 ESPACOS FRONTAIS
```

Todas as *"STRINGS"* ou seja, o conteúdo entre as aspas foram enviadas para o vídeo exatamente como estavam escritas nas instruções *PRINT*.

Exemplo 2.2

```
10 PRINT "  V"
20 PRINT "  E"
30 PRINT "HORIZONTAL"
40 PRINT "  T"
50 PRINT "  I"
60 PRINT "  C"
70 PRINT "  A"
80 PRINT "  L"
90 END

RUN

  V
  E
HORIZONTAL
  T
  I
  C
  A
  L
```

Este exemplo visou tão somente mexer um pouco com sua criatividade.

PRINT SEM COMPLEMENTO

Você pode escrever a instrução *PRINT* sem complemento. Neste caso, uma linha em branco é enviada para o vídeo. Se utilizada entre duas instruções de *PRINT* com complemento, o resultado é uma linha de vídeo em branco (sem caracteres) equivalendo a um espaço duplo entre as linhas.

Exemplo 2.3

```
10 PRINT "ESPACEJAMENTO ENTRE LINHAS"
20 PRINT "NORMAL"
30 PRINT
40 PRINT "DUPLO"
50 PRINT
60 PRINT
70 PRINT "TRIPLO"
80 PRINT
90 PRINT
100 PRINT
110 PRINT "QUADRUPLO"
120 END
```



```
ESPACEJAMENTO ENTRE LINHAS
NORMAL
          espaço criado pela instrução PRINT da linha 30
DUPLO

          espaço criado pela instrução PRINT da linha 50
          espaço criado pela instrução PRINT da linha 60
TRIPLO
          espaço criado pela instrução PRINT da linha 80
          espaço criado pela instrução PRINT da linha 90
          espaço criado pela instrução PRINT da linha 100
QUADRUPLO
```

Cada instrução *PRINT* sem complemento enviou para o vídeo, uma linha "vazia" criando, assim, espaço entre as linhas.

PONTO E VÍRGULA APÓS O COMPLEMENTO

Se após o complemento de uma instrução *PRINT*, for colocado um *PONTO E VÍRGULA*, a próxima instrução de *PRINT* escreverá no vídeo, não no começo da linha seguinte, mas na mesma linha, exatamente após o último caractere escrito anteriormente.

Exemplo 2.4

```
10 PRINT "MINHA TERRA";
20 PRINT "TEM PALMEIRAS"
30 END

RUN

MINHA TERRATEM PALMEIRAS
```

As palavras *TERRA* e *TEM* ficaram "grudadas" uma na outra. Isto ocorreu porque o *PONTO E VÍRGULA* no final da instrução *PRINT* da linha 10 fez com que a *"STRING" TEM PALMEIRAS* se iniciasse imediatamente após o último caractere da primeira *"STRING"*, ligando deste modo, as palavras *TERRA* e *TEM*.

Como o programa não executou aquilo que esperávamos, evidentemente, o mesmo está errado. Na "linguagem" dos já iniciados, diz-se que o programa tem um *"BUG"*. Um *"BUG"* num programa é qualquer tipo de erro no mesmo. Tirar esse *"BUG"*, ou

seja, corrigir o erro, é bem simples. Um modo possível é colocar um espaço no final de *MINHA TERRA*, antes das aspas.

Mudando a linha 10 do programa temos:

Exemplo 2.5

```
10 PRINT "MINHA TERRA "; note o espaço após TERRA.
20 PRINT "TEM PALMEIRAS"
30 END
```

Rodando temos:

```
RUN

MINHA TERRA TEM PALMEIRAS
```

e agora o nosso programa está correto.

COMPLEMENTO SEM ASPAS

Como vimos na instrução *PRINT*, quando o complemento é uma *"STRING"*, o computador não avalia e nem analisa aquilo que esta entre aspas. Simplesmente se limita a enviar uma cópia exata para o vídeo.

Existe, entretanto, um formato da instrução *PRINT* onde o complemento é escrito sem aspas, não se tratando, portanto de uma *"STRING"*. A operação do computador, neste caso, é totalmente diferente.

O complemento neste caso deve ser obrigatoriamente um número ou uma expressão numerica ou uma variável ou expressão numérica ou alfanumérica. Nós vamos tratar, agora, tão somente das duas primeiras possibilidades ou seja, quando o complemento for um número ou uma expressão numérica. As outras possibilidades serão tratadas no próximo capítulo, quando estudaremos variáveis.

NÚMERO OU EXPRESSÃO NUMÉRICA COMO COMPLEMENTO

Quando o complemento for um número ou uma expressão numerica, a operação da instrução *PRINT* é bem simples. Se o

complemento for um número, esse número será enviado para o vídeo; sendo uma expressão numérica, o resultado dessa expressão será enviado para o vídeo.

Exemplo 2.6

```
10 PRINT 100
20 PRINT -200
30 PRINT 33+12
40 PRINT 55-66
50 END
```

Rodando o exemplo 2.6, você terá:

```
RUN

100
-200
45
-11
```

A operação do programa linha a linha é:

10 - Envia para o vídeo, o número 100.

20 - Envia para o vídeo o número -200. Como, neste caso, o número é negativo, o sinal é escrito.

30 - A expressão aritmética 33 + 12 é calculada e o resultado 45 é enviado para o vídeo. Note que, neste caso, o sinal + não é escrito.

40 - A expressão aritmética 55 - 66 é calculada e o resultado - 11 é enviado para o vídeo. O resultado da operação sendo negativo, temos o sinal - à esquerda do número.

Em resumo, este formato de instrução *PRINT*, permite que você escreva num programa uma expressão numérica a qual será calculada pelo computador e cujo resultado será enviado para o vídeo.

MENSAGENS ENRIQUECEM UM PROGRAMA

Em pouco tempo, você estará escrevendo programas não só para o seu uso próprio, como também para terceiros. Um cui

dado que você deve tomar ao escrever esses programas, é incluir mensagens que permitam ao usuário saber a finalidade desses programas, como entrar dados, como interpretar os resultados obtidos, etc.

Entre com o exemplo 2.7 e rode-o

Exemplo 2.7

```
100 PRINT 3 + 5
200 PRINT 6 - 12
300 END

RUN

8
-6
```

O resultado obtido está perfeitamente correto, mas só possui significado para nós, porque conhecemos o programa e esperávamos este resultado. Para um observador estranho ao programa, nada significam aqueles números no vídeo.

Vamos acrescentar algumas linhas a esse programa de modo à torná-lo mais inteligível.

```
10 PRINT "ESTUDO DA INSTRUCAO PRINT"
20 PRINT
30 PRINT "EXPRESSAO ARITMETICA COMO COMPLEMENTO"
40 PRINT
50 PRINT "3 + 5=";
110 PRINT
120 PRINT "6 - 12=";
```

Após entrar essas linhas, listando o programa, você terá:

```
LIST
```


Exemplo 2.8

```
10 PRINT "ESTUDO DA INSTRUCAO PRINT"
20 PRINT
30 PRINT "EXPRESSAO ARITMETICA COMO COMPLEMENTO"
40 PRINT
50 PRINT "3 + 5=";
100 PRINT 3 + 5
110 PRINT
120 PRINT "6 - 12=";
200 PRINT 6 - 12
300 END
```

Rodando este exemplo, você terá:

```
RUN

ESTUDO DA INSTRUCAO PRINT

EXPRESSAO ARITMETICA COMO COMPLEMENTO

3 + 5=8

6 - 12=-6
```

Vamos analisar o programa linha a linha:

10 - Envia para o vídeo a *"STRING" ESTUDO DA INSTRUÇÃO PRINT*, informando, deste modo, a finalidade do programa.

20 - Envia uma linha em branco, para separação das linhas.

30 - Envia a *"STRING" EXPRESSÃO ARITMÉTICA COMO COMPLEMENTO*, completando a informação sobre a finalidade do programa.

40 - Envia uma linha em branco.

50 - Envia a *"STRING"* 3 + 5 =

100 - Calcula a expressão 3 + 5 e envia o resultado para o vídeo. Como a instrução *PRINT* anterior (linha 50) termina em ponto e vírgula, esse resultado é justaposto a "STRING" 3 + 5 =, sendo, portanto, escrito na mesma linha.

110 - Envia uma linha em branco.

120 - Envia a *"STRING"* 6 - 12 =.

200 - Calcula a expressão 6 - 12 e envia o resultado para o vídeo. Aqui também, esse resultado será escrito imediatamente após a *"STRING"* 6 - 12 =, devido ao ponto e vírgula no final da linha 120.

300 - Linha final do programa.

É bastante importante que você distingua, neste exemplo, as diferenças de operação da instrução PRINT nas linhas:

```
50 PRINT"3 + 5=";
100 PRINT 3 + 5
120 PRINT "6 - 12=";
200 PRINT 6 - 12
```

e, por isso, vamos frisar, mais uma vez, as diferenças. Nas instruções *PRINT* das linhas 50 e 120, as expressões, por estarem entre aspas, são "STRINGS", e, deste modo, são enviadas "sem tirar, nem por" para o vídeo. Nas linhas 100 e 200, temos como complemento, as mesmas expressões anteriores, mas, como não estão entre aspas, são calculadas pelo computador e os respectivos resultados são enviados para o vídeo.

USANDO DOIS OU MAIS COMPLEMENTOS

A instrução *PRINT* aceita a utilização de dois ou mais complementos em uma mesma instrução, desde que estes complementos sejam separados, entre si, por um *PONTO E VÍRGULA*. Tal possibilidade economiza trabalho, linhas do programa e também área ocupada na memória.

Exemplo 2.9

```
10 PRINT "DOIS COMPLEMENTOS ";"NUMA MESMA LINHA "
20 PRINT "SOMANDO 13 COM 23 RESULTA "; 13 + 23
30 END

RUN

DOIS COMPLEMENTOS NUMA MESMA LINHA
SOMANDO 13 COM 23 RESULTA 36
```


OUTRAS OPERAÇÕES MATEMÁTICAS

Nos exemplos que apresentamos até agora, dois tipos de operações matemáticas foram utilizadas: *SOMA* e *SUBTRAÇÃO*. Além destas, a linguagem *BASIC* pode efetuar as seguintes operações matemáticas:

PRODUTO, DIVISÃO, POTENCIAÇÃO, EXPONÊNCIAÇÃO, RAIZ QUADRADA, LOGARITMO, SENO, COSSENO, TANGENTE E ARCOTANGENTE.

Já vimos como utilizar as operações de *SOMA* e *SUBTRAÇÃO*. Vamos tratar agora das demais:

PRODUTO, DIVISÃO E POTENCIAÇÃO utilizam sinais gráficos para indicação das operações, que são:

PRODUTO * (asterístico) 3 * 2 (3 vezes 2).

DIVISÃO / (barra) 15 /2 (15 dividido por 2).

POTENCIAÇÃO ^ (circunflexo) 2 ^3 (2 elevado a 3).

As outras funções utilizam uma abreviação do nome da função em Inglês, seguida do valor escrito entre parênteses.

EXPONÊNCIAÇÃO	*EXP (30)*	e elevado a 30 (e=nº neperiano)
RAIZ QUADRADA	*SQR (14)*	raiz quadrada de 14
LOGARÍTMO	*LOG (2)*	logarítmo natural de 2
SENO	*SIN (3.14)*	seno de 3.14 radianos
COSSENO	*COS (6.28)*	cosseno de 6.28 radianos
TANGENTE	*TAN (2.08)*	tangente de 2.08 radianos
ARCOTANGENTE	*ATN (323)*	ângulo em radianos da tangente 323

Exemplo 2.10

```
10 PRINT "OPERACOES MATEMATICAS EM BASIC"
20 PRINT
30 PRINT "16 + 3= "; 16 + 3
40 PRINT "122-12= "; 122 - 12
50 PRINT "5 VEZES 2= "; 5 * 2
60 PRINT "10 DIVIDIDO POR 5= "; 10 / 5
70 PRINT "4 ELEVADO A 3= "; 4 ^ 3
80 PRINT "EXPONENCIAL DE 2= "; EXP (2)
90 PRINT "RAIZ QUADRADA DE 10= "; SQR (10)
100 PRINT "SENO DE 6 RAD= "; SIN (6)
110 PRINT "COSSENO DE 1.5 RAD= "; COS (1.5)
120 PRINT "TANGENTE 0.2 RAD= "; TAN (.2)
130 PRINT "ARCOTANGENTE DE 136= "; ATN (136)
140 END
```

RUN

```
OPERACAO MATEMATICA EM BASIC

16 + 3= 19
122 - 12= 110
5 VEZES 2= 10
10 DIVIDIDO POR 5= 2
4 ELEVADO A 3= 64
EXPONENCIAL DE 2= 7.3890561
RAIZ QUADRADA DE 10= 3.16227766
SENO DE 6 RAD= -.279415497
COSSENO DE 1.5 RAD= .0707372015
TANGENTE DE 0.2 RAD= .202710036
ARCOTANGENTE DE 136= 1.56344352
```

ESCREVENDO NO VÍDEO EM COLUNAS

Nós vimos dois modos de "escrever" no vídeo utilizando a instrução *PRINT*, linha a linha e concatenada. Um terceiro modo existe e consiste em organizar o vídeo em colunas. O número de colunas e a quantidade de caracteres em cada coluna é característica particular a cada modelo de computador. Nossos exemplos estão sendo rodados num computador pessoal que divide o vídeo em 3 colunas, sendo as duas colunas à esquerda com 16 caracteres e a coluna da direita com 8 caracteres.

Para "escrever" em colunas em *BASIC*, basta colocar uma *VÍRGULA* após o complemento, do mesmo modo que se usa *PONTO E VÍRGULA* para a escrita concatenada.

Exemplo 2.11

```
10 PRINT "ESCREVENDO","EM","COLUNAS"
20 PRINT "A","VIRGULA","APOS"
30 PRINT "O COMPLEMENTO","ESCREVE","EM","COLUNAS"
40 END
```

RUN

```
ESCREVENDO      EM        COLUNAS

A               VIRGULA   APOS

O COMPLEMENTO   ESCREVE   EM

COLUNAS
```

RELEMBRANDO NOÇÕES JÁ DEFINIDAS

No capítulo anterior, nós estabelecemos a noção de *"PALAVRA DE COMANDO"* e *"INSTRUÇÃO"* em *BASIC* e tendo em mente aqueles mais esquecidos, vamos relembrar.

Uma *PALAVRA DE COMANDO* em *BASIC* é escrita sem número de linha, não faz parte do programa e é executada pelo computador imediatamente.

Exemplo:

RUN, LIST, NEW.

Uma *INSTRUÇÃO* em *BASIC* é escrita com um número de linha antecedendo-a passa a fazer parte do programa, e será executada quando o programa for rodado.

PRINT tem sido utilizada neste capítulo como uma instrução, no entanto, ela pode ser utilizada como um *COMANDO*. Quando utilizada deste modo, sua ação, como qualquer outro *COMANDO*, é imediata; tão logo você tecle *RETURN*, a mensagem, resultado ou dado, será enviado para o vídeo. Além disso, não irá interferir de modo algum no programa que estiver na memória do computador.

PRINT UTILIZADO COMO COMANDO

Sempre que o computador não estiver rodando um programa, ele estará aguardando que você lhe envie um *COMANDO*.

Vamos testar agora o uso de *PRINT* como um *COMANDO*.

Para isso, utilize em seu computador a seqüência de operação abaixo, como ilustração do que foi dito e, depois, faça você mesmo algumas experiências.

```
PRINT "TESTE DE PRINT COMO COMANDO"
TESTE DE PRINT COMO COMANDO
```

O computador obedeceu ao comando *PRINT* e enviou para o vídeo a *"STRING" TESTE DE PRINT COMO COMANDO*.

```
PRINT "6 + 12 - 3"
6 + 12 - 3
```

Analogamente a *"STRING"* 6 + 12 - 3 é enviada para o vídeo.

```
PRINT 6 + 12 - 3
15
```

Neste caso, o computador calcula a expressão acima e envia o resultado para o vídeo.

```
PRINT ((6-2) * SQR(2) +365) / (2^3); "FACIL, FACIL"
46.3321068 FACIL FACIL
```

Por mais complicada que seja a expressão, o resultado é obtido instantaneamente e sem erros.

Em suma, o uso da instrução *PRINT* como *COMANDO* lhe permitirá enviar, para o vídeo, uma mensagem imediatamente, bem como utilizar seu computador como uma calculadora capaz de avaliar expressões matemáticas.

EXERCÍCIOS

EXERCÍCIO 2.1

Escreva um programa que, quando rodado, resulte no vídeo:

```
          D
        O  E
      D     S
    N         C
  I             E
B                 N
 U                  D
  S                  O
```

EXERCÍCIO 2.2

Escreva um programa que envie para o vídeo:

```
ANTONIMOS

DIA       NOITE
CARO      BARATO
FRIO      QUENTE
LONGE     PERTO
FACIL     DIFICIL
```

EXERCÍCIO 2.3

Escreva um programa que resulte em:

```
NUMERO DE SEGUNDOS EM:

1 MINUTO        60
1 HORA          3600
1 DIA           86400
1 ANO           3153600
```

EXERCÍCIO 2.4

Escreva um programa que resulte em:

```
CALCULO DE AREA

UM QUADRADO DE 52 METROS DE LADO
TEM 2704 METROS QUADRADOS DE AREA
UM CIRCULO COM 133 CENTIMETROS DE
DIAMETRO TEM 13885.865 CENTIMETROS
QUADRADOS DE AREA
```

área do quadrado lado X lado

área do círculo (diâmetro ao quadrado) X 3.1415/4

CAPÍTULO 3

GENERALIZANDO AS SOLUÇÕES

INTRODUÇÃO

Nós vimos como efetuar operações matemáticas em *BASIC*, utilizando a instrução *PRINT*. No entanto, um programa assim escrito resolve a expressão onde os valores estão fixados, e a mudança de um único desses valores, nos obriga a reescrever o programa. Tal condição, obviamente não pode ser aceita e este capítulo mostra como a generalização da solução de um determinado problema é obtida em *BASIC*.

ILUSTRANDO O PROBLEMA

Para que não restem dúvidas, vamos apresentar uma situação onde fica clara a necessidade de se reescrever o programa quando os dados, relativos a um determinado problema, são mudados.

Seja o cálculo da média simples (aritmética) entre 4 valores, os quais são:

8, 3, 6, 7

Um programa, completo, bem feito, com os recursos de que dispomos até agora, seria:

Exemplo 3.1

```
10 PRINT "MEDIA SIMPLES DE 4 VALORES"
20 PRINT
30 PRINT "MEDIA ENTRE 8, 3, 6 E 7"
40 PRINT
50 PRINT "VALOR DA MEDIA "; (8+3+6+7)/4
60 END
```

Rodando este programa, você tem:

```
MEDIA SIMPLES DE 4 VALORES

MEDIA ENTRE 8, 3, 6 E 7

VALOR DA MEDIA  6
```

Suponhamos que temos que resolver o mesmo problema, com a única diferença, que o primeiro valor, ao invés, de 8, seja 4. O programa que está na memória do computador não pode ser usado, pois ele calcula a média entre 8, 3, 6 e 7 e não entre 4, 3, 6 e 7 como queremos agora.

A solução é reescrever o programa, modificando as linhas 30 e 50 como mostradas abaixo:

```
30 PRINT "MEDIA ENTRE 4, 3, 6 E 7"
50 PRINT "VALOR DA MEDIA"; (4+3+6+7)/4
```

O programa, agora, passa a calcular a média entre os valores 4, 3, 6 e 7.

Você, se for um daqueles "altamente entusiasmado", poderá estar pensando: "até que não é tão trabalhoso assim". No entando, se o objetivo é calcular a média das notas dos alunos de um colégio, não há entusiasmo que sobreviva.

Nosso objetivo, portanto, é obter um método de solução das expressões, que seja independente de valores, quer dizer, generalizar a solução da expressão.

VARIÁVEIS

A generalização de uma determinada expressão é obtida utilizando-se o conceito matemático de variável.

Variável é um nome que é usado em substituição a um valor para generalizar a representação de uma expressão.

Seja, por exemplo, o problema da soma de dois números, chamando-se o primeiro, qualquer que seja o seu valor, de *A* e o segundo de *B*, tem-se que a representação genérica da soma desses dois números é:

$$A + B$$

Se o primeiro número for 243 e o segundo 12, *A* é o nome genérico que se dá a 243 e *B* é o nome de 12. Usando este conceito, uma forma genérica para o cálculo de média que referimos, anteriormente, é:

$$(A + B + C + D)/4$$

onde *A* representa o primeiro valor e *B*, *C* e *D*, o segundo, terceiro e quarto valores respectivamente.

Se adotarmos para *A* o valor 8 e para *B*, *C* e *D* 3, 6 e 7 respectivamente, a primeira média a que referimos será calculada. Mudando-se o valor de *A* para 4 teremos o cálculo da segunda média.

Tanto os computadores, como a linguagem *BASIC*, foram criados para operarem sem qualquer problema com expressões generalizadas e com variáveis.

VARIÁVEIS EM BASIC

Em *BASIC* existem dois tipos de variáveis:

- *VARIÁVEIS NUMÉRICAS* e
- *VARIÁVEIS ALFANUMÉRICAS*.

As variáveis numéricas representam os valores numéricos e são utilizadas deste modo, em cálculos.

As variáveis alfanuméricas representam as "STRINGS" e são utilizadas deste modo, para envio de mensagens, avaliação e comparação de *"STRING"*, etc.

NOME DAS VARIÁVEIS

Este é mais um ponto, onde as diversas versões da linguagem *BASIC* apresentam diferenças entre si. A versão mais restrita obriga que o nome de uma variável seja, no máximo, formado por dois caracteres: o primeiro uma letra e o segundo, se for usado, deve obrigatoriamente, ser um número. As *variáveis alfanuméricas* obedecem a esta mesma regra de formação do nome, devendo, no entanto, ser acrescentado um cifrão no final do mesmo. Como pretendemos que os programas aqui apresentados sejam adequados a qualquer tipo de computador, essa regra de formação de nome para variáveis, será adotada.

Assim, podemos ter:

a.) Nomes para variáveis com um caráctere:

As letras do alfabeto, ou seja, 26 nomes para variáveis.

Por exemplo:

Numéricas A, B, C, X, etc.
Alfanuméricas A$, B$, C$, X$, etc.

b.) Nomes para variáveis com dois carácteres:

As letras do alfabeto seguidas dos algarismos de 0 a 9, e portanto, 260 nomes para variáveis.

Por exemplo:
Numéricas A0, A1, B9, X7, etc.
Alfanuméricas A0$, A1$, B9$, X7$, etc.

Podemos ver que mesmo adotando essa condição restrita, temos um total de 286 nomes diferentes para variáveis, o que é muito mais do que normalmente necessário.

ATRIBUINDO VALOR A UMA VARIÁVEL

Definido o conceito de variável e as regras para escolha de seu nome, vamos conhecer, agora, os métodos para se atribuir valores as mesmas.

Quando você usa, pela primeira vez, em um programa o nome de uma variável, o computador, automaticamente, reserva em sua memória, um lugar adequado, formando o par:

NOME DA VARIÁVEL - VALOR DA VARIÁVEL

Enquanto nenhum valor for atribuído a essa variável, a posição da memória reservada para o valor da variável conterá zero, se numérica ou nenhum caractere, se alfanumérica. Não se preocupe como é que isso ocorre dentro do computador. Imagine o mesmo como uma pessoa extremamente organizada e que possui um arquivo com várias pastas e que, para cada variável declarada, uma pasta é por ela reservada para guardar o valor da variável.

Três são os modos de se atribuir valor a uma variável em *BASIC*:

IMEDIATO - o valor da variável é declarado no programa, imediatamente, após a sua denominação.

EXTERNO - o valor da variável é fornecido através de um periférico do computador, normalmente o teclado.

INDIRETO - o valor da variável é obtido de uma tabela escrita no próprio programa.

A INSTRUÇÃO LET

A atribuição de um valor diretamente a uma variável é feita através da instrução *LET*. Esta instrução permite também fazer corresponder a essa variável uma expressão. Deste modo, temos dois formatos para a instrução *LET*.

Para atribuição direta de um valor:

número da linha *LET* (nome da variável) = (valor da variável).

Exemplos:

```
100 LET A = 123
```

Atribui à variável numérica, de nome *A*, o valor 123.

```
100 LET A$ = "BASIC PARA INICIANTES"
```

Atribui à variável alfanumérica, de nome A$, a *"STRING" "BASIC PARA INICIANTES"*.

Para correspondência a uma expressão:

número da linha *LET* (nome da variável) = (expressão).

Exemplos:

```
100 LET  C= A+B
```

A variável *C* assume o valor resultante da soma do valor da variável *A* com o valor da variável *B*. Se tivermos -3 como valor da variável *A* e 133 como valor da variável *B*, após essa instrução, a variável *C* terá o valor 130.

```
100 LET A=A+1
```

A variável *A* assume o valor resultante da soma do valor atual da própria variável *A*, com o número 1. Sendo, por exemplo, 3 o valor atual da variável *A*, após essa instrução, *A* terá o valor de 4.

```
100 LET A$= B$ + C$ + D$
```

As *"STRINGS"* correspondentes à variáveis *B$*, *C$* e *D$* são ligadas e a variável *A$* assume essa nova *"STRING"*. Se *B$= "BASIC"*, *C$ = "PARA"* e *D$ = "INICIANTES"* após essa linha do programa, teremos: *A$ = "BASIC .PARA INICIANTES"*.

ATRIBUIÇÃO DIRETA DE VALORES

Os exemplos de programa a seguir objetivam mostrar o uso da instrução *LET* e, também, a operação da instrução *PRINT* quando usada com variáveis.

Exemplo 3.2

```
100 LET A = 20
200 PRINT A
300 END
```

Rodando, temos

```
RUN

20
```

A operação do programa linha a linha é:

100 - A variável *A* foi declarada e o valor 20 foi levado para a posição de memória reservada para a variável *A*.

200 - Temos aqui, o novo formato da instrução *PRINT*. Quando utilizada deste modo, a instrução *PRINT* busca o valor associado à variável declarada (neste caso *A*) na memória do computador e envia este valor para o vídeo. Ao buscar o valor da variável *A* na memória do computador, o valor 20 lá depositado pela instrução *LET* da linha 100 foi encontrado e foi enviado para o vídeo.

300 - Instrução final do programa.

Exemplo 3.3

```
100 LET C= A + B
200 PRINT C
300 END
```


Rodando, temos:

```
RUN

0
```

Analisando a operação do programa linha a linha temos:

100 - A variável *C* assume o valor resultante da soma do valor da variável *A* com o valor da variável *B*. Como não foram declarados valores para as variáveis *A* e *B*, ambos possuem o valor zero.

200 - O valor de *C* (neste caso zero) é enviado para o vídeo.

300 - Linha final do programa.

Acrescente as linhas abaixo, ao exemplo 3.3:

```
80 LET A= 1536
90 LET B= -689
```

Listando o programa, temos:

Exemplo 3.4

```
80 LET A= 1536
90 LET B= -689
100 LET C= A+B
200 PRINT C
300 END
```

Rodando, temos:

```
RUN

847
```

Como foi feita a declaração prévia dos valores das variáveis *A* e *B* o resultado da soma de 1536 com - 689 foi obtido e enviado para o vídeo.

Exemplo 3.5

```
100  LET A$ = "MINHA TERRA "
200  LET B$ = "TEM PALMEIRAS"
300  LET C$ = A$ + B$
400  PRINT C$
500  END
```

```
RUN

MINHA TERRA TEM PALMEIRAS
```

Analisando o programa linha a linha, temos:

100 - *A variável alfanumérica A$* é declarada e a *"STRING" "MINHA TERRA "* é armazenada na posição de memória reservada a essa variável.

200 - *A "STRING" "TEM PALMEIRAS"* é enviada para a área de memória reservada à variável B$.

300 - Uma nova posição de memória é reservada e os conteúdos das variáveis *A$* e *B$* são concatenados, formando a *"STRING" "MINHA TERRA TEM PALMEIRAS"* a qual é enviada para essa posição.

400 - A *"STRING"* contida na posição de memória *C$* é enviada para o vídeo.

500 - Última linha do programa.

É importante destacarmos que para muitas versões de *BASIC*, o uso da palavra *LET* nas instruções que acabamos de ver, é facultativo. O formato da instrução pode ser, desse modo, simplificado como mostrado abaixo:

NORMAL

```
100 LET A=10
200 LET B$="BASIC "
300 LET C= A + B
```

SIMPLIFICADO

```
100 A=10
200 B$="BASIC "
300 C= A + B
```

Para verificar se a versão de BASIC do seu computador aceita essa simplificação, entre e rode o programa abaixo. Se o seu programa for aceito, isto significa que para o *BASIC* do seu computador o uso da palavra *LET* é dispensável.

Exemplo 3.6

```
10 A$="TESTE "
20 B=123456
30 PRINT A$;B
40 END
```

Embora o computador pessoal que estamos utilizando possua uma versão de *BASIC* onde a palavra *LET* é desnecessária, vamos utilizar em nossos exemplos a forma completa da instrução para que esses exemplos possam ser utilizados com qualquer versão *BASIC*.

PRINT COM EXPRESSÃO

Antes de estudarmos o próximo modo de se atribuir valor a uma variável, vamos analisar a instrução *PRINT* quando o complemento for formado por expressões.

Neste formato da instrução *PRINT*, a expressão que a complementa é calculada automaticamente, e o resultado obtido é enviado para o vídeo. Se o resultado da expressão não for necessário ao desenvolvimento do programa, este procedimento deve ser usado, ao invés de se atribuir a uma variável o valor da expressão.

Exemplo 3.7

```
100  LET A = 1536
200  LET B =  - 689
300  PRINT A + B
400  END
```

Rodando o programa, temos:

```
847
```

Exemplo 3.8

```
100  LET A$ = "MINHA TERRA "
200  LET B$ = "TEM PALMEIRAS"
300  PRINT A$ + B$
400  END
```

Rodando o programa, temos:

```
RUN

MINHA TERRA TEM PALMEIRAS
```

A instrução *PRINT* permite a concatenação de variáveis alfanuméricas, utilizando o sinal *"+"* ou, então, *um ponto e vírgula*.

A INSTRUÇÃO INPUT

A instrução *INPUT* possibilita a atribuição de valores e *"STRING" às variáveis numéricas e às alfanuméricas* respectivamente, durante a execução do programa, através de um periférico, normalmente, o teclado do computador.

O formato da instrução *INPUT é:*

número da linha *INPUT* (nome da variável)

Exemplos de formato da instrução:

```
100 INPUT B    (para variáveis numéricas)

100 INPUT B$   (para variáveis alfanuméricas)
```

Se durante a execução de um programa, o computador encontrar uma linha com a instrução *INPUT*, o processamento do programa é interrompido e um ponto de interrogação é enviado para a tela do vídeo. O computador, a partir daí, fica aguardando que você envie o valor da variável através do teclado. Você deve, então, teclar o valor correspondente da variável e pressionar *RETURN*. O computador atribui à variável este valor e o processamento do programa continua.

Exemplo 3.9

```
100 INPUT A
200 INPUT B
300 PRINT A + B
400 END
```

Rode o programa e você terá no vídeo, um ponto de interrogação. O computador está aguardando que o valor de A seja entrado pelo teclado. Entre com um valor qualquer, digamos 12. Tão logo você entrar com o valor 12, um novo ponto de interrogação correspondente a instrução *INPUT* da linha 200 aparecerá no vídeo. Entre, por exemplo, com o número 45. A soma dos dois números será então calculada e enviada para o vídeo pela instrução *PRINT* da linha 300.

Na tela do vídeo essa seqüência de operações ficará assim:

```
RUN

?12
?45
57
```

Rodando o programa novamente e mudando os valores para 6439 e -6409, teremos:

```
RUN

?6439
?-6409
30
```

MENSAGENS EXPLICATIVAS

No capítulo anterior, falamos da necessidade de se incluir mensagens explicativas em programas para possibilitar que estas fossem utilizáveis por um usuário qualquer. A instrução *INPUT* é tipicamente uma instrução onde este procedimento se faz necessário para orientar o usuário quanto à entrada de dados.

O exemplo 3.9 é muito pobre de informações. Somente através da listagem deste, e conhecendo *BASIC* é que se poderá saber sua finalidade e como utilizá-lo. Para sanar essa deficiência, acrescente-lhe as linhas abaixo:

```
70 PRINT "ESTE PROGRAMA EFETUA A SOMA DE DOIS NUMEROS"
80 PRINT
90 PRINT "TECLE O VALOR DO PRIMEIRO NUMERO E APOS PRESSIONE A
   TECLA MARCADA RETURN"
110 PRINT
120 PRINT "TECLE O VALOR DO SEGUNDO NUMERO E APOS PRESSIONE A
    TECLA MARCADA RETURN"
210 PRINT
300 PRINT "A SOMA DE "; A; "COM "; B; "E' "; A+B
```

Após a introdução dessas linhas, listando o programa, você terá:

Exemplo 3.10

```
70 PRINT "ESTE PROGRAMA EFETUA A SOMA DE DOIS NUMEROS"
80 PRINT
90 PRINT "TECLE O VALOR DO PRIMEIRO NUMERO E APOS PRESSIONE A
   TECLA MARCADA RETURN"
100 INPUT A
110 PRINT
120 PRINT "TECLE O VALOR DO SEGUNDO NUMERO E APOS PRESSIONE A
    TECLA MARCADA RETURN"
200 INPUT B
210 PRINT
300 PRINT "A SOMA DE "; A; "COM "; B; "E' "; A+B
400 END
```

Antes de rodarmos esse exemplo, vou lhe dar uma sugestão. Um bom método de auto-avaliação, e também um excelente treinamento dos conhecimentos adquiridos, é ler os exemplos propostos e antes de rodá-los, procurar descobrir como os mesmos vão operar e quais são os resultados esperados. Faça isso de agora em diante, começando por este exemplo. Não veja o resultado e nem rode-o sem antes analisá-lo.

Bem, agora que você analisou o exemplo 3.10, rode-o

```
RUN

ESTE PROGRAMA EFETUA A SOMA DE DOIS NUMEROS

TECLE O VALOR DO PRIMEIRO NUMERO E APOS PRESSIONE A TECLA
MARCADA RETURN
?1234

TECLE O VALOR DO SEGUNDO NUMERO E APOS PRESSIONE A TECLA MARCADA
RETURN
?634

A SOMA DE 1234 COM 634 E' 1868
```

O programa "rodou" como você esperava ? Qualquer que seja a sua resposta, continue a analisar os programas antes de rodá-los.

ATRIBUINDO VALORES SIMULTANEAMENTE

Você pode com uma mesma instrução de *INPUT* atribuir valores a várias variáveis. Neste caso, o formato da instrução *INPUT* é:

número da linha *INPUT* (nome da variável),.....,(nome da variável).

Os valores podem ser entrados um a um, ou numa mesma linha, desde que separados por vírgula.

Exemplo 3.11

```
10  PRINT "SOMA DE 4 NUMEROS"
20  PRINT
30  PRINT "ENTRE COM OS NUMEROS, SEPARANDO-OS"
32  PRINT "POR VIRGULAS E APOS, TECLE RETURN"
40  INPUT A,B,C,D
50  PRINT
60  PRINT "A SOMA E' ";A + B + C + D
70  END
```

```
RUN
SOMA DE 4 NUMEROS

ENTRE COM OS NUMEROS, SEPARANDO-OS
POR VIRGULAS E APOS, TECLE RETURN
?12,-456,678,12564

A SOMA E' 12798
```

INPUT COM MENSAGENS

Como a instrução *INPUT* deve ter sempre associada uma instrução de *PRINT*, várias versões de *BASIC* permitem o seguinte formato:

número da linha *INPUT* "mensagem"; (variável)

Exemplo:

```
100 INPUT "ENTRE COM O PRIMEIRO NUMERO "; A
```

Antes de reescrevemos o Exemplo 3.10, verifique se o seu computador permite esse formato de *INPUT*, rodando o exemplo a seguir:

Exemplo 3.12

```
100 INPUT "ENTRE COM SEU NOME "; A$
200 PRINT "VOCE SE CHAMA ";A$
300 END

RUN

ENTRE COM O SEU NOME JOSE
VOCE SE CHAMA JOSE
```

Se o exemplo 3.12 "rodou" sem problemas, então você pode utilizar este formato de *INPUT*. Se este não foi o caso, então continue a usar a associação *PRINT* e *INPUT*. Nós também a utilizaremos nos nossos exemplos exceto, é claro, no exemplo 3.13, que é o exemplo 3.10 reescrito.

Exemplo 3.13

```
100 PRINT "ESTE PROGRAMA EFETUA A SOMA DE DOIS NUMEROS"
200 PRINT
300 INPUT "TECLE O VALOR DO PRIMEIRO NUMERO E APOS PRESSIONE A
    TECLA MARCADA RETURN "; A
400 PRINT
500 INPUT "TECLE O VALOR DO SEGUNDO NUMERO E APOS PRESSIONE A
    TECLA MARCADA RETURN "; B
600 PRINT
700 PRINT "A SOMA DE "; A; " COM "; B; " E' "; A+B
800 END
```

O resultado deste programa é exatamente o mesmo do exemplo 3.10 e, por isso, não vamos apresentá-lo

CONCLUINDO

Vimos, neste capítulo, dois modos de se atribuir valores a uma variável, um imediato, "embutido" no programa através da instrução LET e outro externo, via teclado, através da instrução INPUT. O modo indireto que faz uso de uma tabela escrita no programa, será visto num capítulo futuro, pois, para sua adequada utilização, algumas instruções ainda não vistas são necessárias.

A instrução PRINT à qual foi dedicado todo o capítulo 2 reapareceu aqui com a ampliação de seu campo de utilização, incluindo agora também as variáveis.

EXERCÍCIOS

EXERCÍCIO 3.1

Escreva um programa que resulte em:

```
QUAL E' O SEU NOME?
?CARLOS (NOME ENTRADO VIA TECLADO)
QUANDO VOCE NASCEU?
?1952   (ANO ENTRADO VIA TECLADO)
MEUS PARABENS, CARLOS. ESTE ANO VOCE COMPLETA 30 ANOS
```

EXERCÍCIO 3.2

Escreva um programa que resulte em:

```
ENTRE COM UM NÚMERO
36 (número entrado via teclado)
O DOBRO DE 36, MAIS 16, MULTIPLICADO POR 18 E DIVIDI
DO POR 6, RESULTA EM 264. DÚVIDA ?
```

EXERCÍCIO 3.3

Escreva um programa que resulte em:

```
QUAL A SUA IDADE ?
21 (idade entrada via teclado)
PUXA ! VOCÊ TEM MAIS QUE 11037600 MINUTOS DE IDADE.
```

CAPÍTULO 4

MODIFICANDO O FLUXO DO PROGRAMA

INTRODUÇÃO

Os programas que escrevemos até agora tiveram suas instruções executadas na seqüência crescente dos números das linhas, iniciando pela linha de menor e terminando na linha de maior valor. Essa forma seqüencial e contínua de execução das instruções do programa constitui uma limitação de tal monta que nenhum programa de média complexidade pode, nessas condições, ser desenvolvido. Consequentemente, instruções que permitem modificar o fluxo do programa foram incluídas na linguagem *BASIC*.

VÁ PARA ...

Calma ! Não vá perder a paciência e nem a sua fleuma. O título acima é a tradução da expressão *GOTO* utilizada na instrução que permite a mudança incondicional da seqüência de execução de um programa. O formato dessa instrução é:

número da linha *GOTO* número da linha a ser executada.

Exemplos:

```
150 GOTO 320
```

A próxima linha a ser executada será a linha 320, mesmo que existam outras linhas no programa, entre as linhas 150 e a 320.

```
100 GOTO 5
```

A linha seguinte a ser executada pode ser uma linha anterior à linha da instruções *GOTO*. Neste caso, a linha a ser executada será a linha 5.

As regras para utilização da instrução *GOTO* são bem simples. Você pode desviar a execução do programa para qualquer linha existente no mesmo, anterior ou posterior à linha da ins

trução. Um desvio para uma linha vazia, ou seja, não existente no programa, provocará a interrupção da execução do mesmo, e uma mensagem de erro será enviada para o vídeo.

O exemplo 4.1 não possui outra utilidade a não ser mostrar como a instrução *GOTO* opera. Analise o programa, acompanhando mentalmente a seqüência de execução do mesmo e, somente após isso, entre com mesmo e rode-o.

Exemplo 4.1

```
10 PRINT "INSTRUCAO GOTO"
20 GOTO 100
30 PRINT "VENHO DA LINHA 110 E ESTOU NA 30"
40 PRINT "EXECUTEI A 40"
50 GOTO 80
60 PRINT "ESTA E' A LINHA 60"
70 GOTO 120
80 PRINT "VIM PARA A LINHA 80 A PARTIR DA 50"
90 GOTO 60
100 PRINT "PULEI DA LINHA 20 PARA A LINHA 100"
110 GOTO 30
120 PRINT "SAI DA LINHA 70 PARA A 120 E TERMINO NA 130"
130 END
```

Rodando temos:

```
RUN

INTRUCAO GOTO
PULEI DA LINHA 20 PARA A LINHA 100
VENHO DA LINHA 110 E ESTOU NA 30
EXECUTEI A 40
VIM PARA A LINHA 80 A PARTIR DA 50
ESTA E' A LINHA 60
SAI DA LINHA 70 PARA A LINHA 120 E TERMINO NA 130
```

A execução do programa agora não é puramente seqüencial e, devido as instruções *GOTO* presentes, essa ordem, como você pode comprovar é: 10, 20, 100, 110, 30, 40, 50, 80, 90,60, 70, 120, 130.

Vamos imaginar um programa que possua uma instrução *GOTO* que faça o programa voltar a uma instrução anteriormente executada. Se o programa, a partir daí, seguir a mesma seqüência, anteriormente executada, mais cedo ou mais tarde, a instrução *GOTO* será novamente executada "fechando o círculo". O programa, neste caso, possui um fluxo em malha fechada, o qual é chamado de "loop".

O exemplo 4.2 é um programa que possui uma malha de fluxo fechado, representada pelas instruções das linhas 300, 400 e 500.

Exemplo 4.2

```
100 PRINT "EXEMPLO DE PROGRAMA COM MALHA FECHADA"
200 PRINT
300 PRINT "ESTA E' UMA MALHA FECHADA"
400 PRINT
500 GOTO 300
600 PRINT
700 PRINT "FIM DO PROGRAMA"
800 END
```

Analisando o programa, você verá que o "loop" formado pelas linhas 300, 400 e 500 será executado indefinidamente, não existindo possibilidade de que as linhas 600, 700 e 800 sejam executadas, ou seja, que o programa saia do "loop".

De fato, uma vez iniciada a execução desse programa a *"STRING" "EXEMPLO DE PROGRAMA COM MALHA FECHADA"*, será enviada, e após uma linha em branco, a *"STRING" "ESSA É UMA MALHA FECHADA"* seguida de uma linha em branco, será enviada após o que a instrução *GOTO* faz com que o programa volte à linha 300, a qual torna a enviar a *"STRING" "ESTA É UMA MALHA FECHADA"* seguida de uma linha em branco e de novo a instrução *GOTO* reinicia o ciclo. O resultado é o envio de uma seqüência infinita da *"STRING" "ESTA É UMA MALHA FECHADA"* seguida da linha em branco para o vídeo, preenchendo totalmente a tela. Por mais que você deixe o computador rodando, o envio da *"STRING"* e da linha em branco, para o vídeo, não será interrompido. Evidentemente, este programa não está correto, tendo como objetivo mos-

trar a ocorrência do fluxo em malha fechada infinito.

Entretanto, nem todo fluxo fechado leva a um *"LOOP"* infinito. Nós veremos, posteriormente, instruções chamadas condicionais que possibilitam controlar a permanência ou não do programa num *"LOOP"*. Mas, se ao se estabelecer esse controle, um erro for cometido, um *"LOOP"* infinito poderá ser criado, resultando numa situação semelhante a do nosso Exemplo 4.2.

Como fazer para interromper a execução de um programa numa situação desta ? Se você tentar, você verá que mesmo pressionando teclas, o programa não é interrompido. Um modo um tanto "drástico" mas eficiente, é desligar o computador, embora essa atitude acarrete na perda do programa.

SAINDO DA ARMADILHA

Para interromper o processamento neste caso e em qualquer outra circunstância, foi previsto em *BASIC* um comando externo enviado via teclado do computador. Trata-se de:

CONTROL C

O envio desse comando é feito pressionando a tecla marcada *CONTROL* (ou *CTRL* abreviadamente) e sem soltá-la pressiona-se também a *TECLA C*. Em seguida, deve-se soltar a *TECLA C* e, após, a tecla *CONTROL*. Isto feito, o computador irá interromper o processamento e enviará uma mensagem indicando em qual linha do programa o mesmo foi interrompido.

Agora que você já sabe como sair do *"LOOP INFINITO"*, rode o programa e comprove nossas afirmações.

ESCOLHENDO O DESTINO

A instrução *GOTO*, como vimos, possibilita a mudança do fluxo do programa para uma determinada linha do mesmo. Situações existem, no entanto, nas quais uma escolha da linha de destino, entre algumas opções, é desejável. Essa possibilidade de escolha do destino é dada pela instrução em *BASIC* de formato:

número linha *ON* (variável) *GOTO* número de linha, ... número de linha

Vamos às regras dessa instrução:

1.) A variável indicada deve ser obrigatoriamente numerica e inteira.

2.) O valor dessa variável indicará ordenadamente para qual linha a execução do programa será desviada.

Por exemplo:

```
500 ON X GOTO 100, 220, 340, 600, 1234
```

A escolha do nome da variável é arbitrária, neste caso, escolhemos *X*. Qualquer outro nome, poderia ter sido usado.

A linha a ser executada dependerá do valor de *X*, sendo executada a linha 100 se *X* for 1, a linha 220 se *X* for 2, a linha 340 se *X* for 3 e assim por diante.

Vamos escrever um pequeno programa para exemplificar o uso desta instrução:

Exemplo 4.3

```
10 PRINT "ENTRE COM UM NUMERO INTEIRO DE 1 A 3"
15 PRINT
20 INPUT A
30 ON A GOTO 50,80,90
40 GOTO 110
50 PRINT "VOCE ESCOLHEU O NUMERO UM"
55 PRINT
60 GOTO 10
70 PRINT "VOCE ESCOLHEU O NUMERO DOIS"
75 PRINT
80 GOTO 10
90 PRINT "VOCE ESCOLHEU O NUMERO TRES"
95 PRINT
100 GOTO 10
110 PRINT"O NUMERO ESCOLHIDO OU E' MENOR QUE UM OU E'
    MAIOR QUE TRES"
120 END
```

Rodando o programa, teremos:

```
RUN
ENTRE COM UM NUMERO INTEIRO DE 1 A 3
?2
VOCE ESCOLHEU O NUMERO DOIS
ENTRE COM UM NUMERO INTEIRO DE 1 A 3
```

Quando você escolheu o número dois, esse valor foi atribuído à variável *A* e, deste modo, na instrução *ON A GOTO* a linha de destino escolhida foi a segunda, ou seja, a de número 70. O programa foi desviado para a linha 70, a mensagem *"VOCÊ ESCOLHEU O NÚMERO DOIS"* foi enviada para a tela, o programa prosseguiu para a linha 75, enviando uma linha em branco, e a instrução *GOTO* 10 da linha 80 desvia o programa para a linha 10, a qual volta a solicitar a escolha de um número inteiro de 1 a 3. Escolhendo o número 3, você tem a seqüência de execução das linhas 20, 30 daí para 90 e 95 e de volta a 10, resultando no vídeo:

```
?3
VOCE ESCOLHEU O NUMERO TRES
ENTRE COM UM NUMERO INTEIRO DE 1 A 3
```

Vamos verificar o que ocorre caso você envie um número não inteiro, por exemplo: 1,25

```
?1.25
VOCE ESCOLHEU O NUMERO 1
ENTRE COM UM NUMERO INTEIRO DE 1 A 3
```

O computador usa tão somente à parte inteira do número que você enviou e pula para a linha correspondente ao número de ordem desse inteiro.

Se você escolheu um número menor que 1 ou maior que o número de linhas relacionadas, a execução do programa continuará na linha seguinte a atual. Se você entrar com um número negativo, o programa será interrompido e uma mensagem de erro será enviada.

Entre o número 5 e teremos:

```
?5
O NUMERO ESCOLHIDO OU E' MENOR QUE UM OU E' MAIOR QUE
TRES
```

A escolha de um valor para a variável superior ao número de linhas relacionadas na instrução faz com que o computador execute a instrução da linha seguinte, neste caso, a 40, que desvia o programa para a linha 110, a qual enviou a mensagem acima e após a linha 120, terminando a execução do programa.

UM PROGRAMA ÚTIL

Como novo exemplo de uso da instrução *ON ...GOTO*, vamos escrever um programa o qual já possui um pequeno grau de utilidade e que servirá de guia para programas semelhantes.

O programa a ser escrito fará conversões de centímetros para polegadas, graus centígrados para fahrenheit e vice-versa. Inicialmente uma lista de opções será apresentada a qual lhe permitirá a escolha da conversão a ser feita. Isso feito, o valor a ser convertido será solicitado. Após, o resultado será apresentado e o programa voltará a escolha inicial. Para sair do programa, uma das opções é exatamente *"SAIR DO PROGRAMA"*. Apresentaremos agora, uma listagem do programa e a seqüência obtida quando rodamos o mesmo.

Exemplo 4.4

```
100  PRINT "CONVERSAO DE UNIDADES"
110  PRINT
120  PRINT "PARA CONVERTER:"
130  PRINT
140  PRINT "     1- CENTIMETROS EM POLEGADAS"
150  PRINT "     2- POLEGADAS EM CENTIMETROS"
160  PRINT "     3- GRAUS CENTIGRADOS EM FAHRENHEIT"
170  PRINT "     4- GRAUS FARENHEIT EM CENTIGRADOS"
180  PRINT "     5- SAIR DO PROGRAMA"
190  PRINT
200  PRINT "ESCOLHA:1, 2, 3, 4, OU 5"
210  PRINT
220  INPUT X
230  ON X GOTO 250,300,350,400,460
240  GOTO 500
250  PRINT "ENTRE COM O VALOR EM CENTIMETROS"
260  INPUT A
270  LET B = A * .3937
280  PRINT A;" CENTIMETROS EQUIVALEM A ";B;" POLEGADAS
     "
290  GOTO 200
300  PRINT "ENTRE COM O VALOR EM POLEGADAS"
310  INPUT A
320  LET B = A * 2.54
330  PRINT A;" POLEGADAS EQUIVALEM A ";B;" CENTIMETROS
     "
340  GOTO 200
350  PRINT "ENTRE COM O VALOR EM GRAUS CENTIGRADOS"
360  INPUT A
370  LET B = (A * 9 / 5) + 32
380  PRINT A;" GRAUS CENTIGRADOS EQUIVALEM A ";B;" GRA
     US FAHRENHEIT"
390  GOTO 200
400  PRINT "ENTRE COM O VALOR EM GRAUS FAHRENHEIT"
410  INPUT A
420  LET B = (A - 32) * 5 / 9
430  PRINT A;" GRAUS FAHRENHEIT EQUIVALEM A ";B;" GRAU
     S CENTIGRADOS"
440  GOTO 200
460  PRINT "ATE A PROXIMA VEZ"
470  END
```


Rodando o programa, temos:

```
CONVERSAO DE UNIDADES

PARA CONVERTER:

    1- CENTIMETROS EM POLEGADAS
    2- POLEGADAS EM CENTIMETROS
    3- GRAUS CENTIGRADOS EM FAHRENHEIT
    4- GRAUS FARENHEIT EM CENTIGRADOS
    5- SAIR DO PROGRAMA

ESCOLHA:1, 2, 3, 4, OU 5

?1
ENTRE COM O VALOR EM CENTIMETROS
?1.25
1.25 CENTIMETROS EQUIVALEM A .492125 POLEGADAS
ESCOLHA:1, 2, 3, 4, OU 5

?2
ENTRE COM O VALOR EM POLEGADAS
?2
2 POLEGADAS EQUIVALEM A 5.08 CENTIMETROS
ESCOLHA:1, 2, 3, 4, OU 5

?5
ATE A PROXIMA VEZ
```

ORGANIZANDO TAREFAS REPETITIVAS

Muitas vezes, ocorre a necessidade de se usar exatamente a mesma seqüência de instruções em vários pontos do programa. Com a finalidade de se evitar o trabalho de se escrever repetidas vezes essas instruções e, também, para possibilitar a redução do programa, foi introduzida em *BASIC* o conceito de subrotina.

REPETINDO INSTRUÇÕES

Como não é fácil explicar a solução de um problema, sem que o mesmo seja conhecido, vamos escrever um programa onde a repetição de uma mesma seqüência de operações ocorre deliberadamente.

O programa visa a calcular o volume em litros de um conjunto de caixas, cujas dimensões dadas em polegadas devem ser previamente convertidas em milímetros.

Exemplo 4.5

```
10  PRINT "CALCULO DE VOLUME EM LITROS COM"
11  PRINT "CONVERSAO DAS MEDIDAS DE POLEGADAS"
12  PRINT "PARA MILIMETROS"
20  PRINT
30  PRINT "ESCOLHA:"
40  PRINT "1- PARA CALCULO"
50  PRINT "2- PARA SAIR DO PROGRAMA"
60  PRINT
70  INPUT X
80  ON X GOTO 100,900
90  GOTO 1000
100  PRINT
110  PRINT "ENTRE COM O VALOR EM POLEGADAS"
120  INPUT Y
130  LET A = Y * 25.4
140  PRINT
150  PRINT "VALOR CORRESPONDENTE EM MILIMETROS ";A
160  LET M = A / 100
170  PRINT
180  PRINT "ENTRE COM O VALOR EM POLEGADAS"
190  INPUT Y
200  LET A = Y * 25.4
210  PRINT
220  PRINT "VALOR CORRESPONDENTE EM MILIMETROS ";A
230  LET N = A / 100
240  PRINT
250  PRINT "ENTRE COM O VALOR EM POLEGADAS"
260  INPUT Y
270  LET A = Y * 25.4
280  PRINT
290  PRINT "VALOR CORRESPONDENTE EM MILIMETROS ";A
300  LET P = A / 100
310  PRINT
320  LET V = M * N * P
330  PRINT "VOLUME EM LITROS ";V
340  GOTO 20
900  PRINT
910  PRINT "ATE A PROXIMA VEZ"
1000  END
```

Não existe nada de errado com este programa. Você pode entrá-lo e rodá-lo e se assim o fizer, você terá:

Rodando, temos:

```
CALCULO DE VOLUME EM LITROS COM
CONVERSAO DAS MEDIDAS DE POLEGADAS
PARA MILIMETROS

ESCOLHA:
1- PARA CALCULO
2- PARA SAIR DO PROGRAMA

?1

ENTRE COM O VALOR EM POLEGADAS
?12

VALOR CORRESPONDENTE EM MILIMETROS 304.8

ENTRE COM O VALOR EM POLEGADAS
?

BREAK IN 190

]RUN
CALCULO DE VOLUME EM LITROS COM
CONVERSAO DAS MEDIDAS DE POLEGADAS
PARA MILIMETROS

ESCOLHA:
1- PARA CALCULO
2- PARA SAIR DO PROGRAMA

?1

ENTRE COM O VALOR EM POLEGADAS
?12

VALOR CORRESPONDENTE EM MILIMETROS 304.8

ENTRE COM O VALOR EM POLEGADAS
?3

VALOR CORRESPONDENTE EM MILIMETROS 76.2

ENTRE COM O VALOR EM POLEGADAS
?5

VALOR CORRESPONDENTE EM MILIMETROS 127

VOLUME EM LITROS 2.94967152

ESCOLHA:
1- PARA CALCULO
2- PARA SAIR DO PROGRAMA

?2

ATE A PROXIMA VEZ
```

Pela listagem, você pode ver claramente que a seguinte seqüência de instruções aparece 3 vezes no programa:

```
PRINT
PRINT"ENTRE COM O VALOR EM POLEGADAS"
INPUT Y
LET A=Y*25.4
PRINT
PRINT"VALOR CORRESPONDENTE EM MILIMETROS "; A
```

Visando diminuir o comprimento do programa e, conseqüentemente, o espaço ocupado na memória do computador desenvolveu-se a seguinte idéia.

"Separar este conjunto de instruções que é repetido, escrevendo-o uma única vez e criar uma instrução que, quando for necessário utilizá-lo desvie o programa, execute esse conjunto de instruções e retorne ao ponto de partida".

GOSUB E RETURN

Este conjunto de instruções (que é repetido no programa) é chamado de SUBROTINA e a instrução que permite mudar a seqüência do programa, para executar o conjunto de instruções da subrotina tem o formato:

número da linha GOSUB número da primeira linha da subrotina.

Exemplo:

```
100 GOSUB 500
```

Uma vez desviada a seqüência de execução do programa através da instrução *GOSUB*, a execução do programa deverá retornar a seqüência anterior, quando o conjunto de instruções da subrotina tiver sido executado. Isto é feito através da instrução *RETURN*. A instrução *RETURN*, portanto, é a última instrução executada em uma *SUBROTINA* e não poderá haver *SUBROTINA* sem pelo menos uma instrução de *RETURN*.

O retorno ao ponto de chamada é feito automaticamente. O número da linha da instrução seguinte a *GOSUB* é guardado na memória e o programa retorna a essa linha através da instrução *RETURN*.

MODIFICANDO E INCLUINDO A SUBROTINA

Vamos modificar o nosso programa para substituir no mesmo, aquele conjunto de instruções repetidas por uma subrotina.

Vamos colocar nossa subrotina em um ponto próximo ao final do programa, por exemplo, a partir da linha 800. A posição da subrotina no programa não é importante, tradicionalmente usa-se colocá-la no seu final. Entre as linhas abaixo:

```
800 PRINT
810 PRINT "ENTRE COM O VALOR EM POLEGADAS"
820 INPUT Y
830 LET A=Y*25.4
840 PRINT
850 PRINT "VALOR CORRESPONDENTE EM MILIMETROS ";A
860 RETURN
```

Essa é a nossa *SUBROTINA*. Ela inicia na linha de número 800 e este é, portanto, o número para a referenciarmos. Note também que a última instrução a ser executada é um *RETURN*.

A primeira vez no programa que utilizamos essa seqüência de instruções foi a partir da linha 100, donde na linha 100 fazemos:

```
100 GOSUB 800
```

Assim, quando o programa executar essa linha, a seqüência será desviada para a linha 800 e serão executadas as linhas de 800 a 860 da subrotina. Deste modo as linhas 110,

120, 130, 140, 150 do nosso programa não são mais necessárias. Limpe essas linhas do programa. (Se você não está lembrando, para limpar uma linha, basta enviar o número desta linha)

Analogamente as linhas 170 e 240 deverão ser substituídas por:

```
170 GOSUB 800
240 GOSUB 800
```

e as linhas 180, 190, 200, 210, 220, 250, 260, 270, 280 e 290 devem ser suprimidas.

Isto feito, listando o programa, você terá:

Exemplo 4.6

```
10  PRINT "CALCULO DE VOLUME EM LITROS COM "
11  PRINT "CONVERSAO DAS MEDIDAS DE POLEGADAS"
12  PRINT "PARA MILIMETROS"
20  PRINT
30  PRINT "ESCOLHA:"
40  PRINT "1- PARA CALCULO"
50  PRINT "2- PARA SAIR DO PROGRAMA"
60  PRINT
70  INPUT X
80  ON X GOTO 100,900
90  GOTO 1000
100  GOSUB 800
160  LET M = A / 100
170  GOSUB 800
230  LET N = A / 100
240  GOSUB 800
300  LET P = A / 100
310  PRINT
320  LET V = M * N * P
330  PRINT "VOLUME EM LITROS ";V
340  GOTO 20
800  PRINT
810  PRINT "ENTRE COM O VALOR EM POLEGADAS"
820  INPUT Y
830  LET A = Y * 25.4
840  PRINT
850  PRINT "VALOR CORRESPONDENTE EM MILIMETROS ";A
860  RETURN
900  PRINT
910  PRINT "ATE A PROXIMA VEZ"
1000  END
```


Como você pode ver, das 37 linhas originais, o programa ficou com 28 linhas, ou seja, uma redução de quase 25% no seu comprimento.

Rodando o programa, você poderá ver que o mesmo opera exatamente como antes, resultando no que foi apresentado na página de número 67. O uso de subrotina deve ser uma prática adotada por você e nós a utilizaremos sempre que for necessário nos exemplos futuros.

ESCOLHENDO A SUBROTINA

Uma instrução correspondente a *ON GOTO*, existe para a *GOSUB* e seu formato é:

número de linha *ON* (variável) *GOSUB* número de uma linha,...número de uma linha

Esta instrução lhe permite escolher entre várias subrotinas, qual deve ser executada.

O programa a seguir, além de exemplificar o uso dessa instrução, demostra mais uma vez o uso de uma subrotina.

Exemplo 4.7

```
10  PRINT "CALCULOS MATEMATICOS"
20  PRINT
30  PRINT "ESCOLHA"
40  PRINT "1- SOMA DE DOIS NUMEROS"
50  PRINT "2- PRODUTO DE DOIS NUMEROS"
60  PRINT "3- QUOCIENTE DE DOIS NUMEROS"
70  INPUT X
80  ON X GOSUB 200,300,400
90  GOSUB 600
100  PRINT "NOVO CALCULO?"
110  PRINT
120  PRINT "1- SIM"
130  PRINT "2- NAO"
140  INPUT Y
150  ON Y GOTO 10,700
200  GOSUB 500
210  LET C = A + B
220  LET M$ = "SOMA "
230  RETURN
300  GOSUB 500
310  LET C = A * B
320  LET M$ = "PRODUTO "
330  RETURN
400  GOSUB 500
410  LET C = A / B
420  LET M$ = "QUOCIENTE "
430  RETURN
500  PRINT "ENTRE COM O PRIMEIRO NUMERO"
510  INPUT A
520  PRINT "ENTRE COM O SEGUNDO NUMERO"
530  INPUT B
540  RETURN
600  PRINT
610  PRINT M$;"DOS DOIS NUMEROS E' ";C
620  RETURN
700  END
```

O programa inicia na linha 10 e segue sem nenhuma novidade até a linha 80, onde dependendo do valor de X, uma das três subrotinas de números 200, 300 ou 400 será executada.

Vamos supor que o número 1, seja escolhido e assim, o programa muda sua seqüência para a linha de número 200 a qual é a primeira linha da subrotina 200.

O computador guarda na memória o valor da linha de

retorno ou seja o número 90.

A linha 200, no entanto, chama uma nova subrotina, a de número 500 e assim novamente o fluxo do programa é mudado e mais uma linha de retorno é guardada na memória do computador: a de número 210.

A subrotina 500 é executada e quando a instrução *RETURN* é executada, o computador busca na memória o último valor da linha de retorno ou seja 210 e continua a execução do programa a partir desta linha. A instrução *RETURN* a seguir retira o valor 90 da memória como ponto de retorno, e a execução do programa segue sem mais novidades.

Rodando, temos:

```
CALCULOS MATEMATICOS

ESCOLHA
1- SOMA DE DOIS NUMEROS
2- PRODUTO DE DOIS NUMEROS
3- QUOCIENTE DE DOIS NUMEROS
?3
ENTRE COM O PRIMEIRO NUMERO
?2405
ENTRE COM O SEGUNDO NUMERO
?12

QUOCIENTE DOS DOIS NUMEROS E' 200.416667
NOVO CALCULO?

1- SIM
2- NAO
?1
CALCULOS MATEMATICOS

ESCOLHA
1- SOMA DE DOIS NUMEROS
2- PRODUTO DE DOIS NUMEROS
3- QUOCIENTE DE DOIS NUMEROS
?2
ENTRE COM O PRIMEIRO NUMERO
?234
ENTRE COM O SEGUNDO NUMERO
?12

PRODUTO DOS DOIS NUMEROS E' 2808
NOVO CALCULO?

1- SIM
2- NAO
?2
```

ENCADEAMENTO DE SUBROTINA

A utilização de uma subrotina dentro de outra subrotina é denominado de *NESTING*. Para que esse encadeamento seja possível, é necessário que o computador seja capaz de armazenar mais de um valor da linha de retorno da subrotina. Cada *NESTING* ou seja cada encadeamento necessita de uma posição de memória para guardar o valor do retorno. O número máximo de *NESTING* possível é característico de cada computador, mas o número máximo é 8. Portanto o encadeamento máximo que você deve usar é subrotina da subrotina da subrotina da subrotina da subrotina da subrotina da subrotina da subrotina, o que é mais que suficiente.

Deve ficar claro que o que é limitado em 8 é o número máximo de encadeamento possível e não o número de subrotinas que pode ser usado em um programa, para o que não existe limite algum.

EXERCÍCIOS

EXERCÍCIO 4.1

Qual a seqüência de execução do programa abaixo ?

```
10 GOTO 100
20 GOTO 30
30 GOTO 50
40 GOTO 20
50 GOTO 90
60 GOTO 110
70 GOTO 80
80 GOTO 60
90 GOTO 70
100 GOTO 40
110 END
```

EXERCÍCIO 4.2

Qual a mensagem que será enviada para o vídeo se escolhermos os números 1,2 nesta ordem, quando o programa abaixo for rodado ?

```
10 PRINT "MENSAGEM SURPRESA"
20 PRINT "ENTRE COM 2 NUMEROS SEPARADOS POR VIRGULA
   ENTRE 1 E 3, PODENDO SEREM REPETIDOS"
30 INPUT A, B
40 ON A GOTO 100, 200, 300
50 GOTO 500
100 ON B GOTO 110, 120, 130
105 GOTO 500
110 LET M$= "EU TE AMO"
115 GOTO 1000
120 LET M$= "VOCE ME AMA?"
125 GOTO 1000
130 LET M$= "NOS NOS AMAMOS"
135 GOTO 1000
200 ON B GOTO 210, 220, 230
205 GOTO 500
210 LET M$= "EU TE ODEIO"
215 GOTO 1000
220 LET M$= "VOCE ME ODEIA?"
225 GOTO 1000
230 LET M$= "NOS NOS ODIAMOS"
235 GOTO 1000
300 ON B GOTO 310, 320, 330
305 GOTO 500
310 LET M$= "EU QUERO CASAR-ME COM VOCE"
315 GOTO 1000
320 LET M$= "VOCE QUER CASAR-SE COMIGO?"
325 GOTO 1000
330 LET M$= "NOS QUEREMOS NOS CASAR"
335 GOTO 1000
500 PRINT "OS NUMEROS DEVEM ESTAR ENTRE 1 E 3"
510 GOTO 20
1000 PRINT
1100 PRINT M$
1200 END
```

EXERCÍCIO 4.3

Mude o programa do exercício anterior de modo que ao invés da instrução *ON GOTO* seja usada a instrução *GOSUB*.

CAPÍTULO 5

AÇÕES CONDICIONAIS

Várias são as oportunidades em nossa vida em que estabelecemos condições para a realização ou não de uma determinada ação. Como exemplos:

"Se chover, então não vou te buscar"

"Se der cara, então você paga a conta"

"Se o tanque estiver com menos de 1/4, então reabasteça"

"Se amanhã chover ou o jogo for televisionado então ficaremos em casa"

Em todas as expressões acima existe uma condição a qual deve ser avaliada e, dependendo do resultado, uma ação será tomada ou não.

A criação literal de uma ação condicional do tipo:

SE (condição) *ENTÃO* (ação)

introduziu na linguagem *BASIC* uma de suas mais poderosas instruções. A palavra em Inglês equivalente ao nosso *SE*, é *IF* e ao *ENTÃO*, é *THEN*, o que veio a criar a instrução:

IF THEN

Esta instrução permite utilizarmos em um programa a ação condicional a que referimos. Seu formato é:

número da linha *IF* (condição) *THEN* (ação)

Se a condição for válida é chamada de *VERDADEIRA* e, então, a ação será realizada. Caso contrário, ou seja, se a condiçao não for válida, é chamada de *FALSA* e a instrução da linha seguinte do programa será executada.

Vamos apresentar e comentar alguns exemplos de *IF THEN*, os quais se destinam apenas a mostrar o formato da instrução. As explicações após cada exemplo objetivam satisfazer

àqueles mais ansiosos. Não entre em pânico se as coisas não estiverem totalmente claras para você, tudo a seu tempo.

Exemplos

```
10 IF A=2 THEN PRINT "VALOR DE A E' 2"
```

Condição: *A = 2*

Ação: *PRINT "VALOR DE A E' 2"*

O computador verifica se o valor da variável A é 2, se esta condição for *VERDADEIRA*, então a ação de enviar a mensagem *"O VALOR DE A E' 2"* será executada. Caso contrário, a próxima linha do programa será executada.

```
100 IF A$= "SIM" THEN GOTO 220
```

Condição: *A$ = "SIM"*

Ação: *GOTO 220*

Se a variável alfanumérica A$ contiver a *"STRING" "SIM"*, então o programa irá para a linha 220. Se não, a próxima linha será executada.

```
30 IF X+3>Z  THEN INPUT Z
```

Condição: *X + 3 > Z*

Ação: *INPUT Z*

Se o valor da variável X somando com 3 for maior que o valor da variável Y, então o computador aguardará a entrada de um valor para a variável Z. Se a condição for *FALSA*, então o programa prossegue na linha seguinte.

```
230 IF J=3 OR B$= "NADA" THEN C$= "PULE SUA VEZ"
```

Condição: *J = 3 OR B$ = "NADA"*

Ação: *LET C$ = "PULE SUA VEZ"*

Se o valor da variável *J* for 3 ou a variável alfanumérica *B$* contiver a *"STRING" NADA*, então a ação será executada, ou seja a *"STRING" PULE SUA VEZ* será atribuída a variável alfanumérica *C$*.

Além de apresentarem o formato, os exemplos mostrados, puderam dar uma "pálida idéia" das possibilidades da instrução *IF THEN* e também deixaram antever que essas possibilidades baseiam-se, principalmente, sobre a extensão das *"CONDIÇÕES"* e das *"AÇÕES"* possíveis. Quanto mais amplo for o campo das condições e das ações, mais poderosa será esta instrução. Portanto, para conhecer esta instrução, é fundamental conhecer bem as condições e as ações que podem ser utilizadas.

CONDIÇAO

A condição a ser estabelecida após a palavra *IF* deverá estar sob a forma de uma expressão possível de ser avaliada pelo computador em termos de *VERDADEIRA* ou *FALSA*. O formato dessa expressão é:

(VALOR, VARIÁVEL OU EXPRESSÃO) SINAL RELACIONAL (VALOR, VARIÁVEL OU EXPRESSÃO).

Devido ao fato de ser formada por duas partes ligadas por sinal que estabelece uma relação, esta expressão é chamada de expressão relacional.

Os sinais de operações relacionais aceitos em *BASIC*, são:

= *IGUAL*

> *MAIOR QUE*

< *MENOR QUE*

=> *IGUAL OU MAIOR QUE*

=< *IGUAL OU MENOR QUE*

<> *DIFERENTE DE*

Vamos apresentar alguns exemplos de operações relacionais e uma breve descrição do significado de cada uma.

NUMÉRICAS

a.) *A = 3*

O computador avalia essa expressão relacional, buscando o valor atual da variável *A* arquivado na memória e comparando-o com *3*. Se for *3*, então a expressão relacional é *VERDADEIRA*, qualquer outro valor implica numa expressão *FALSA*.

b.) *C > 12*

Será avaliada como *VERDADEIRA* sempre que o valor da expressão à esquerda do sinal de operação for igual ou maior que o valor da expressão à direita.

d.) *A < > B*

Será *VERDADEIRA* sempre que o valor da variável *A* for diferente do valor da variável *B*.

ALFANUMÉRICAS

a.) *H$ = "SIM"*

Será considerada *VERDADEIRA* se a variável alfanumérica *H$* contiver a *"STRING" SIM*, outra *"STRING"* qualquer implicará em *FALSA*.

b.) *T$ > "D"*

Será *VERDADEIRA* se a variável alfanumérica *T$* contiver uma *"STRING"* maior que *D*. Por exemplo, se *T$* contiver *"H"*, então a operação relacional será avaliada como *VERDADEIRA*.

VALOR DE UMA "STRING"

Para se fazer comparações entre *"STRINGS"* em termos de maior e menor, é necessário que se defina uma relação de valor entre essas *"STRINGS"*. Esta relação de valor, quando a *"STRING"* fosse apenas uma letra poderia corresponder à ordem alfabética mas, como uma *"STRING"* pode possuir além de letras, números e sinais gráficos (espaço, ponto de interrogação, ponto de exclamação, etc) a seguinte relação de ordem foi definida, estando os valores em ordem crescente:

*(espaço) ! # $ % & ' ( ) * + , - . / 0 1 2 3 4 5 6 7 8 9 : ; , <*

= > ? @ A B C D E F G H I J K L M N O P Q R S T U V X Z

[/] ^ _ ` a b c d e f g h i j k l m n o p q r s t u v x z

O menor valor é representado pelo *ESPAÇO* e o maior valor é o *z*.

A comparação de *"STRINGS"* com dois ou mais caracteres é feita da esquerda para a direita e quando uma *"STRING"* possui menos caracteres que a outra, os caracteres faltantes são preenchidos com espaços à direita. Deste modo, temos que:

"ABD" é maior que "AB"

"BC" é maior que "B"

"C" é maior que "BC"

"ab" é maior que "AB"

Cuidado com o resultado quando *"STRINGS"* contendo algarismos são comparadas, pois, por exemplo, a *"STRING" "3"* é maior que a *"STRING" "198"* .

AÇÃO

A segunda parte da instrução *IF THEN* constitui-se na *AÇÃO* a ser executada pelo computador, caso a condição estabelecida seja *VERDADEIRA*. Essa ação deve ser uma das instruções da linguagem *BASIC*, no entanto, nem todas as instruções podem ser usadas. Aqui, mais uma vez, existe uma dependência da versão *BASIC* que você está utilizando. Deste modo, utilizaremos como anteriormente feito, a condição mínima entre todas as versões.

O conjunto mínimo de instruções que podem ser utilizadas como ação na instrução *IF THEN* é:

PRINT, INPUT, LET, GOTO, GOSUB, RETURN, GET, READ, RESTORE, STOP, PEEK, POKE.

Algumas das instruções acima ainda não foram anali

sadas, mas não se preocupe, não serão esquecidas. Para verificar se outras instruções podem fazer parte desta relação, na versão *BASIC* do seu computador, a receita é a mesma de sempre: escreva um programa com a instrução *IF THEN* utilizando-a e rode o programa. Se isto não provocar erros na execução do programa, então acrescente esta instrução na linha das possíveis.

EXEMPLOS DE USO

Vamos apresentar alguns programas utilizando a instrução *IF THEN*. Não deixe de entrá-los no seu computador, rodá-los e procure introduzir algumas variações ou modificações nos mesmos.

Exemplo 5.1

```
10 PRINT "SOMA DE DOIS NUMEROS"
20 PRINT
30 PRINT "ENTRE COM O PRIMEIRO NUMERO"
40 INPUT A
50 PRINT "ENTRE COM O SEGUNDO NUMERO"
60 INPUT B
70 PRINT "A SOMA DOS DOIS NUMEROS E' " ; A+B
80 PRINT "NOVA SOMA?  RESPONDA SIM OU NAO"
90 INPUT R$
100 IF R$= "NAO" THEN GOTO 140
110 IF R$= "SIM" THEN GOTO 20
120 PRINT "RESPONDA SIM OU NAO"
130 GOTO 90
140 PRINT
150 PRINT "TCHAU. ATE UMA PROXIMA VEZ"
160 END
```

Rodando, temos:


```
RUN

SOMA DE DOIS NUMEROS

ENTRE COM O PRIMEIRO NUMERO
?2345
ENTRE COM O SEGUNDO NUMERO
?1253
A SOMA DOS DOIS NUMEROS E' 3598
NOVA SOMA? RESPONDA SIM OU NAO
?SIM

ENTRE COM O PRIMEIRO NUMERO
?11111
ENTRE COM O SEGUNDO NUMERO
?0.222
A SOMA DOS DOIS NUMEROS E' 11111.222
NOVA SOMA? RESPONDA SIM OU NAO
?NAO

TCHAU. ATE UMA PROXIMA VEZ.
```

Até a linha número 70 inclusive, este programa não apresenta novidade nenhuma. As demais linhas, no entanto, devem ser comentadas.

80 - A questão *"NOVA SOMA ? RESPONDA SIM OU NÃO"* é enviada para o vídeo.

90 - A execução do programa é interrompida e o computador aguarda que uma *"STRING"* seja entrada via teclado para associá-la a variável *R$*.

100 - Se a *"STRING"* entrada para *R$* for *NÃO*, então o programa continua na linha 150, ou seja, a mensagem *"TCHAU, ATÉ UMA PRÓXIMA VEZ"* é enviada para o vídeo e o programa segue na linha 160, onde termina.

110 - Se a *"STRING"* entrada para *R$* for *SIM*, então o programa volta na linha 20 e novo cálculo de soma é feito e a seqüência é repetida até que se entre *NÃO*.

120 - Se a *"STRING"* entrada não for nem *SIM* nem *NÃO*, então nenhum dos desvios acima (GOTO 140 e GOTO 20) ocorrem e, portanto, esta linha do programa é executada. A mensagem *"RESPONDA SIM OU NÃO"*, lembrando que a resposta deve ser obrigatoriamente ou *SIM* ou *NÃO*, é enviada para o vídeo.

130 - O programa retorna a linha 90, onde uma nova *"STRING"* é aguardada para ser associada à variável *R$*.

140 - Envia uma linha em branco para o vídeo.

150 - Envia a mensagem final do programa.

160 - Última linha do programa.

Exemplo 5.2

```
10 PRINT "ENTRE COM DOIS NUMEROS E EU DIREI QUAL
   DELES E' O MAIOR"
20 PRINT
30 PRINT "QUAL O PRIMEIRO NUMERO?"
40 INPUT A
50 PRINT "QUAL O SEGUNDO NUMERO?"
60 INPUT B
70 IF A > B THEN GOTO  110
80 IF A < B THEN GOTO  130
90 ID A=B THEN PRINT "NAO VALE.  OS DOIS NUMEROS SAO
   IGUAIS"
100 GOTO 140
110 PRINT "O PRIMEIRO NUMERO E' O MAIOR"
120 GOTO 140
130 PRINT "O SEGUNDO NUMERO E' O MAIOR"
140 END
```

Rodando, temos:

```
ENTRE COM DOIS NUMEROS E EU DIREI QUAL DELES E' O
MAIOR

QUAL O PRIMEIRO NUMERO?
?10
QUAL O SEGUNDO NUMERO?
?10
NAO VALE. OS DOIS NUMEROS SAO IGUAIS.
```

O exemplo 5.3 a seguir é uma versão do exemplo 5.2 utilizando variáveis alfanuméricas.

Exemplo 5.3

```
100 PRINT "ESCREVA DUAS PALAVRAS E EU DIREI QUAL
    DELAS VEM EM PRIMEIRO LUGAR NO DICIONARIO"
110 PRINT
120 PRINT "PRIMEIRA PALAVRA?"
130 INPUT A$
140 PRINT "SEGUNDA PALAVRA?"
150 INPUT B$
160 IF A$ > B$ THEN GOTO 200
170 IF A$ < B$ THEN GOTO 220
180 IF A$ = B$ THEN PRINT "NAO VALE.  VOCE ENTROU A
    MESMA PALAVRA DUAS VEZES"
190 GOTO 230
200 PRINT "A SEGUNDA VEM PRIMEIRO"
210 GOTO 230
220 PRINT "A PRIMEIRA VEM PRIMEIRO"
230 END
```

Rodando, temos:

```
ESCREVA DUAS PALAVRAS E EU DIREI QUAL DELAS VEM EM
PRIMEIRO LUGAR NO DICIONARIO

PRIMEIRA PALAVRA
?CASA
SEGUNDA PALAVRA
?MENINO
A PRIMEIRA VEM PRIMEIRO
```

ADEUS MUNDO CRUEL...

Vamos deixar as coisas sérias de lado e vamos testar um pouco de sua coragem. Que tal uma *"ROLETA RUSSA ?"*

Bem se você não sabe o que é, vamos explicar:

A *ROLETA RUSSA* é um jogo em que só as pessoas extremamente corajosas (e também extremamente idiotas) se aventuram.

Pegue um revolver tipo tambor, coloque uma única bala, gire o tambor, feche-o e encoste o revólver na sua cabeça e puxe o gatilho. Se você conseguir fazer isso por 10 vezes, você é um completo idiota vitorioso.

Felizmente, temos um computador e podemos testar nossa *"coragem"*, sem risco de vida. Entre e rode o exemplo 5.4 e divirta-se

Exemplo 5.4

```
10  PRINT "ROLETA RUSSA"
20  PRINT
30  PRINT "AQUI ESTA UM REVOLVER"
40  PRINT
50  PRINT "VOCE TEM 10 CHANCES DE"
52  PRINT "ESTOURAR  SEUS  MIOLOS"
60  PRINT
70  PRINT "TECLE 1 PARA GIRAR O TAMBOR"
72  PRINT "E PUXAR O GATILHO"
80  PRINT "TECLE 2 PARA DESISTIR"
90  LET N = 0
100  INPUT I
110  IF I = 1 THEN  GOTO 150
120  IF I = 2 THEN  GOTO 330
130  PRINT "VOCE ESTA MUITO NERVOSO - TECLE 1 OU 2 "
140  GOTO 100
150  LET N = N + 1
160  IF  RND (1) > .857 THEN  GOTO 220
170  IF N > 9 THEN  GOTO 370
180  PRINT
190  PRINT "     CLICK"
200  PRINT "FALTAM ";10 - N;" TENTATIVAS"
210  GOTO 100
220  PRINT "      BANG!!! VOCE ESTA MORTO!"
230  PRINT "ENVIAREMOS UMA COROA DE FLORES"
240  PRINT "E SUA FAMILIA SERA AVISADA"
250  PRINT
260  PRINT "MAIS UM DISPOSTO A ESTOURAR OS MIOLOS?"
270  PRINT "TECLE S OU N"
280  INPUT A$
290  IF A$ = "S" THEN  GOTO 30
300  PRINT
310  PRINT "QUE PENA. ADORO SANGUE"
320  GOTO 390
330  PRINT
340  PRINT "SEU MARICAS! COM MEDO DE"
342  PRINT "UMA BALINHA NA CUCA"
350  GOTO 260
360  PRINT
370  PRINT "VOCE VENCEU!!"
380  GOTO 260
390  END
```


Rodando, temos:

```
ROLETA RUSSA

AQUI ESTA UM REVOLVER

VOCE TEM 10 CHANCES DE
ESTOURAR  SEUS  MIOLOS

TECLE 1 PARA GIRAR O TAMBOR
E PUXAR O GATILHO
TECLE 2 PARA DESISTIR
?3
VOCE ESTA MUITO NERVOSO - TECLE 1 OU 2
?1

     CLICK
FALTAM 9 TENTATIVAS
?1

     CLICK
FALTAM 8 TENTATIVAS
?2

SEU MARICAS! COM MEDO DE
UMA BALINHA NA CUCA
MAIS UM DISPOSTO A ESTOURAR OS MIOLOS?
TECLE S OU N
?S
AQUI ESTA UM REVOLVER

VOCE TEM 10 CHANCES DE
ESTOURAR  SEUS  MIOLOS

TECLE 1 PARA GIRAR O TAMBOR
E PUXAR O GATILHO
TECLE 2 PARA DESISTIR
?1

     CLICK
FALTAM 9 TENTATIVAS
?1

     CLICK
FALTAM 8 TENTATIVAS
?1

     CLICK
FALTAM 7 TENTATIVAS
?1

     CLICK
FALTAM 6 TENTATIVAS
?1
      BANG!!! VOCE ESTA MORTO!
ENVIAREMOS UMA COROA DE FLORES
E SUA FAMILIA SERA AVISADA
```

```
MAIS UM DISPOSTO A ESTOURAR OS MIOLOS?
TECLE S OU N
?N

QUE PENA. ADORO SANGUE
```

A única linha do programa que vamos comentar é a 160. Nesta linha, utilizamos uma função matemática em *BASIC* que é *RND (1)*. Esta função gera um número qualquer entre 0 e 1, portanto um número fracionário, cada vez que é utilizada e cujo valor é imprevisível, podendo ser usada para sorteios. *RND* vem de *RANDOM* em Inglês que significa aleatório, imprevisível.

O valor gerado pela função *RND(1)* é comparado com .857 que corresponde ao valor 6/7 ou seja seis das sete cavidades do tambor do revólver vazias.

MÚLTIPLAS CONDIÇÕES

Você pode estabelecer mais de uma condição na instrução *IF THEN* através do uso de operações lógicas.

Os operadores lógicos utilizáveis são:

NOT - Não. Inverte (nega) o valor da expressão racional. Se *FALSA* transforma em *VERDADEIRA* e vice-versa.

OR - Ou. Interliga duas expressões racionais. O resultado será *VERDADEIRO* se qualquer uma das expressões racionais for *VERDADEIRA*.

AND - E. Interliga duas expressões racionais. O resultado será *VERDADEIRO*, somente se as duas funções racionais forem *VERDADEIRAS* simultaneamente.

Exemplos:

a.) *NOT A = 3*

Será *VERDADEIRA* quando *A* for diferente de *3* ou seja é equivalente a *A < > 3*.

b.) *A$ = "SIM" OR A$ = "S"*

Será *VERDADEIRA* quando *A$* for igual a *"SIM"* ou igual a *"S"*.

c.) *R$ = "PAR" AND B = 2*

Será *VERDADEIRA* quando *R$* for igual a *"PAR"* e *"B"* for igual a *2* simultaneamente.

UM PROGRAMA ÚTIL E DIVERTIDO

Que tal saber qual o dia da semana de uma data qualquer ? O exemplo 5.5 abaixo, permite que o dia da semana de qualquer data seja conhecido.

Exemplo 5.5

```
10  PRINT "DIA DA SEMANA"
20  PRINT
30  PRINT "ENTRE COM O DIA, MES E ANO NO FORMATO:"
32  PRINT "DD,MM,AAAA"
40  PRINT "PARA TERMINAR ENTRE 0,0,0"
50  PRINT
60  INPUT D,M,A
70  IF D = 0 OR M = 0 OR A = 0 THEN  GOTO 220
80  IF M > 2 THEN  GOTO 110
90  LET M = M + 12
100  LET A = A - 1
110  LET N = D + 2 * M +  INT (.6 * (M + 1)) + A +  INT
     (A / 4) -  INT (A / 100) +  INT (A / 400) + 2
120  LET N =  INT ((N / 7 -  INT (N / 7)) * 7 + .5)
130  IF N = 0 THEN N$ = "SABADO"
140  IF N = 1 THEN N$ = "DOMINGO"
150  IF N = 2 THEN N$ = "SEGUNDA"
160  IF N = 3 THEN N$ = "TERCA"
170  IF N = 4 THEN N$ = "QUARTA"
180  IF N = 5 THEN N$ = "QUINTA"
190  IF N = 6 THEN N$ = "SEXTA"
200  PRINT N$
210  GOTO 20
220  PRINT "VOLTE SEMPRE"
230  END
```

Rodando o programa, temos:

```
DIA DA SEMANA

ENTRE COM O DIA, MES E ANO NO FORMATO:
DD,MM,AAAA
PARA TERMINAR ENTRE 0,0,0

?22,4,1500
DOMINGO

ENTRE COM O DIA, MES E ANO NO FORMATO:
DD,MM,AAAA
PARA TERMINAR ENTRE 0,0,0

?7,9,1822
SABADO

ENTRE COM O DIA, MES E ANO NO FORMATO:
DD,MM,AAAA
PARA TERMINAR ENTRE 0,0,0

?31,12,1999
SEXTA

ENTRE COM O DIA, MES E ANO NO FORMATO:
DD,MM,AAAA
PARA TERMINAR ENTRE 0,0,0

?0,0,0
VOLTE SEMPRE
```


VAMOS ADIVINHAR UM NÚMERO?

Este exemplo, além de reforçar os conhecimentos da instrução *IF THEN*, mostra um caso particular dessa instrução.

Quando a ação for a instrução *GOTO*, esta pode ser omitida ou então a palavra *THEN*, mas nunca ambas.

O jogo que vamos apresentar consiste em *"encurralar"* e adivinhar um número gerado aleatoriamente pelo computador entre 1 e 100. Você deve entrar com dois números e o computador dirá se o número gerado é menor, maior ou se está *"encurralado "* entre os dois números escolhidos por você. Se um desses dois números coincidir com o número do computador, você ganhou o jogo, desde que você faça isso em sete tentativas.

Exemplo 5.6

```
10  PRINT "ENCURRALADO"
20  PRINT
30  LET X =  INT (100 *  RND (1)) + 1
40  LET Y = 0
50  LET Y = Y + 1
60  PRINT "TENTATIVA NUMERO ";Y
70  PRINT "ENTRE COM DOIS NUMEROS"
80  INPUT A,B
100  IF X = A OR X = B THEN 250
110  IF X < A AND X < B GOTO 150
120  IF X > A AND X > B THEN  GOTO 170
125  IF Y > 6 GOTO 190
130  PRINT "O MEU NUMERO ESTA ENCURRALADO"
140  GOTO 180
150  PRINT "O MEU NUMERO E' MENOR QUE OS SEUS"
160  GOTO 180
170  PRINT "O MEU NUMERO E' MAIOR QUE OS SEUS"
180  IF Y < 6 GOTO 50
190  PRINT "VOCE PERDEU!"
195  GOTO 260
200  PRINT
210  PRINT "NOVA TENTATIVA? S OU N "
220  INPUT Z$
230  IF Z$ = "S" THEN 20
240  GOTO 280
250  PRINT "VOCE GANHOU NA TENTATIVA NUMERO ";Y
260  PRINT "O MEU NUMERO E' ";X
270  GOTO 210
280  PRINT "AGRADECEMOS A PREFERENCIA"
290  END
```

Rodando, temos:

```
ENCURRALADO

TENTATIVA NUMERO 1
ENTRE COM DOIS NUMEROS
?20,80
O MEU NUMERO E' MENOR QUE OS SEUS
TENTATIVA NUMERO 2
ENTRE COM DOIS NUMEROS
?10,18
O MEU NUMERO E' MENOR QUE OS SEUS
TENTATIVA NUMERO 3
ENTRE COM DOIS NUMEROS
?1,5
O MEU NUMERO E' MAIOR QUE OS SEUS
TENTATIVA NUMERO 4
ENTRE COM DOIS NUMEROS
?6,8
O MEU NUMERO E' MAIOR QUE OS SEUS
TENTATIVA NUMERO 5
ENTRE COM DOIS NUMEROS
?9,9
VOCE GANHOU NA TENTATIVA NUMERO 5
O MEU NUMERO E' 9
NOVA TENTATIVA? S OU N
?N
AGRADECEMOS A PREFERENCIA
```

SENÃO...

Algumas versões de *BASIC* incluem na instrução *IF THEN* uma terceira palavra em Inglês: *ELSE*, que significa senão, dando origem ao formato:

IF (condição) *THEN* (ação) *ELSE* (ação)

Este formato permite o estabelecimento de duas ações. A primeira, logo após a *THEN*, será executada se a condição for verdadeira; e a segunda, após *ELSE*, será executada se a condição não for verdadeira.

Exemplo:

```
50 IF A$= "S" THEN GOTO 100 ELSE GOTO 200
```

Se *A$* contiver a *"STRING" "S"* então o programa continuará na linha 100; senão contiver, continuará na linha 200.

Embora esse formato seja muito útil, nós não o utilizaremos neste livro por não ser um formato de uso universal a todas as versões *BASIC*.

EXERCÍCIOS

EXERCÍCIO 5.1

Escreva no formato *IF THEN* as seguintes condições e ações abaixo:

a.) Se a variável A for maior que 10, então execute a linha 100 do programa.

b.) Se A$ contiver *"SIM"*, então mude o valor de B para 100.

c.) Se R1$ contiver *"PAGO"* e R2$ contiver *"PASSO"*, envie a mensagem *"FIM DO JOGO"*.

EXERCÍCIO 5.2

Para as condições abaixo, defina quando as mesmas resultam em verdadeiras.

a) `IF T= > 1000 or T < 1100 THEN...`

b) `IF J$ <> "NAO" AND J$ <> "SIM" THEN....`

EXERCÍCIO 5.3

Qual o erro de cada uma dessas linhas abaixo:

a) `IF R$ <> "S" OR R$ <> "N" THEN...`

b) `IF R$= "S" AND R$= "N" THEN...`

CAPÍTULO 6

OPERAÇÕES REPETIDAS CONTROLADAS

Nós já vimos em capítulos anteriores, programas que possuiam *"LOOPS"*, ou seja, programas que numa determinada linha, retornavam a uma linha anterior e voltavam a executar as instruções entre essas duas linhas. Vimos também que esses *"LOOPS"* podem constituir-se numa seqüência infinita e vimos que podíamos controlar a permanência ou não no *"LOOP"* através de instruções condicionais.

Nós vamos ver, neste capítulo, uma instrução que permite criar um *"LOOP"* cuja duração em número de repetição ou ciclos, pode ser previamente definida. Esta instrução é dividida em duas partes as quais indicam o início e o término da área do programa onde as instruções serão repetidas.

O formato desta instrução é:

Primeira parte da instrução:

número da linha *FOR* (variável) = (constante) *TO* (constante)

Segunda parte da instrução:

número da linha *NEXT* (variável)

Exemplo 6.1

```
120 FOR I= 1 TO 20
130 ..... Estas Instruções
140 ..... serão executadas
150 ..... 20 vezes
160 NEXT I
```

A primeira parte da instrução define o nome de uma variável denominada *INDICE* (e por isso comumente de nome I), o seu valor inicial e o seu valor final.

A segunda parte da instrução incrementa este índice unitariamente (soma 1 ao seu valor) e verifica se o valor final declarado na primeira parte já foi ultrapassado. Enquanto isso não ocorre, a execução do programa retorna a linha seguin

te a da primeira parte da instrução. Quando o valor do índice for maior que o valor final declarado, o programa continua a partir da instrução seguinte a instrução *NEXT*.

Exemplo 6.2

```
10 FOR I= 1 TO 10
20 PRINT "ASDFGH"
30 NEXT I
40 END
```

Rodando, temos:

```
RUN

ASDFGH
ASDFGH
ASDFGH
ASDFGH
ASDFGH
ASDFGH
ASDFGH
ASDFGH
ASDFGH
ASDFGH
```

Este exemplo mostra como a execução repetida de uma determinada instrução pode ser controlada. A instrução *PRINT "ASDFGH"* foi executada 10 vezes, pois, como o valor inicial do índice foi definido com 1 e o final como 10, e a cada *"LOOP"* o valor desse índice foi aumentado de 1, foram necessários 10 *"LOOPS"* para que o valor final 10 fosse ultrapassado. A análise do programa linha a linha é:

10 - O valor inicial do *ÍNDICE* é definido como 1 e o final como 10.

20 - A *"STRING" ASDFGH* é enviada para o vídeo.

30 - O valor do índice é incrementado de 1 e o programa retorna a linha 20. Este *"LOOP"* permanece até que o valor do índice ultrapasse 10 quando o programa segue para a linha 40.

40 - Linha final do programa.

Mudando-se o valor inicial e final do índice, obviamente, poderemos ter um resultado diferente. Mude a linha 10 do exemplo 6.2 para:

```
10 FOR I= 8 TO 10
```

Listando o programa, temos:

Exemplo 6.3

```
10 FOR I= 8 TO 10
20 PRINT "ASDFGH"
30 NEXT I
40 END
```

Rodando, temos:

```
RUN

ASDFGH
ASDFGH
ASDFGH
```

Como o valor inicial de I foi definido como 8, tivemos apenas 3 linhas com a *"STRING" ASDFGH*, a primeira correspondente ao valor 8 do índice, a segunda a 9 e a terceira a 10.

LIMPANDO A TELA DO VIDEO

Existe em algumas versões de *BASIC*, uma função que permite a limpeza da tela do vídeo, resultando em uma tela em branco, ou seja, sem nenhum caractere.

Seu formato é:

número de linha *HOME*

Como essa instrução não consta de todas as versões de *BASIC* não a utilizaremos em nossos exemplos. No entanto, vamos apresentar no Exemplo 6.4, um modo de se obter esse resultado, sem utilizar a instrução *HOME*.

Exemplo 6.4

```
10 FOR I= 1 TO 42
20 PRINT
30 NEXT I
40 END
```

Rodando este programa, você observerá que as linhas constantes da tela do vídeo irão subindo até desaparecerem, resultando uma tela *"LIMPA"* e com o cursor na última linha.

Isso ocorre porque o programa envia para a tela do vídeo 42 linhas em branco, as quais irão substituir as linhas atuais do seu vídeo. O número de linhas a serem enviadas e, portanto, o valor final do índice do *"LOOP"* depende do número de linhas que compõem o texto no vídeo de seu computador. Existem computadores com 20, 22, 30, 40 etc., linhas por telas de vídeo. Como utilizamos o número 42, estamos abrangendo todos os casos possíveis. Você poderá mudar o valor do índice para o caso adequado ao seu computador e utilizar esse exemplo como uma subrotina em seus programas para limpeza da tela.

O INDICE É UMA VARIÁVEL

A variável utilizada como índice é exatamente igual a qualquer outra variável e é chamada de *ÍNDICE* simplesmente para facilitar sua identificação. Do mesmo modo, o nome adotado para a variável I, pode ser qualquer outro nome permitido para variáveis.

Sendo uma variável, seu valor pode ser usado no programa, em outras instruções, mostrando-se particularmente útil quando, no programa, o uso de um valor crescente ou decrescente de uma variável for necessário.

Vamos escrever um programa que envie para o vídeo os 5 primeiros números inteiros. Um primeiro exemplo desse programa é:

Exemplo 6.5

```
100 LET A= 0
200 FOR I= 1 TO 5
300 LET A= A + 1
400 PRINT A
500 NEXT I
600 END
```

Rodando, temos:

```
RUN

1
2
3
4
5
```

O programa está correto, mas nós podemos escrever um programa bem mais simples se utilizarmos o valor do índice diretamente, como mostrado no exemplo 6.6.

Exemplo 6.6

```
100 FOR A= 1 TO 5
200 PRINT A
300 NEXT A
400 END
```

Rodando, teremos:

```
RUN

1
2
3
4
5
```

Neste exemplo, ao invés de utilizarmos uma outra variável, a qual é incrementada a cada *"LOOP"*, o próprio valor do índice é enviado para o vídeo, simplificando, deste modo, o programa, tornando-o mais rápido e ocupando menos espaço na memória.

O VALOR DO ÍNDICE EM CÁLCULOS

Quando você utilizar o valor do índice diretamente em cálculos no programa, você deverá estar atento ao fato de que se o valor do índice for modificado como resultado desses cálculos, o número de *"LOOPS"* do programa estará sendo afetado.

Suponhamos que você queira escrever um programa que envie para o vídeo os 5 primeiros números pares. Uma primeira idéia, não muito feliz, como veremos, é acrescentar a linha 150 abaixo, ao exemplo 6.6.

```
150 LET  A= A * 2
```

o que listando, resulta em:

Exemplo 6.7

```
100 FOR A= 1 TO 5
150 LET A= A * 2
200 PRINT A
300 NEXT A
400 END
```

O resultado do programa rodado, é:

```
RUN

2
6
```

Obviamente, temos aí um *"BUG"*. O resultado não foi o esperado. Vamos analisar o programa para descobrirmos onde se *"ESCONDEU"* esse *"BUG"*.

O programa inicia com o valor do índice A igual a 1, o qual é multiplicado por 2 pela instrução da linha 150, e é enviado para o vídeo pela linha 200, resultando no vídeo o valor 2. A instrução *NEXT A* incrementa esse valor, resultando, portanto, 3, o qual é comparado com o valor final 5. Como esse valor ainda não foi ultrapassado, o programa retorna a linha 150, onde o valor 3, de A é multiplicado por 2, resultando 6, o qual é enviado para o vídeo. A instrução *NEXT A* novamente incrementa o valor de A, resultando 7, que ultrapassa o valor final, sendo executada, neste caso, a instrução *END*, terminando a execução do programa.

Como você deve ter notado, o erro do programa constitui em modificar o valor do índice na operação da linha 150. Para corrigirmos esse programa, envie as seguintes linhas:

```
150 LET B= A * 2
200 PRINT B
```

Listando, você terá:

Exemplo 6.8

```
100 FOR A= 1 TO 5
150 LET B= A * 2
200 PRINT B
300 NEXT A
400 END
```

Rodando, teremos:

```
RUN

2
4
6
8
10
```

Vamos apresentar a seguir, mais um exemplo de programas, onde o valor do índice é usado em cálculos sem que isto afete o número de ciclos de *"LOOP"*. O exemplo não será comentado e mais uma vez estamos sugerindo que você faça, por si mesmo a análise.

Exemplo 6.9

```
10  PRINT "TABUADA DE MULTIPLICACAO"
20  PRINT
30  PRINT "TABUADA DE QUAL NUMERO?"
40  INPUT N
50  PRINT
60  PRINT "TABUADA DE MULTIPLICACAO DO NUMERO ";N
70  PRINT
80  FOR I = 1 TO 10
90  PRINT N;" X ";I;" = ";N * I
100 NEXT I
110 END
```

Rodando, você terá:

```
TABUADA DE MULTIPLICACAO

TABUADA DE QUAL NUMERO?
?6

TABUADA DE MULTIPLICACAO DO NUMERO 6

6 X 1 = 6
6 X 2 = 12
6 X 3 = 18
6 X 4 = 24
6 X 5 = 30
6 X 6 = 36
6 X 7 = 42
6 X 8 = 48
6 X 9 = 54
6 X 10 = 60
```

VALORES INICIAIS E FINAIS DO INDICE COMO VARIAVEL

Quando estabelecemos o formato da instrução *FOR...TO*, os valores inicial e final do índice foram definidos como uma constante. Contudo, estes valores podem ser representados também por variáveis que implica no seguinte formato:

número da linha *FOR* (variável) = (variável) *TO* (variável)

Uma das vantagens que esse formato possibilita é o fornecimento dos valores inicial e final do índice via teclado, possibilitando, assim, a escolha desses valores externamente ao programa.

Como primeiro exemplo, vamos escrever um programa que calcula a soma dos N primeiros números inteiros.

Exemplo 6.10

```
100 PRINT "SOMA DOS N PRIMEIROS NUMEROS INTEIROS"
110 LET S= 0
120 PRINT "QUANTOS NUMEROS?"
130 INPUT N
140 FOR A= 1 TO N
150 LET S= S + A
160 NEXT A
170 PRINT "A SOMA DOS ";N;" PRIMEIROS NUMEROS E' "; S
180 END
```


Rodando o programa, temos:

```
RUN

SOMA DOS N PRIMEIROS NUMEROS
QUANTOS NUMEROS?
?10
A SOMA DOS 10 PRIMEIROS NUMEROS E' 55
```

Analisando o programa linha a linha, temos:

100 - Esta linha envia para o vídeo a mensagem inicial do programa, indicando a finalidade do mesmo.

110 - A variável *S* onde iremos acumular a soma, é igualada a zero.

120 - Envia para o vídeo a questão *"QUANTOS NÚMEROS ?"*

130 - O programa aguarda que o valor da variável *N* seja entrado via teclado.

140 - Instrução inicial do *"LOOP"*. O valor inicial do índice *A* é definido como 1 e o final é igualado ao valor da variável *N*, entrado via teclado na linha anterior.

150 - Esta linha do programa faz a soma acumulativa dos índices o que corresponde a soma dos números inteiros que queremos obter. No primeiro ciclo de *"LOOP"* temos S = 0 e A = 1, donde S fica com o valor 1. No segundo ciclo, temos S = 1 e A = 2, portanto S fica sendo 3 e assim por diante, até que o número de ciclos escolhido termine.

160 - Instrução final do *"LOOP"*.

170 - A mensagem declarando o valor da soma obtida é enviada para o vídeo.

180 - Instrução final do programa.

No próximo exemplo, os valores inicial e final do índice do *"LOOP"* são fornecidos externamente ao programa, via teclado.

Este programa fornece uma tabela de conversão de graus centígrados em fahrenheit, grau em grau com o valor inicial e final da tabela, escolhido via teclado.

Exemplo 6.11

```
100 PRINT "TABELA CENTIGRADOS FAHRENHEIT "
200 PRINT
300 PRINT "QUAL O VALOR INICIAL DA TABELA?"
400 INPUT A
500 PRINT "QUAL O VALOR FINAL DA TABELA?"
600 INPUT B
700 PRINT
800 PRINT "CENTIGRADOS", "FAHRENHEIT"
900 FOR I = A TO B
1000 PRINT I, (9/5 * I)+ 32
1100 NEXT I
1200 END
```

Rodando o programa, temos:

```
RUN

TABELA CENTIGRADOS FAHRENHEIT

QUAL O VALOR INICIAL DA TABELA?
?-5
QUAL O VALOR FINAL DA TABELA?
?5

CENTIGRADO      FAHRENHEIT
-5              23
-4              24.8
-3              26.6
-2              28.4
-1              30.2
0               32
1               33.8
2               35.6
3               37.6
4               39.2
5               41
```

Escolhemos um valor negativo para início da tabela para que você pudesse saber ser possível utilizar números negativos.

MUDANDO O INCREMENTO DO ÍNDICE

Várias versões de *BASIC* permitem a modificação do incremento do índice de unitário para qualquer outro valor inteiro, positivo ou negativo.

Não podem ser usados valores fracionários, bem como incremento zero.

O formato da instrução neste caso é modificado para:

(número da linha) *FOR* (variável) = (constante) *TO* (constante) *STEP* (constante)

O incremento que se quer dar ao índice, a cada ciclo do *"LOOP"* é definido após a palavra *STEP* (degrau em Inglês). Aqui também, o valor do degrau poderá ser uma variável.

Exemplo 6.12

```
10 FOR I = 0 TO 100 STEP 10
20 PRINT I
30 NEXT I
40 END
```

Rodando, temos:

```
RUN

0
10
20
30
40
50
60
70
80
90
100
```

A cada *"LOOP"*, o índice foi incrementado de 10, iniciando com o valor 0 e terminado em 100.

O próximo exemplo define todos os múltiplos de um dado número até 100.

Exemplo 6.13

```
10 PRINT "MULTIPLOS DE UM NUMERO"
20 PRINT
30 PRINT "ENTRE COM UM NUMERO E LHE DAREI TODOS SEUS
   MULTIPLOS MENORES QUE CEM"
40 PRINT
50 PRINT "QUAL NUMERO?"
60 INPUT N
70 PRINT
80 PRINT "MULTIPLOS DE "¦N;" MENORES QUE CEM"
90 PRINT
100 FOR I = N TO 100 STEP N
110 PRINT I
120 NEXT I
130 END
```

Rodando o programa, temos:

```
RUN

MULTIPLOS DE UM NUMERO

ENTRE COM UM NUMERO E LHE DAREI TODOS SEUS MULTIPLOS
MENORES QUE CEM

QUAL NUMERO?
?23

MULTIPLOS DE 23 MENORES QUE CEM

23
46
69
92
```

O valor do incremento pode ser negativo, sendo obrigatório, neste caso, que o valor inicial seja maior que o valor final.

Exemplo 6.14

```
10 PRINT "INCREMENTO NEGATIVO"
20 PRINT
30 FOR I = 10 TO 0 STEP -1
40 PRINT I
50 NEXT I
60 END
```

Rodando o programa, temos:

```
RUN

INCREMENTO NEGATIVO

10
9
8
7
6
5
4
3
2
1
0
```

"LOOPS" DENTRO DE "LOOP"

Um programa pode conter, se necessário, vários *"LOOPS"*. No entanto estes *"LOOPS"* ou devem ser isolados entre si, ou devem estar contidos um no outro. Não pode haver *"LOOPS"* se cruzando em um programa.

Nós vamos apresentar exemplos de *"LOOPS"* de cada caso. A análise de cada um ficará a seu cargo. No exemplo 6.17, os *"LOOPS"* se cruzam. Verifique rodando-os, qual a mensagem de erro que o seu computador enviará.

Exemplo 6.15

```
10 PRINT "LOOPS ISOLADOS"
20 FOR I= 1 TO 5
30 PRINT I
40 NEXT I
50 FOR A= 1 TO 3
60 PRINT A
70 NEXT A
80 END
```

Rodando o programa, temos:

```
RUN

LOOPS ISOLADOS

1
2
3
4
5
1
2
3
```

Exemplo 6.16

```
10 PRINT "LOOP B INTERNO AO A"
20 FOR A= 1 TO 3
30 FOR B= 1 TO 5
40 PRINT A;B
50 NEXT B
60 NEXT A
70 END
```

```
RUN

LOOP B INTERNO AO A

 11
 12
 13
 14
 15
 21
 22
 23
 24
 25
 31
 32
 33
 34
 35
```

Exemplo 6.17

```
10 PRINT "EXEMPLO COM ERRO - LOOP CRUZADO"
20 FOR A= 1 TO 3
30 FOR B= 1 TO 5
40 PRINT A;B
50 NEXT A
60 NEXT B
70 END
```

Rodando o programa, temos:

```
RUN

EXEMPLO COM ERRO - LOOP CRUZADO

 11
NEXT WITHOUT FOR ERROR
```


EXERCÍCIOS

EXERCÍCIO 6.1

Escreva um programa que calcule o fatorial de um número.

EXERCÍCIO 6.2

Reescreva o Exemplo 6.8, simplificando-o e não utilizando a variável B.

EXERCÍCIO 6.3

Escreva um programa que apresente ou a tabuada de multiplicação ou a de soma, por escolha via teclado.

EXERCÍCIO 6.4

Escreva um programa que calcule a equação $y = 3x^2 - 2x - 6$ para os valores de x de -10 a 10, com incremento unitário.

CAPÍTULO 7

ORGANIZANDO INFORMAÇÕES

Mesmo que você não tenha utilizado em matemática, variáveis indexadas, várias vezes você tem resolvido problemas que, se traduzidos em uma linguagem matemática, fariam uso das mesmas.

Como exemplo dessas situações, temos:

"Some somente as 12 primeiras parcelas"

"Procure da ficha 132 a 643, um paciente de nome Marcos"

"Relacione os 15 primeiros clientes"

"As 5 primeiras anotações estão corretas, as demais estão erradas"

O que existe de comum a todos esses exemplos é uma organização seqüencial de parcelas, fichas, clientes e anotações associadas a um número de ordem.

Esta organização seqüencial associada a um número de ordem permite que se defina claramente um elemento de um conjunto através de seu nome e de seu primeiro número de ordem.

Assim, se quisermos indicar uma das 12 parcelas do exemplo *"some somente as 12 primeiras parcelas"*, diríamos:

Veja a terceira parcela

Do mesmo modo, considerando os outros exemplos, cada elemento poderá ser identificado através de uma designação e de um número de ordem.

Analise a ficha número 153.

Qual o nome do cliente número 5 ?

Corrija a décima anotação.

Esse tipo de organização nos leva a uma imagem da mesma, como uma fila, pelo fato dos elementos serem colocados seqüêncialmente, um após o outro. Uma gaveta de arquivo com

100 pastas numeradas em seqüência de 1 a 100 é um exemplo real de uma organização deste tipo. E para esta gaveta de arquivo aplica-se perfeitamente o conceito de *ÍNDICE*. Para se buscar uma determinada informação que se sabe estar na pasta de número de ordem 35, basta dizer:

PEGUE A PASTA 35 e, se convencionado, abreviar pasta pela letra *P*, poder-se-ia dizer: *PEGUE A P 35*.

CONJUNTO DE VARIÁVEIS INDEXADAS

A linguagem *BASIC* permite que seja criado na memória do computador um arquivo seqüencial, o qual é definido por um nome e por um número de ordem, ou seja um arquivo do tipo que descrevemos.

As regras para atribuição de um nome para este arquivo são exatamente as mesmas que vimos para designações de variáveis:

ARQUIVO NUMÉRICO - uma letra ou uma letra seguida de um algarismo.

ARQUIVO ALFANUMÉRICO - uma letra seguida de cifrão, ou uma letra seguida de um algarismo e de um cifrão.

Cada elemento desse arquivo é uma variável que possui o mesmo nome do arquivo e à qual está associado um número de ordem denominado índice, o qual é escrito entre parênteses, em seguida ao nome da variável.

Exemplos:

Nomes de arquivos numéricos *A, B1, C2*

Nomes de arquivos alfanuméricos *A$, B1$, C2$*

Nomes de variáveis do arquivo *A:*

A (0), A (1), ..., A (32), A (33), etc.

Nomes de variáveis do arquivo *B1$*

B1$ (0), B1$ (1), ..., B1$ (55), B1$ (56), etc.

Como cada variável possui um índice, é denominada de variável indexada, e como o arquivo é o conjunto dessas variáveis recebe o nome de conjunto de variáveis indexadas.

Obs.:

Em Inglês, os conjuntos de variáveis indexadas são chamadas de *"ARRAY"*, nome adotado pelos já iniciados em computação aqui no Brasil. Manteremos, entretanto, por razões didáticas, o uso do nome *CONJUNTO DE VARIÁVEIS INDEXADAS*.

Exemplos:

```
A (3)
```

Variável indexada *A* de número de ordem *3* do conjunto de variáveis indexadas *A*.

```
B5 (38)
```

Variável indexada *B5* de número de ordem *38* do conjunto de variáveis indexadas *B5*.

O exemplo abaixo visa tão somente mostrar como uma variável indexada é escrita num programa e como pode ser utilizada.

Exemplo 7.1

```
10 LET A(5)= 1234
20 PRINT A(5)
30 END
```

Rodando, você terá:

```
RUN

1234
```

O valor 1234 é associado a variável indexada *A(5)* e posteriormente é enviado para o vídeo.

GENERALIZANDO O CONJUNTO DE VARIÁVEIS

Aparentemente, a colocação de um *ÍNDICE* numa variável resulta tão somente no aumento do número de variáveis que

podem ser usadas em um programa e nada mais.

No entanto, em *BASIC*, o *ÍNDICE* da variável pode ser, ele próprio, uma variável inteira positiva. Isto permite que as variáveis indexadas de um mesmo conjunto possam ser genericamente representadas pela substituição do *ÍNDICE*, pelo nome de uma variável.

Deste modo, um outro formato possível para uma variável indexada é:

NOME DA VARIÁVEL (VARIÁVEL INTEIRA)

como exemplos:

A(I)

Variável indexada A, do conjunto de variáveis indexadas A, de número de ordem igual ao valor da variável inteira e positiva I. Se I possuir no instante da utilização dessa variável o valor 5, será utilizada a variável A (5).

B5(X)

Variável indexada B5, do conjunto de variáveis indexadas B5, de número de ordem igual ao valor da variável inteira e positiva X. Se X possuir no instante da utilização dessa variável o valor 32, será utilizada a variável B5 (32).

Exemplo 7.2

```
10 LET I= 5
20 LET A(5)= 12
30 PRINT A(I)
40 END
```

Rodando, temos:

```
RUN

12
```

A análise desse exemplo linha a linha é:

10 - A variável *I* assume o valor 5.

20 - O valor 12 é associado a variável indexada *A (5)*.

30 - Sendo 5 o valor do índice I, o computador enviará para o vídeo o valor associado a variável indexada *A (5)*, o qual foi declarado na instrução anterior.

40 - Instrução final do programa.

MANIPULANDO UM CONJUNTO DE DADOS

Se você realmente é um iniciante em programação, talvez não lhe seja muito fácil ver de imediato as possibilidades de uma variável indexada. Essa dificuldade surge do fato de que a criação de variáveis indexadas representa uma solução para problemas de manipulação de variáveis, problemas estes que você possivelmente não tenha ainda experimentado, mas que os criadores da linguagem *BASIC* conheciam e muito bem.

Suponhamos que você queira escrever um programa que receba 8 dados via teclado e depois os envie para o vídeo na mesma ordem que foram entrados, incluindo a soma dos mesmos. É um programa simples, mas servirá para mostrar como o uso de variáveis indexadas pode facilitar as coisas.

Inicialmente, vamos escrever o programa sem o uso de variáveis indexadas. Como queremos entrar 8 variáveis, vamos escolher como nomes das mesmas, as 8 primeiras letras do alfabeto. Um possível programa é mostrado abaixo:

Exemplo 7.3

```
100 INPUT A
110 INPUT B
120 INPUT C
130 INPUT D
140 INPUT E
150 INPUT F
160 INPUT G
170 INPUT H
180 PRINT A
190 PRINT B
200 PRINT C
210 PRINT D
220 PRINT E
230 PRINT F
240 PRINT G
250 PRINT H
260 LET S=A+B+C+D+E+F+G+H
270 PRINT "SOMA TOTAL";S
280 END
```

Rodando, temos:

```
RUN

?100
?233
?133
?215
?19
?1
?-33
?-45
100
233
133
215
19
1
-33
-45
SOMA TOTAL 623
```

O programa, como dissemos, é simples, mas se imaginarmos um número de dados maior, as coisas podem ficar bem complicadas. Um dos problemas que pode ocorrer é esgotar todos os possíveis nomes de variáveis. Como, felizmente, para nossa salvação existem as variáveis indexadas, vamos reescrever o programa utilizando-as.

Exemplo 7.4

```
100  LET S = 0
110  FOR X = 1 TO 8
120  INPUT A(X)
130  NEXT X
140  FOR Y = 1 TO 8
150  PRINT A(Y)
160  LET S = S + A(Y)
170  NEXT Y
180  PRINT "SOMA TOTAL ";S
190  END
```


Rodando, temos:

```
?100
?233
?133
?215
?19
?1
?-33
?-45
100
233
133
215
19
1
-33
-45
SOMA TOTAL 623
```

A solução adotada foi criar um primeiro *"LOOP"* e indexar a variável com o próprio *ÍNDICE* do *"LOOP"* possibilitando, assim, a entrada dos 8 dados. Um segundo *"LOOP"* é usado para a obtenção da saída e soma dos 8 dados.

Vamos analisar o exemplo 7.4 linha a linha:

100 - A variável S onde a soma será acumulada é zerada; isto de fato é desnecessário pois todas as variáveis inicialmente possuem valor zero. Entretanto é um bom procedimento pois num programa mais extenso, pode ocorrer que a mesma já tenha sido anteriormente utilizada.

110 - A variável a ser utilizada como índice é declarada, bem como seu valor inicial e final.

120 - Um conjunto de variáveis indexadas *A (X)* é declarado e o valor de cada variável do conjunto deve ser entrado via teclado. Na primeira vez quando o índice X for 1, o valor entrado 100 é associado a variável *A (1)*. Na vez seguinte o índice X será 2, e o valor entrado 233 será associado à variável *A (2)* e assim por diante, até o final do *"LOOP"*.

130 - O valor do *ÍNDICE X* é incrementado de 1 e o programa retorna a linha 120. Quando X for maior que 8, a linha 140 será executada.

140 - Uma outra variável Y é declarada como índice e seu valor inicial é definido como 1 e o final como 8.

150 - O conjunto de variáveis *A (Y)* é declarado e o valor de cada variável do conjunto deve ser enviado para o vídeo. Na primeira vez, quando o valor do índice Y for 1, o valor da variável *A (1)* que é 100, será enviado para o vídeo, na vez seguinte, o índice Y será 2 e o valor da variável *A (2)* que é 233 será enviado, e assim por diante, até o final do *"LOOP"*.

160 - A soma é acumulada na variável de nome S.

170 - O valor do índice Y é incrementado de 1 e a execução do programa retorna a linha 150. Quando o valor de Y for 9, o programa continua na linha seguinte.

180 - A frase *"SOMA TOTAL"* seguida do valor da variável S é enviada para o vídeo.

VARIAVEIS EM UMA FILA

Agora já deve estar bem mais claro a *"IMAGEM"* de *"FILA"* que fizemos para as variáveis indexadas. Durante a execução deste último programa, o computador usou um conjunto de variáveis indexadas de nome A e, que, após a entrada dos dados possuía os valores:

VARIÁVEL	*A(1)*	*A(2)*	*A(3)*	*A(4)*	*A(5)*	*A(6)*	*A(7)*	*A(8)*
VALOR	*100*	*233*	*133*	*215*	*19*	*1*	*-33*	*-45*

Um detalhe importante, e que não deve passar desapercebido a você, é que a variável é identificada pelo nome do conjunto e pelo valor do seu índice. Deste modo:

A(Y), A(T), A(V)

representam a mesma variável

A(3)

se Y, T e V forem 3 e a qual, no exemplo, possuía o valor 133.

Vamos escrever dois programas como ilustração do que acabamos de apresentar.

Exemplo 7.5

```
1000  FOR M = 1 TO 3
1100  INPUT X(M)
1200  NEXT M
1300  PRINT X(3),X(2),X(1)
1400  END
```

Rodando, teremos:

```
?100
?200
?300
300             200             100
```

Os valores 100, 200 e 300 entrados via teclado foram associados, respectivamente, às variáveis X(1), X(2) E X(3) e, posteriormente, foram enviados para o vídeo na ordem inversa de entrada, ou seja, X(3), X(2) e X(1) respectivamente 300, 200, 100.

Exemplo 7.6

```
100  LET A(1) = 111
200  LET A(2) = 222
300  LET A(3) = 333
400  LET A(4) = 444
500  FOR I = 1 TO 4
600  PRINT "A(";I;") = ";A(I)
700  NEXT I
800  END
```

Rodando, teremos:

```
RUN
A(1) = 111
A(2) = 222
A(3) = 333
A(4) = 444
```

As variáveis A(1), A(2) e A(3) e A(4) do conjunto de variáveis indexadas A, receberam respectivamente os valores 111 222, 333 e 444. Uma instrução de *PRINT* utilizando o índice I enviou, para o vídeo, uma listagem dos valores das variáveis do conjunto A.

DIMENSIONANDO O CONJUNTO DE VARIAVEIS INDEXADAS

Sempre que você utiliza um conjunto de variáveis indexadas, o computador reserva um espaço na memória, correspondente a 10 variáveis indexadas deste conjunto. Esta reserva de espaço na memória é automática e independe do número de variáveis que você realmente utilizar. Nada de anormal ocorrerá se você utilizar um número de variáveis indexadas igual ou menor que 10. Para utilizar um conjunto de variáveis indexadas com um número de variáveis superior a 10, o computador deve ser previamente preparado.

Preparar o computador para um determinado número de variáveis é chamado de *"DIMENSIONAR A VARIÁVEL"*.

A instrução em *BASIC* para dimensionar uma variável é:

DIM

e possui o formato:

número da linha *DIM* nome do conjunto (número de variáveis)

Exemplo de dimensionamentos

```
10 DIM A (183)
```

O computador prepara em sua memória um espaço para 183 variáveis do conjunto de variáveis indexadas A.

```
20 DIM B (1230), C (2560)
```

Você pode dimensionar vários conjuntos numa mesma instrução, desde que separe o dimensionamento de cada conjunto com vírgulas. Neste caso, o conjunto B, pode utilizar até 1230 variáveis, e o conjunto C, até 2560 variáveis.

Como o computador deve ser previamente preparado, as instruções de dimensionamento devem vir no início do programa, antes de se fazer uso do conjunto de variáveis.

O número máximo de variáveis permitido para cada conjunto de variáveis e, também, o número de variáveis indexadas como um todo depende do computador que você está usando, e também da memória de dados disponível do mesmo. O manual de instrução do seu computador normalmente contém essa informação, mas, caso esteja faltando, o melhor processo de saber, é o conhecido *"TENTATIVA E ERRO"*.

Obs.:

A primeira variável de um conjunto de variáveis indexadas é a variável de índice 0 e não a de índice 1 como normalmente esperaríamos. Assim, ao dimensionar um conjunto de 15 variáveis indexadas de nome H, a primeira variável será *H(0)* e a última *H(14)*.

Exemplo 7.7

```
10  DIM V(15)
20  PRINT "ANALISE DE RESULTADO DE VENDAS"
30  PRINT
40  PRINT "QUANTOS VENDEDORES? (MAXIMO 15)"
50  INPUT N
60  FOR I = 0 TO N - 1
70  PRINT
80  PRINT "QUAL O RESULTADO DO VENDEDOR ";I + 1;" ? "
90  INPUT V(I)
100  LET S = S + V(I)
110  NEXT I
120  PRINT
130  PRINT "GRAFICO COMPARATIVO DAS VENDAS"
140  PRINT
150  FOR I = 0 TO N - 1
160  LET P = (V(I) / S) * 100
170  PRINT I + 1;" ";
180  FOR M = 1 TO P
190  PRINT "*";
200  NEXT M
205  PRINT
210  NEXT I
220  END
```

Rodando, teremos:

```
ANALISE DE RESULTADO DE VENDAS

QUANTOS VENDEDORES? (MAXIMO 15)
?5

QUAL O RESULTADO DO VENDEDOR 1 ?
?85800

QUAL O RESULTADO DO VENDEDOR 2 ?
?260500

QUAL O RESULTADO DO VENDEDOR 3 ?
?462100

QUAL O RESULTADO DO VENDEDOR 4 ?
?575300

QUAL O RESULTADO DO VENDEDOR 5 ?
?346200

GRAFICO COMPARATIVO DAS VENDAS

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

Vamos deixar a seu cargo a análise desse programa. Uma observação, no entanto, deve ser feita: O *"LOOP"* iniciado na linha 180 possui como valor final, o valor da variável P que pode não ser inteira. Algumas versões de *BASIC* podem acusar erro nessa linha. Se esse for o seu caso, incluà no seu programa a linha abaixo:

```
175 LET P= INT (P)
```

Esta instrução *INT* (que será vista num próximo capítulo) retira a parte decimal de um número, deixando somente o valor inteiro. Assim, 11.25 passa a ser 11; 0.256, passa a ser 0, etc.

VARIÁVEIS ALFANUMÉRICAS INDEXADAS

Você pode criar também conjuntos de variáveis indexadas com variáveis alfanuméricas, do mesmo modo que para as numéricas. A diferença básica é que a maioria dos computadores não cria um dimensionamento automático para as variáveis alfanuméricas, e portanto, você deve fazer um dimensionamento prévio, independentemente da quantidade de variáveis a serem utilizadas.

Exemplos 7.8

```
10 DIM A$(10)
20 PRINT "ENTRE COM 5 NOMES"
30 FOR I=0 TO 4
40 INPUT A$(I)
50 NEXT I
60 PRINT
70 PRINT "SEQUENCIA INVERSA DOS NOMES ENTRADOS"
80 PRINT
90 FOR X=4 TO 0 STEP-1
100 PRINT A$(X)
110 NEXT X
120 END
```

Rodando, temos:

```
ENTRE COM 5 NOMES
?JOAO
?JOSE
?PEDRO
?MANOEL
?ANTONIO

SEQUENCIA INVERSA DOS NOMES ENTRADOS

ANTONIO
MANOEL
PEDRO
JOSE
JOAO
```

INTERLIGANDO CONJUNTOS DE VARIÁVEIS INDEXADAS

Muitas vezes ocorre a necessidade de se associar conjuntos de variáveis indexadas entre si. Por exemplo, na elaboração de um programa de controle de estoque, podemos ter os seguintes conjuntos de variáveis indexadas:

Descrição dos ítens	D$
Quantidade de cada ítem	Q
Custo de cada ítem	C

Supondo um estoque como o abaixo:

ITEM	*QUANTIDADE*	*CUSTO*
Parafuso	10.000	1.00
Porca	13.600	.75
Arruela	12.500	.45
Rebite	5.800	.65

a associação ordenada desses dados às variáveis dos conjuntos, resulta em:

Conjunto descrição do item:

```
D$  (0) = "PARAFUSO"
D$  (1) = "PORCA"
D$  (2) = "ARRUELA"
D$  (3) = "REBITE"
```

Conjunto quantidade de cada ítem:

```
Q  (0) = 10.000
Q  (1) = 13.600
Q  (2) = 12.500
Q  (3) =  5.800
```

Conjunto custo de cada ítem:

```
C (0) = 1.00
C (1) =  .75
C (2) =  .45
C (3) =  .65
```


É facil ver que os dados referentes a um determinado ítem possuem o mesmo valor de índice. Assim, para se estabelecer a ligação entre estes dados, basta se usar uma mesma variável para índice dos diversos conjuntos.

De acordo com esse raciocínio, os conjuntos ficam:

D$ (I), Q (I), C (I) e deste modo, para I = 0, temos

```
D$ (0) = "PARAFUSO"
Q (0) = 10.000
C (0) =  1.00
```

Para I = 1, temos:

```
D$ (1) = "PORCA"
Q  (1) = 13.600
C  (1) =   .75
```

e assim por diante.

Para se obter os dados referentes a um ítem do estoque, basta dar ao índice das variáveis, o número de ordem referente a esse item.

Exemplo 7.9

```
10  DIM D$(20),C(20),Q(20)
20  PRINT "CALCULO DO VALOR DO ESTOQUE"
30  PRINT
40  PRINT "QUANTOS ITENS? (MAXIMO 20)"
50  INPUT A
60  FOR I = 0 TO A - 1
70  PRINT "DENOMINACAO DO ITEM"
80  INPUT D$(I)
90  PRINT "QUANTIDADE"
100  INPUT Q(I)
110  PRINT "CUSTO"
120  INPUT C(I)
130  LET S = S + Q(I) * C(I)
140  NEXT I
150  PRINT "VALOR DE ESTOQUE"
170  PRINT
180  PRINT "ITEM","QUANT.","CUSTO"
190  FOR I = 0 TO A - 1
200  PRINT D$(I),Q(I),C(I)
210  PRINT
220  NEXT I
230  PRINT "VALOR TOTAL DO ESTOQUE ";S
240  END
```

Evidentemente este exemplo não tem a intensão de ser considerado como um programa para controle e/ou valorização de estoque, mas tão somente mostrar um ponto inicial para tal programa.

Rodando, temos:

```
CALCULO DO VALOR DO ESTOQUE

QUANTOS ITENS? (MAXIMO 20)
?4
DENOMINACAO DO ITEM
?PARAFUSO
QUANTIDADE
?10000
CUSTO
?1.00
DENOMINACAO DO ITEM
?PORCA
QUANTIDADE
?13600
CUSTO
?.75
DENOMINACAO DO ITEM
?ARRUELA
QUANTIDADE
?12500
CUSTO
?.45
DENOMINACAO DO ITEM
?REBITE
QUANTIDADE
?5800
CUSTO
?.65
VALOR DE ESTOQUE

ITEM            QUANT.          CUSTO
PARAFUSO        10000           1

PORCA           13600           .75

ARRUELA         12500           .45

REBITE          5800            .65

VALOR TOTAL DO ESTOQUE 29595
```


EXERCÍCIOS

EXERCÍCIO 7.1

O programa abaixo está errado. Por quê ?

```
10 DIM A$ (10)
20 FOR I = 1 TO 10
30 INPUT A$ (I)
40 NEXT I
50 END
```

EXERCÍCIO 7.2

Qual o resultado de se rodar o programa abaixo ?

```
10 DIM A (10)
20 FOR I = 10 TO 20
30 INPUT A (I)
40 PRINT B + A(I)
50 NEXT I
60 END
```

EXERCÍCIO 7.3

Escreva um programa que simule a retirada aleatória de cartas de um baralho, com a reposição da carta retirada ao baralho. A apresentação deve ser como o mostrado abaixo:

Exemplo:

```
AS DE OURO
NOVA CARTA ?
?S
7 DE PAUS
NOVA CARTA ?
?S
5 DE COPAS
NOVA CARTA ?
?N
```

CAPÍTULO 8

UMA TABELA COM VALORES

No capítulo 3, vimos dois modos de se atribuir valores a variáveis. Um imediato, utilizando a instrução *LET*, e outro, onde o valor da variável era fornecido via teclado do computador, com o uso da instrução *INPUT*.

Um terceiro modo, citado, mas não desenvolvido naquele capítulo, consiste na obtenção dos valores das variáveis de uma tabela escrita no próprio programa. Este modo deve ser utilizado quando o programa fizer uso de um grande número de dados fixos.

Duas instruções distintas são necessárias: uma para definição e criação da tabela de dados, e a outra para leitura dos dados da tabela.

CRIANDO A TABELA DE DADOS

Escrever a tabela de dados é bem simples. A tabela é identificada pela palavra *DATA* (DADO em Inglês), a qual é seguida pelos valores colocados na seqüência que serão utilizados no programa, separados por vírgulas.

Portanto, o formato dessa tabela é:

número da linha *DATA* valor, valor ... valor

Exemplos:

```
100 DATA 123, 5, -12, 16, 3.413
```

Tabela de dados numérica, composta de 5 valores.

```
200 DATA JANEIRO, 30, FEVEREIRO, 28, MARCO, 31
ABRIL 30, MAIO, 31
```

Tabela contendo dados alfanuméricos e numéricos. Note que os dados alfanuméricos não estão escritos entre aspas.

Não existe limite na quantidade de dados a serem escritos na tabela. Existe, no entanto, um limite de dígitos pa

ra cada linha do programa, o qual é usualmente de 256 caracteres no total, sendo computados os dígitos do número da linha, da instrução, dos dados, as vírgulas, os espaços, enfim, tudo. Se por acaso, a capacidade de linha for insuficiente para se escrever os dados a serem colocados na tabela, divida essa tabela em quantas forem necessárias. O computador, assim que terminar a leitura de dados da primeira instrução *DATA* do programa, verificará se existe outra instrução *DATA* e continuará a ler os dados desta nova tabela, e assim por diante.

Você pode colocar a tabela de dados em qualquer posição do programa:no início, no meio, ou no fim. A posição da tabela independe também da posição da instrução de leitura de dados, podendo ser colocada antes ou depois da mesma. Geralmente se usa colocar a tabela de dados no final do programa.

LENDO OS DADOS DE UMA TABELA

A leitura de dados de uma tabela, também é bem simples, no entanto, é necessário que você compreenda bem o *"MECANISMO"* dessa leitura para evitar *"BUGS"* em seu programa.

A instrução para leitura de dados de uma tabela é *READ* (LER em Inglês) e o seu formato é:

número da linha *READ* (nome da variável), ... (nome da variável)

Exemplos:

```
100 READ A
```

A cada execução desta instrução, um valor numérico é buscado na tabela e é associado à variável *A*.
Obs.: Se uma *"STRING"* for encontrada, o computador interrompe a execução do programa.

```
200 READ A$
```

A cada execução desta instrução, uma *"STRING"* é buscada na tabela e é associada à variável *A$*.
Obs.: Se um valor numérico for encontrado, o mesmo será considerado como uma *"STRING"* e o programa continuará sem interrupção.


```
300 READ A, A$
```

A cada execução desta instrução, um valor numérico e uma *"STRING"*, nesta ordem, são buscados da tabela e são associados às variáveis *A* e *A$*. As observações feitas acima, são válidas também neste caso.

A instrução *READ*, pode conter uma ou mais variáveis numéricas, alfanuméricas ou ambas, identificadas por seus nomes e separadas por vírgulas. Quando a instrução for executada, o dado ou conjunto de dados correspondente ao número de variáveis, constantes na instrução, será lido, ficando o computador automaticamente preparado para ler o dado ou conjunto de dados seguinte, da tabela.

Exemplo 8.1

```
10 FOR I= 1 TO 5
20 READ A
30 PRINT A
40 NEXT I
50 DATA 111, 222, 333, 444, 555
60 END
```

Rodando temos:

```
RUN

111
222
333
444
555
```

A leitura de dados da tabela é feita em ordem seqüencial, a partir do início da tabela.

Exemplo 8.2

```
100 FOR X= 1 TO 3
200 READ A$
300 PRINT A$
400 NEXT X
500 READ B$
600 PRINT B$
700 READ C$
800 PRINT C$
900 DATA  VERDE, AMARELO, AZUL, BRANCO, ROXO, CINZA
1000 END
```

Rodando, temos:

```
RUN

VERDE
AMARELO
AZUL
BRANCO
ROXO
```

A instrução *READ A$* da linha 200, foi executada 3 vezes, sendo responsável, portanto, pela leitura das 3 primeiras *"STRINGS"* da tabela, ou seja, *VERDE, AMARELO,* e *AZUL*. A instrução *READ B$* da linha 500 lê a *"STRING"* seguinte da tabela, ou seja *BRANCO* e a instrução *READ C$* lê a *"STRING" ROXO*. A última *"STRING"* da tabela, não foi lida pelo programa. Este fato não é detectado pelo computador, não sendo originada, portanto, mensagem de erro, e não cria qualquer tipo de problema para a execução do programa.

Exemplo 8.3

```
100  READ A$,A
200  READ B,B$
300  READ C
400  READ C$,D$,E$,F$
500  PRINT A$;A;B$;C$;D$;B;C$;E$;C;F$
600  DATA  "DIA ",22,1500,  " DE "
700  DATA  15," ABRIL "," DE "," AS "
800  DATA " HORAS "
900  END
```


Rodando, temos:

```
DIA 22 DE  ABRIL  DE 1500 ABRIL  AS 15 HORAS
```

Os dados podem ser colocados em várias tabelas (instrução DATA) mas devem estar na exata seqüência em que serão lidos. Quando você quiser incluir espaços antes e/ou depois de uma *"STRING"*, você tem de escrevê-las entre aspas, para que estes espaços fiquem claramente definidos.

Exemplo 8.4

```
100 FOR I= 1 TO 10
200 READ A$
300 PRINT A$
400 NEXT I
500 DATA  BANANA, MANGA, CAJU, UVA, LIMA
600 END
```

Rodando, temos:

```
BANANA
MANGA
CAJU
UVA
LIMA
OUT OF DATA ERROR  (OU QUALQUER OUTRA MENSAGEM
DE ERRO)
```

O computador inicia a leitura da tabela, lê a primeira *"STRING"* e a envia para o vídeo, e assim por diante, até a quinta e última *"STRING"* da tabela, sem qualquer problema. Ao tentar ler a sexta *"STRING"*, verifica que a tabela terminou. O programa é interrompido, e uma mensagem de erro é enviada para o vídeo.

Para que o programa opere corretamente, o número de dados da tabela deve ser, no mínimo, igual ao número dos dados a serem lidos. Para corrigir esse programa basta acrescentar as *"STRINGS"* faltantes ao programa. Para isso não é necessário reescrever a tabela; basta tão somente acrescentar uma nova tabela em continuação a anterior. Isto posto, acrescente a seguinte linha ao seu programa:

```
550 DATA PERA, MARACUJA, CAQUI, PESSEGO,
    DAMASCO
```

que o mesmo *"RODARÁ"* sem erros.

ESCREVENDO COMENTÁRIOS PARA O SEU USO PRÓPRIO

Vamos fazer uma pequena pausa em nosso estudo das instruções *DATA E READ* para apresentarmos uma instrução bastante útil, e que faz absolutamente nada no programa.

COMO É MESMO ? ! ! !

Essa instrução permitirá que você coloque no programa, linhas que não serão processadas pelo computador, podendo conter, desse modo, notas, comentários ou observações, explicando e/ou separando partes do programa, tornando-o mais fácil de entendimento.

O formato dessa instrução é:

número da linha *REM* complemento

onde o complemento é qualquer nota ou comentário que você achar por bem incluir em seu programa.

Exemplo:

```
100 REM ESTE E' UM COMENTÁRIO
```

Esta linha do programa não será processada pelo computador, mas estará presente no mesmo, quando este for listado.

Sempre que você escrever um programa longo, faça uso de comentários, para separar seções do mesmo e para esclarecer o que é que um determinado conjunto de instruções faz. A dificuldade de se redescobrir como um programa de sua própria autoria funciona, aumenta com o decorrer do tempo e diminui com o número de comentários contidos no mesmo.

No entanto, seja parcimonioso com o uso de comentários, pois os mesmos utilizam áreas da memória sem serem necessários a execução do programa.

Quando um programa for tão longo que a quantidade de memória utilizada se torna crítica, um *"TRUQUE"* é usar uma versão do programa com os comentários para futura análise e estudos e uma outra versão para execução pelo computador, na qual as linhas com os comentários foram eliminados.

Para exemplificarmos o uso de comentários em programas, vamos utilizá-los no exemplo seguinte deste capítulo.

VOLTANDO A LER O INÍCIO DA TABELA

Se no decorrer de um programa for necessária a leitura da mesma seqüência de dados da tabela, isto não requer que os mesmos sejam repetidos. Você pode através de uma instrução, reiniciar a leitura da tabela.

O formato dessa instrução é:

número da linha *RESTORE*

Exemplo 8.5

```
100 REM EXEMPLO DE USO DA INSTRUCAO RESTORE
110 PRINT "CALCULO DO LUCRO"
120 PRINT
130 REM ENTRADA DE DADOS DE CUSTO
140 FOR I= 1 TO 5
150 READ N$
160 PRINT "CUSTO DAS "; N$
170 INPUT C (I)
180 NEXT I
190 REM VOLTANDO AO INICIO DA TABELA
200 RESTORE
210 REM ENTRADA DE DADOS DE VENDAS
220 FOR I= 1 TO 5
230 READ N$
240 PRINT "VALOR DAS VENDAS DAS "; N$
250 INPUT V (I)
260 NEXT I
270 REM VOLTANDO AO INICIO DA TABELA
280 RESTORE
290 REM CALCULO DO LUCRO
300 FOR I= 1 TO 5
310 PRINT "LUCRO DAS "; N$; V (I)- C (I)
320 NEXT I
330 DATA "LARANJAS ","UVAS ","LIMAS ","AMEIXAS ",
    "BANANAS "
340 END
```

Rodando, temos:

```
RUN

CALCULO DO LUCRO

CUSTO DAS LARANJAS
?2350
CUSTO DAS UVAS
?8500
CUSTO DAS LIMAS
?4300
CUSTO DAS AMEIXAS
?7800
CUSTO DAS BANANAS
?1800
VALOR DAS VENDAS DAS LARANJAS
?4300
VALOR DAS VENDAS DAS UVAS
?12300
VALOR DAS VENDAS DAS LIMAS
?6800
VALOR DAS VENDAS DAS AMEIXAS
?12300
VALOR DAS VENDAS DAS BANANAS
? 3500
LUCRO DAS LARANJAS 1950
LUCRO DAS UVAS     3800
LUCRO DAS LIMAS    2500
LUCRO DAS AMEIXAS  4500
LUCRO DAS BANANAS  1700
```

A leitura da tabela de dados é reiniciada pelas linhas 200 e 280.

EVITANDO LER ALÉM DA TABELA

Às vezes, a quantidade de dados a serem lidos de uma tabela não pode ser previamente conhecida e o risco de erro de leitura de dados em excesso, deve mesmo assim ser evitado. O próximo exemplo mostra um modo de se obter esse resultado.

Exemplo 8.6

```
100  PRINT "ADIVINHANDO"
110  PRINT
120  READ P$,L,S$
130  IF P$ = "FIM" GOTO 290
140  PRINT "QUAL A PALAVRA QUE TEM ";L;" LETRAS "
145  PRINT " E SIGNIFICA ";S$;"?"
150  PRINT
160  PRINT "SE NAO SOUBER, TECLE RETURN E LHE DIREI"
170  INPUT R$
180  IF R$ = P$ GOTO 260
190  PRINT
200  PRINT "A PALAVRA E' ";P$
210  PRINT
220  PRINT "NOVA PALAVRA ? (S OU N)"
230  INPUT R$
240  IF R$ = "S" GOTO 110
250  GOTO 300
260  PRINT "PARABENS. VOCE ACERTOU"
270  GOTO 210
280  DATA   BAIXO  ,5,  VIL  ,AMBIGUO,7,IMPRECISO,CELIBATARIO,
     10,SOLTEIRO, FIM,0,FIM
290  PRINT "O MEU ESTOQUE DE PALAVRAS TERMINOU.TCHAU"
300  END
```

Rodando, temos:

```
RUN

ADIVINHANDO

QUAL A PALAVRA QUE TEM 5 LETRAS
 E SIGNIFICA VIL  ?

SE NAO SOUBER, TECLE RETURN E LHE DIREI
?VADIO

A PALAVRA E' BAIXO

NOVA PALAVRA ? (S OU N)
?S

QUAL A PALAVRA QUE TEM 7 LETRAS
 E SIGNIFICA IMPRECISO?

SE NAO SOUBER, TECLE RETURN E LHE DIREI
?AMBIGUO
PARABENS. VOCE ACERTOU

NOVA PALAVRA ? (S OU N)
?S

QUAL A PALAVRA QUE TEM 10 LETRAS
 E SIGNIFICA SOLTEIRO?
```

```
SE NAO SOUBER, TECLE RETURN E LHE DIREI
?

A PALAVRA E' CELIBATARIO

NOVA PALAVRA ? (S OU N)
?S

O MEU ESTOQUE DE PALAVRAS TERMINOU.TCHAU
```

A tabela de dados vai sendo lida até que a *"STRING" FIM* é encontrada quando o programa é desviado e é terminado sem erro.

EXERCÍCIOS

EXERCÍCIO 8.1

Utilizando as instruções *READ* e *DATA*, escreva um programa que apresente no vídeo os feriados oficiais do ano de 1984.

EXERCÍCIO 8.2

Utilizando as instruções *READ/DATA*, *RESTORE* escreva um programa que calcule o custo médio de cinco mercadorias em três supermercados diferentes.

EXERCÍCIO 8.3

Introduza comentários aos programas escritos para os exercícios 8.1 e 8.2.

CAPÍTULO 9

MANUSEANDO "STRINGS"

Nós aprendemos a enviar para o vídeo mensagens linha a linha, concatenadamente ou organizadas em colunas. No entanto, estas possibilidades não encerram a capacidade da linguagem *BASIC* em operar com *"STRINGS"*. Você poderá, a partir deste capítulo, saber qual o comprimento de uma *"STRING"*, acrescentar espaços entre *"STRINGS"*, dividir uma *"STRING"* em pedaços, colocar uma *"STRING"* ou espaços entre porções da *"STRING"* dividida, ou utilizar tão somente uma dessas porções no vídeo. Além disso, você poderá transformar variáveis numéricas em *"STRINGS"* e vice-versa, e também *"TABULAR"* o vídeo vertical e horizontalmente.

FUNÇÕES DE VARIÁVEIS ALFANUMÉRICAS OU DE "STRINGS"

Todo este manuseio de *"STRINGS"* é obtido através do uso de funções de *"STRING"*. Estas funções são semelhantes em formato às funções numéricas que vimos em capítulo anterior. Após a designação da função, segue-se um ou mais argumentos colocados entre parênteses.

NÚMERO DE CARACTERES

Como ponto inicial, desse capítulo, vamos conhecer a função em *BASIC* que nos fornece o número de caracteres presentes em uma *"STRING"* ou em uma variável alfanumérica.

O formato dessa função é:

LEN (argumento)

onde o argumento deve ser uma *"STRING"* ou uma variável alfanumérica.

Exemplos:

```
100 LET A= LEN ("ABACATE")
```

O número de caracteres da *"STRING" "ABACATE"* é associado à variável A que deste modo fica com o valor 7.

```
200 LET B= LEN ("")
```

A variável B foi associada ao número de caracteres de uma *"STRING"* vazia. O valor de B ficará, portanto, igual a zero.

```
300 PRINT LEN ("PULO DO GATO")
```

Esta linha de programa, quando executada, envia para o vídeo o número 12, correspondente ao número de caracteres na *"STRING" "PULO DO GATO"*.

```
400 LET A= LEN (A$)
```

O valor da variável A é igual ao número de caracteres constantes na *"STRING" A$*. Se *A$* contiver a *"STRING" "MINHA TERRA"*, o valor 11 será atribuído à variável A.

```
500 IF LEN (A$)>30 THEN PRINT "MUITO LONGA"
```

O comprimento da variável alfanumérica *A$* é comparado com o número 30. Sendo maior, a mensagem *MUITO LONGA* será enviada para o vídeo.

Vamos ao nosso primeiro exemplo aplicativo.

Exemplo 9.1

```
10 PRINT "NUMERO DE CARACTERES"
20 PRINT
30 PRINT "ENTRE COM O SEU NOME"
40 INPUT A$
50 PRINT
60 PRINT "O SEU NOME TEM "; LEN (A$); " CARACTERES "
70 END
```

Rodando, temos:

```
RUN

NUMERO DE CARACTERES

ENTRE COM O SEU NOME
?ASTROGILDO

O SEU NOME TEM 10 CARACTERES
```


ESPAÇOS EM MENSAGENS CONCATENADAS

Sempre que temos utilizado a instrução *PRINT* para escrevermos concatenadamente, temos incluído um ou mais espaços ao final de cada *"STRING"* para que as palavras não fiquem *"GRUDADAS"* quando enviadas para o vídeo. (Lembra-se do *MINHA TERRATEM PALMEIRAS*, do capítulo 2 ?)

A inclusão desses espaços, no entanto, pode ser feita através de uma função, tornando-se desnecessária a colocação dos mesmos na *"STRING"*.

O formato dessa função é:

SPC (argumento)

onde o argumento deve ser um valor ou uma variável numérica, representando o número de espaços a serem enviados.

Exemplo do formato:

```
SPC (12)
```

12 espaços serão enviados para o vídeo.

```
SPC (I)
```

serão enviados para o vídeo tantos espaços quanto for o valor de I.

Obs.:

O valor máximo de espaços que essa função pode enviar é 255. Além disso, deve ser um número positivo, e os valores fracionários serão arredondados automaticamente, dispensando-se a parte fracionária.

Vamos utilizar a instrução *PRINT* no modo direto, para avaliarmos a operação dessa função.

Exemplos:

```
PRINT "A"; SPC (1);"B"

A B
```

1 espaço foi incluído entre A e B

```
PRINT "A"; SPC (3); "B"

A   B
```

3 espaços foram incluídos entre A e B

```
PRINT "A"; SPC (21); "B"

A                     B
```

21 espaços foram incluídos entre A e B

```
PRINT SPC (3); "A"; SPC (5); "B"; SPC (6); "C"

   A     B      C
```

3 espaços foram colocados antes do A, 5 entre A e B e 6 entre B e C.

Vamos apresentar, agora, um exemplo no modo programado.

Exemplo 9.2

```
100 PRINT "QUANTOS ESPACOS (0 A 255)?"
200 INPUT X
300 PRINT "A"; SPC (X);"B"
400 PRINT
500 PRINT "TEMOS"; SPC (1); X; SPC (1);"ESPACOS
    ENTRE A E B"
600 END
```

Rodando, temos:

```
RUN

QUANTOS ESPACOS (0 A 255)?
?10
A          B

TEMOS 10 ESPACOS ENTRE A E B
```

O valor da variável X, entrado via teclado pela instrução *INPUT* da linha 200, é utilizado para gerar os espaços entre *"A"* e *"B"* na função *SPC (X)* da linha 300. Na linha 500 utilizamos a função *SPC (1)* para separarmos o valor da variável X das *"STRINGS"* à guisa de exemplo. É evidente que a inclusão,

neste caso, de espaços de separação nas *"STRINGS"* é bem mais simples.

Exemplo 9.3

```
100 FOR I= 0 TO 10
200 PRINT SPC (I); I
300 NEXT I
400 END
```

Rodando, temos:

```
RUN

0
 1
  2
   3
    4
     5
      6
       7
        8
         9
          10
```

A cada *"LOOP"*, o índice é incrementado, aumentando o número de espaços que antecedem o seu valor no vídeo.

ESCREVENDO EM COLUNAS A SEU GOSTO

A utilização conjunta dessas duas funções permite que se possa escrever em colunas no vídeo, num formato diferente do formato fixado pelo computador. (veja *ESCREVENDO EM COLUNAS*, capítulo 2).

Vamos escrever em nosso vídeo em duas colunas. A primeira, iniciando-se na posição zero e, a segunda, na posição 12. Para isso, a instrução *PRINT* deverá preencher com espaços as posições à direita da primeira palavra até a segunda palavra. A quantidade de espaços a ser enviada depende, além do número de caracteres da coluna, do comprimento da primeira palavra. No presente caso, temos uma coluna de 11 caracteres. Assim, se a primeira palavra for composta, digamos, de 3 caracteres, 8 espaços deverão ser enviados para *"COMPLETAREM"* a pri

meira coluna. Como você pode ver, o número de espaços resulta diretamente da diferença entre a largura da coluna e o comprimento da palavra. A expressão em *BASIC* correspondente a esse cálculo é:

```
SPC (L - LEN (A$))
```

onde *L* é a largura da coluna e *A$* é a variável representativa da primeira palavra e, portanto, *LEN (A$)* é o valor de seu comprimento.

Exemplo 9.4

```
100 PRINT SPC (3); "ANTONIMOS"
110 PRINT
120 FOR I= 1 TO 5
130 READ A$, B$
140 PRINT A$; SPC (11 - LEN (A$)); B$
150 NEXT I
160 DATA BONITO, FEIO, CLARO, ESCURO, ALTO, BAIXO,
    MAGRO, GORDO, RICO, POBRE
170 END
```

Rodando, teremos:

```
RUN

   ANTONIMOS

BONITO      FEIO
CLARO       ESCURO
ALTO        BAIXO
MAGRO       GORDO
RICO        POBRE
```

Analisando o programa, temos:

100 - Envia para a tela 3 espaços e após o título *ANTONIMOS*.

110 - Envia uma linha em branco.

120 - Inicia um *"LOOP"*.

130 - Lê duas variáveis *A$* e *B$* de uma tabela.

140 - Envia para o vídeo o conteúdo da variável *A$* e após, espaços correspondentes à largura da primeira coluna, menos o comprimento da *"STRING" A$*, e em seguida o conteúdo da variável *B$*.

150 - Instrução final do *"LOOP"*.

160 - Tabela de dados do programa a ser lida pela instrução *READ*.

170 - Instrução final do programa.

RETIRANDO PARTES DE UMA "STRING"

Existem em *BASIC* funções que permitem retirar uma parte de uma *"STRING"* ou de uma variável alfanumérica, e criar com essa parte retirada, uma nova *"STRING"* ou variável alfanumérica.

A parte retirada pode ser a parte esquerda, direita ou central, originando assim, tres funções: *LEFT$*, *RIGHT$* e *MID$*.

RETIRANDO UMA PARTE DA ESQUERDA

A função que retira uma parte da esquerda de uma *"STRING"* ou variável alfanumérica tem como formato:

LEFT$ (argumento 1, argumento 2)

onde:

Argumento 1 é a *"STRING"* ou variável alfanumérica que será selecionada.

Argumento 2 é um valor ou variável numérica, representando o número de carácteres que serão retirados para formar a nova *"STRING"* ou variável alfanumérica.

Exemplos:

```
LET B$= LEFT$ ("EXTRAORDINARIO", 5)
```

serão retirados os cinco primeiros caracteres da *"STRING" "EXTRAORDINÁRIO"* para formar a nova *"STRING" B$*, resultando em *"EXTRA"*.

```
LET H$= LEFT$ (A$, 6)
```

os seis primeiros caracteres à esquerda da *"STRING"* representados pela variável *A$* serão associados à variável *H$*. Se *A$* contiver uma *"STRING"* menor que seus caracteres, toda ela será associada a variável *H$*. Supondo *A$ = "PAPEL"*, teremos *H$ = "PAPEL"*.

Lembra-se do *ADIVINHANDO* do capítulo passado ? O exemplo a seguir é uma versão do mesmo, com algumas modificações.

Exemplo 9.5

```
100 PRINT "ADIVINHANDO UMA PALAVRA"
110 PRINT
120 READ P$,E$
130 IF P$= "FIM" THEN GOTO 370
140 PRINT E$
150 PRINT
160 PRINT "TEM "; LEN (P$); " LETRAS "
170 PRINT
180 PRINT "LHE DOU "; LEN (P$)-3; " TENTATIVAS"
190 FOR I= 1 TO LEN (P$)-3
200 PRINT
210 PRINT "COMECA COM "; LEFT$ (P$,I)
220 PRINT
230 PRINT "QUAL E'?"
240 INPUT R$
250 IF R$= P$ THEN GOTO 310
260 PRINT "ESTA ERRADO"
270 NEXT I
280 PRINT
290 PRINT "VOCE PERDEU.  A PALAVRA ERA "; P$
300 GOTO 330
310 PRINT "VOCE ACERTOU.  PARABENS"
320 PRINT
330 PRINT "NOVA PALAVRA? (S OU N)"
340 INPUT R$
350 IF R$= "S" THEN GOTO 140
360 DATA BURITI, PALMEIRA DE FRUTO AMARELO,
    CLANDESTINO, ILEGAL, DEFENDER, PROTEGER,
    ORNAMENTAR, DECORAR, LEOPARDO, PANTERA, FIM, FIM
370 END
```


Rodando, temos:

```
RUN

ADIVINHANDO UMA PALAVRA

PALMEIRA DE FRUTO AMARELO

TEM 6 LETRAS

LHE DOU 3 TENTATIVAS

COMECA COM B

QUAL E'?
?BANANA
ESTA ERRADO

COMECA COM BU

QUAL E'?
?BUBUIA
ESTA ERRADO

COMECA COM BUR

QUAL E'?
?BURITI
VOCE ACERTOU.  PARABENS

NOVA PALAVRA? (S OU N)
?N
```

AGORA, A VEZ DA DIREITA

A função que retira a parte da direita de uma *"STRING"*, ou de uma variável alfanumérica é de todo semelhante a função *LEFT$* com a única diferença de atuar no lado direito.

Exemplos:

```
LET B$= RIGHT$ ("EXTRAORDINARIO", 9)
```

A variável *B$* será igualada à *"STRING" "ORDINÁRIO"*.

```
LET H$= RIGHT$ (A$, 6)
```

Os seis primeiros caracteres contados a partir da direita da *"STRING"* representada pela variável *A$*, serão associados à variável *H$*. Se *A$* contiver uma *"STRING"* menor que seis

caracteres, toda a *"STRING"*, será associada à variável *H$*. Se *A$ = "PAPEL"*, teremos *H$ = "PAPEL"*.

Exemplo 9.6

```
100 PRINT "ENTRE COM UMA PALAVRA"
110 INPUT P$
120 PRINT
130 FOR I= 1 TO LEN (P$)
140 PRINT SPC (LEN (P$)- I); RIGHT$ (P$,I)
150 NEXT I
160 END
```

Rodando, temos:

```
ENTRE COM UMA PALAVRA
?EXTRATERRESTRE

             E
            RE
           TRE
          STRE
         ESTRE
        RESTRE
       RRESTRE
      ERRESTRE
     TERRESTRE
    ATERRESTRE
   RATERRESTRE
  TRATERRESTRE
 XTRATERRESTRE
EXTRATERRESTRE
```

A cada *"LOOP"*, a variável I é incrementada, diminuindo o número de espaços criados pela função *SPC (LEN (P$) - I)* e aumentando o número de caracteres gerado pela função *RIGHT$ (P$,I)*.

RETIRANDO DO MEIO

Quando se quer retirar uma parte qualquer de uma *"STRING"* ou de uma variável alfanumérica para formar uma nova *"STRING"* ou variável alfanumérica, a função *BASIC* é:

MID$ (argumento 1, argumento 2, argumento 3)

onde:

Argumento 1 é a *"STRING"* ou variável alfanumérica que será secionada.

Argumento 2 é um valor ou variável numérica que indica a partir de qual caractere, a nova *"STRING"* ou variável alfanumérica será formada. Esse valor deve estar no intervalo de 1 a 255.

Argumento 3 é o valor ou variável numérica representando o número de caracteres que serão retirados.

Exemplo do formato:

```
LET B$= MID$ ("EXTRAORDINARIO", 3,5)
```

A partir do terceiro, serão retirados cinco caracteres, resultando *B$ = "TRAOR"*.

```
LET H$= LEFT$ (A$, 8,2)
```

Dois caracteres, o oitavo e nono da variável *A$* serão associados à variável *H$*. Se esses caracteres não existirem a variável *H$* ficará *"VAZIA"*, ou seja, sem caracteres.

Vamos utilizar, novamente, a instrução *PRINT* no modo direto, para avaliarmos a operação dessa função.

Exemplos:

```
PRINT MID$ ("LABORATORIO", 1,5)
```

```
LABOR
```

Cinco letras a partir da primeira, inclusive, foram enviadas para o vídeo.

```
PRINT MID$ ("LABORATORIO", 4,8)
```

```
ORATORIO
```

Oito letras a partir da quarta letra, inclusive, foram enviadas para o vídeo.

```
PRINT MID$ ("LABORATORIO, 6,4)
```

```
ATOR
```

Quatro letras a partir da sexta letra, inclusive, foram enviadas para o vídeo.

```
PRINT MID$ ("LABORATORIO", 9,3)

RIO
```

Três letras a partir da nona, inclusive, foram enviadas para o vídeo.

Exemplo 9.7

```
100 PRINT "ENTRE COM SEU NOME"
110 PRINT
120 INPUT N$
130 FOR I= 1 TO LEN (N$)
140 PRINT MID$ (N$, I, 1)
150 NEXT I
160 END
```

Rodando, teremos:

```
ENTRE COM SEU NOME

?JOSE MARIA
J
O
S
E

M
A
R
I
A
```

Na linha 130, o valor máximo do índice da instrução *FOR TO* é definido como o número de caracteres da variável *N$*. A linha 140 retira a cada *"LOOP"* um caractere desta variável e a envia para o vídeo, resultando no nome escrito na vertical.

TRANSFORMAÇÕES DE VARIÁVEIS

Todo esse manuseio de *"STRINGS"* e variáveis alfanuméricas que vimos, não se aplica a valores ou a variáveis numéricas.

Para que isso não constituisse em uma séria limitação, foram criadas em *BASIC*, funções que transformam uma variável numérica em alfanumérica e vice-versa.

Isto vem permitir que programas, principalmente na área econômica, que utilizam como dados valores e *"STRINGS"*, possam ser escritos de modo a que esses dados sejam entrados como *"STRINGS"*, possibilitando, assim, o manuseio dos mesmos para apresentação no vídeo a seu gosto.

As *"STRINGS"* correspondentes a dados numéricos são transformadas, em valores, os cálculos são efetuados e os resultados obtidos são transformados em *"STRINGS"*, para serem enviados para o vídeo.

NUMÉRICAS EM ALFANUMÉRICAS

A função que transforma um valor numa *"STRING"* ou uma variável numérica em alfanumérica tem o formato:

STR$ (argumento)

onde o argumento é o valor ou a variável a ser transformada.

Exemplos:

```
LET B$= STR$ (1.2345)
```

A *"STRING" "1.2345"* é associada à variável alfanumérica *B$*.

```
LET A$= STR$ (A)
```

O valor da variável A é transformado em uma *"STRING"* a qual é associada à variável alfanumérica *A$*. Se A for igual a 1034, então *A$* será *"1034"*.

Exemplo 9.8

```
100 FOR I= 1 TO 10
200 LET I$= STR$ (I)
300 PRINT I; SPC (3); I$
400 NEXT I
500 END
```

Rodando, temos:

```
RUN

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

ALFANUMÉRICAS EM NUMÉRICAS

A função que transforma uma *"STRING"* em um valor ou uma variável alfanumérica em numérica tem o formato:

VAL (argumento)

onde o argumento é a *"STRING"* ou a variável alfanumérica a ser transformada.

Mais uma vez, vamos avaliar a operação desta função utilizando a instrução *PRINT* no modo direto.

Exemplos:

```
PRINT VAL ("20.5")  + 100
120.5
```

A *"STRING" "20.5"* é transformada em valor o qual é somado com 100 e o resultado 120.5 é enviado para o vídeo.

```
PRINT VAL ("22 DE ABRIL DE 1500")

22
```

Os caracteres numéricos constantes na *"STRING"* são transformados em valor até que um caractere não numérico é encontrado, encerrando-se aí a transformação.

```
PRINT VAL ("MAIO DE 1900")

0
```


Quando a *"STRING"* não se inicia por um caractere numérico ou todos os caracteres não são numéricos, o valor resultante é zero.

Exemplo 9.9

```
100 DIM P$ (100)
110 PRINT "SOMA"
120 PRINT
130 PRINT "QUANTAS PARCELAS?"
140 PRINT
150 INPUT P
160 PRINT
170 FOR I= 1 TO P
180 PRINT "VALOR DA PARCELA?"
190 INPUT P$ (I)
200 NEXT I
210 PRINT
220 FOR I= 1 TO P
230 LET S= S+ VAL (P$(I))
240 PRINT SPC (15- LEN (P$(I)));  P$(I)
250 NEXT I
260 PRINT
270 PRINT"SOMA";  SPC (11- LEN (STR$(S$)));S
280 END
```

Rodando, temos:

```
RUN

SOMA

QUANTAS PARCELAS?

?5

VALOR DA PARCELA?
?101
VALOR DA PARCELA?
?33
VALOR DA PARCELA?
?55
VALOR DA PARCELA?
?1222
VALOR DA PARCELA?
?555233

           101
            33
            55
          1222
        555233

SOMA     556644
```

TABULANDO...

As máquinas de escrever possuem um dispositivo muito útil, o qual permite o deslocamento da posição de impressão para uma posição predeterminada, facilitando, deste modo, o trabalho de datilografar em colunas, ou do alinhamento de parágrafos, sendo denominado de Tabulador.

A linguagem *BASIC* possui duas funções relacionadas a instrução *PRINT* que permitem uma tabulação horizontal semelhante a da máquina de escrever e uma tabulação vertical não existente naquela.

TABULAÇÃO HORIZONTAL

A tabulação de uma linha, em *BASIC*, é obtida através da função de formato:

TAB (argumento)

onde o argumento deve ser um valor ou uma variável numérica,que define a posição de tabulação na linha do vídeo. Este valor deve estar no intervalo 1 a 255.

Exemplos, com *PRINT* no modo direto:

```
PRINT TAB (10);"* ESTE ASTERISCO ESTA NA COLUNA 10"
         * ESTE ASTERISCO ESTA NA COLUNA 10
```

A posição inicial da mensagem no vídeo foi definida como 10.

```
PRINT TAB (10); "BONITO"; TAB (15); "FEIO"

          BONITOFEIO
```

A posição da palavra *BONITO* inicia na posição 10, indo até a posição 15. Como a palavra *FEIO* deveria ser escrita a partir de uma posição já ocupada, a função *TAB (15)* não foi considerada.

```
PRINT TAB (20); "ALTO"; TAB (10); "MAGRO"

                    ALTOMAGRO
```


A função TAB não tabula para trás.

No modo programado, temos:

Exemplo 9.10

```
10 FOR I= 1 TO 10
20 PRINT TAB (I); I
30 NEXT I
40 END
```

Rodando, temos:

```
RUN
1
 2
  3
   4
    5
     6
      7
       8
        9
         10
```

Exemplo 9.11

```
100  PRINT  TAB( 4);"ANTONIMOS"
110  PRINT
120  FOR I = 1 TO 5
130  READ A$,B$
140  PRINT A$; TAB( 12);B$
150  NEXT I
160  DATA  BONITO,FEIO,CLARO,ESCURO,ALTO,BAIXO,MAGRO,GORDO
     ,RICO,POBRE
170  END
```

Rodando, temos:

```
   ANTONIMOS

BONITO      FEIO
CLARO       ESCURO
ALTO        BAIXO
MAGRO       GORDO
RICO        POBRE
```

Temos aqui versões do exemplo 9.3 e do exemplo 9.4. Com o tempo, você verá que um mesmo resultado pode ser obtido com as mais variadas versões de programas. Cada programador tem seu modo particular de *"TRADUZIR"* em linguagem *BASIC*, a solução de um programa. São traços característicos, o modo de organizar o programa, a seqüência numérica das linhas, um conjunto preferido de instruções, os nomes das variáveis, dos índices, etc. Enfim, cada programa traz dentro de si o *"ESTILO"* próprio do programador.

TABULAÇÃO VERTICAL

Você pode escolher, em *BASIC*, para qual linha do vídeo a instrução *PRINT* enviará a mensagem. Isto é obtido através da função *VTAB* de formato:

número da linha *VTAB* (argumento)

onde o argumento deve ser um valor ou uma variável numérica, definindo a linha a ser utilizada pela instrução *PRINT*. A contagem das linhas inicia-se na parte superior do vídeo, com o valor 1 e o valor final depende do número de linhas de texto no vídeo do seu computador.

Na descrição do formato foi incluído o *"NÚMERO DA LINHA"* pois esta função deve ser escrita numa linha precedente a instrução *PRINT* e não como parte integrante dela.

Vamos apresentar dois exemplos de aplicação desta função, sem, no entanto, mostrarmos o resultado na tela do vídeo, por motivos que ficarão claros quando você rodar o programa.

Exemplo 9.12

```
100 REM LIMPEZA DA TELA
200 FOR I= 1 TO 30
300 PRINT
400 NEXT I
500 REM LEVANDO O CURSOR PARA LINHA 1
600 VTAB (1)
700 PRINT "ESCREVENDO NA LINHA 15"
800 PRINT "ENTRE COM UMA FRASE QUALQUER"
900 INPUT A$
1000 REM LEVANDO O CURSOR PARA A LINHA 15
1100 VTAB (15)
1200 PRINT A$
1300 REM LEVAR O CURSOR PARA A LINHA 4
1400 VTAB (4)
1500 REM ROTINA DE CONTINUAR OU NAO NO PROGRAMA
1600 PRINT "NOVA FRASE? (S OU N)"
1700 INPUT R$
1800 IF R$= "S" GOTO 200
1900 END
```

Exemplo 9.14

```
100 REM LIMPEZA DA TELA
200 FOR I= 1 TO 30
300 PRINT
400 NEXT I
500 REM MENSAGENS INICIAIS A PARTIR DA LINHA 1
600 VTAB (1)
700 PRINT "RODANDO UMA FRASE"
800 PRINT
900 PRINT "ENTRE COM UMA FRASE QUALQUER"
1000 REM ENTRANDO A FRASE NA LINHA 15
1100 VTAB (15)
1200 INPUT A$
1300 REM RODAR A FRASE DA LINHA 15
1400 LET A$= MID$ (A$, 2, LEN (A$)- 1)+ LEFT$ (A$-1)
1500 PRINT A$
1600 REM ESTABELECER A VELOCIDADE DE ROTACAO
1700 FOR X= 0 TO 100
1800 NEXT X
1900 REM RODAR NOVAMENTE
2000 GOTO 1400
2100 END
```

(Antes de rodar este programa, procure descobrir qual é o resultado do mesmo).

Para interromper a execução deste programa, você deve entrar com o *CONTROL C*, pois o *"LOOP"* da linha 2000 é um

"LOOP" infinito.

Mude a linha 1700 do programa para:

```
1700 FOR X=0 TO 1000
```

e veja o resultado.

EXERCÍCIOS

EXERCÍCIO 9.1

Escreva um programa que converta um número entrado na notação Americana, a qual utiliza ponto decimal para a nossa notação onde a vírgula é utilizada.
Por exemplo:

12345.367 convertido para 12345,367

EXERCÍCIO 9.2

Escreva um programa que escreva de trás para frente uma palavra ou frase entrada via teclado.

EXERCÍCIO 9.3

Escreva um programa que transforme um código de letras em valor numérico entrado, utilizando a palavra chave PERNAMBUCO.
Por exemplo:

O número 136 seria transformado em PRM.

EXERCÍCIO 9.4

Escreva um programa que codifique uma palavra ou frase, utilizando o código:

ZENIT
POLAR

onde cada letra é trocada pela sua oposta, ou seja, o Z pelo P e vice-versa, o E pelo O e vice-versa, e assim por diante.
Exemplo:

PALMEIRAS ficaria ZINMOATIS.

CAPÍTULO 10

"BUGS" E MATEMÁTICA

Este capítulo completa a apresentação das instruções e funções fundamentais da linguagem *BASIC*. Inicialmente serão vistas instruções que facilitam a análise de programas a procura dos *"BUGS"* que fatalmente aparecem durante o seu desenvolvimento. A parte final do capítulo será dedicada ao estudo de uma instrução que permitirá a definição de uma função matemática qualquer.

MODUS OPERANDI

Nossa intenção não é ensinar latim, mas sim relembrar e complementar o que foi dito até agora sobre o *"MODO DE OPERAR"* de um computador pessoal típico.

Ao ser ligado, o computador limpa as áreas de memória do programa, variáveis, vídeo e envia para o seu vídeo uma mensagem qualquer indicando que está pronto para ser usado.

O computador nesta situação opera de dois modos:

COMANDO - Este modo de operação corresponde a aceitar palavras de comando enviadas via teclado e executar a ação ou ações correspondentes, imediatamente. Como exemplo dessas palavras de comando, temos: *RUN, NEW, PRINT, LET, GOTO,* etc.

EDIÇÃO - Este modo de operação corresponde a toda a tarefa, envolvendo a entrada, listagem, correção, inclusão e cancelamento de linhas, de programas.

Nós vamos adotar para essa condição operacional o nome de *MODO DE COMANDO E EDIÇÃO*.

Uma vez carregado com um programa em *BASIC*, o computador recebendo um comando adequado passa a segunda condição operacional a qual é denominada de *MODO PROGRAMADO*. Este modo consiste em executar as instruções constituintes do programa até que essas terminem ou até que essa execução seja interrompida por uma instrução no próprio programa, ou pela ocor

rência de um erro ou pelo envio de um comando externo. Ocorrendo qualquer uma dessas possibilidades, o computador volta automaticamente ao modo de Comando e Edição. Essa mudança de modo de operação não modifica o conteúdo da memória do computador. O programa lá arquivado continua exatamente como antes, como também os últimos valores de cada variável do programa.

ESSES "BUGS"!!

A análise de programas para a descoberta de erros é uma tarefa rotineira no mundo da computação. Muitas vezes, o programa "roda" perfeitamente, apresenta os resultados na forma esperada, mas ... errados. Um dos processos de testar a exatidão de um programa é entrar com um conjunto de dados, rodar, e comparar os resultados com os obtidos manualmente, ou através de um outro programa que se saiba correto.

A descoberta de um erro, no entanto, nem sempre permite a identificação da sua origem dentro do programa.

"Esse resultado será devido à inclusão de uma parcela inadequada ? Ou será devido a um erro de cálculo do percentual da alíquota ? _ Vamos ver se modificando ..."

Para facilitar a solução de problemas deste tipo, foram criados em *BASIC*, uma instrução que interrompe a execução do programa num ponto previamente escolhido e um comando que permite a continuação desta execução.

PROGRAMANDO A INTERRUPÇÃO

A instrução em *BASIC* que permite a interrupção da execução de um programa, numa determinada linha, tem como formato:

número da linha *STOP*.

Esta instrução quando executada pelo programa causará a interrupção do mesmo, e enviará uma mensagem para o vídeo, indicando o número da linha na qual o programa parou, evidentemente, o da própria instrução.

Exemplo 10.1

```
100 LET X= 10* RND (1)
200 IF X > 5 GOTO 400
300 STOP
400 STOP
500 END
```

Rodando, temos:

```
RUN

BREAK IN 400
```

Mesmo que exista no programa mais que uma instrução *"STOP"*, a identificação da linha desta instrução permite uma análise do comportamento do programa. A interrupção do programa na linha 400, leva a concluir que o valor atribuído a X pela função *10 * RND (1)* é maior que 5.

Esta interrupção de programa leva o computador ao modo de Comando e Edição, sendo mantidos na memória os últimos valores de cada variável do programa, os quais podem ser verificados através da instrução *PRINT* no modo direto.

Vamos supor que um programador tenha escrito o programa abaixo com intuito de calcular o seguinte problema:

"quanto é duas vezes a terça parte de um número mais o seu dobro ?"

Exemplo 10.2

```
100 PRINT "SOLUCAO DO PROBLEMA"
110 PRINT "ENTRE COM O VALOR"
120 REM CALCULO DA TERCA PARTE
130 LET Y= X/3
140 INPUT X
150 REM CALCULO DA EXPRESSAO
160 LET Y= Y * 2
170 LET X= X * 2
180 PRINT "RESULTADO "; X + Y
190 END
```

Rodando, temos:

```
SOLUCAO DO PROBLEMA
ENTRE COM O VALOR
?3
RESULTADO 6
```

O programa rodou e aparentemente o resultado apresentado está correto. No entanto, fazendo-se o cálculo mentalmente, verifica-se que o resultado correto é 8.

Vamos colocar uma instrução *STOP* antes do cálculo da expressão, para verificarmos os valores das variáveis naquele ponto.

Exemplo 10.3

```
100 PRINT "SOLUCAO DO PROBLEMA"
110 PRINT "ENTRE COM O VALOR"
120 REM CALCULO DA TERCA PARTE
130 LET Y= X/3
140 INPUT X
145 STOP
150 REM CALCULO DA EXPRESSAO
160 LET Y= Y * 2
170 LET X= X * 2
180 PRINT "RESULTADO "; X + Y
190 END
```

Rodando, temos:

```
RUN

SOLUCAO DO PROBLEMA
ENTRE COM O VALOR
?3
BREAK IN 145
```

O programa foi interrompido na linha 145 e o computador está no modo de Comando e Edição. Vamos verificar qual o valor das variáveis que foram utilizadas até esse ponto com o uso da instrução *PRINT* no modo direto.

```
PRINT X
3
```

Está correto. Este foi o valor que entramos para X.

```
PRINT Y
0
```

Está errado. Deveria ser 1. Portanto, o erro está por aqui.

Analisando as linhas 130 e 140, verificamos que existe uma inversão de posição das mesmas, pois o valor de Y é calculado antes da entrada do valor de X.

Corrigindo a posição destas linhas e suprimindo a linha 145, temos:

Exemplos 10.4

```
100 PRINT "SOLUCAO DO PROBLEMA"
110 PRINT "ENTRE COM O VALOR"
120 REM CALCULO DA TERCA PARTE
130 INPUT X
140 LET Y= X/3
150 REM CALCULO DA EXPRESSAO
160 LET Y= Y* 2
170 LET X= X * 2
180 PRINT "RESULTADO "; X + Y
190 END
```

Rodando, temos:

```
RUN

SOLUCAO DO PROBLEMA
ENTRE COM O VALOR
?3
RESULTADO 8
```

RETOMANDO A EXECUÇÃO

O comando em *BASIC* que permite a continuação da execução de um programa é:

CONT

Este comando retorna a execução do programa a partir da instrução imediatamente após o *STOP* e preserva os valores de todas as variáveis utilizadas pelo programa.

Exemplo 10.5

```
100 LET A= 15
110 LET B= 3
120 PRINT A * B
130 STOP
140 PRINT A - B
150 STOP
160 PRINT 2 * A + B
170 STOP
180 PRINT 3 * A + 5 * B
190 END
```

Rodando, temos:

```
RUN

45
BREAK IN 130
CONT
12
BREAK IN 150
CONT
33
BREAK IN 170
CONT
60
```

Após cada interrupção do programa, o comando *CONT*, retomou a execução do mesmo, sendo preservados os valores das variáveis, como pode ser verificado pela análise dos resultados obtidos.

ENTRANDO COM NOVOS VALORES

Tendo sido interrompida a execução do programa, os valores das variáveis não só podem ser verificados mas como também podem ser modificados.

Rodando o exemplo 10.4 novamente, temos:

```
RUN

45
BREAK IN 130
```


As variáveis A e B possuem respectivamente, neste instante, os valores 5 e 3. Vamos mudá-los, utilizando a instrução *LET*, no modo direto.

```
LET A= 1000
LET B= 200
```

Continuando a execução do programa, temos:

```
CONT

800
BREAK IN 150
```

O computador continuou a execução do programa, utilizando os novos valores das variáveis A e B, o que nos demonstra que a modificação dos valores de variáveis durante a interrupção de um programa é permitida.

"O uso da instrução *LET* no modo direto não modifica também o programa ?"

A melhor resposta a essa pergunta é a listagem do mesmo e portanto:

```
LIST

100 LET A= 15
110 LET B= 3
120 PRINT A * B
130 STOP
140 PRINT A - B
150 STOP
160 PRINT 2 * A + B
170 STOP
180 PRINT 3 * A + 5 * B
190 END
```

O programa continua exatamente o mesmo de antes de entrarmos a instrução *LET* no modo direto. As variáveis foram mudadas tão somente na memória do computador.

Continuando a execução, temos:

```
CONT

2200
BREAK IN 170
```

O fato de termos listado o programa, não impediu a continuação da execução do mesmo. Suponhamos, agora, que verificando a listagem do programa, tenhamos detectado um erro na linha 180. A expressão deveria ser 5A + 3B e não 3A + 5B. Vamos, portanto corrigir, esta linha do programa.

```
180 PRINT 5 * A + 3 * B
```

Retomando a execução do programa, temos:

CONT

CAN'T CONTINUE ERROR (ou qualquer outra mensagem equivalente).

A execução do programa não pode ser continuada. Qualquer modificação, inclusão ou remoção de linhas no programa, impede o uso do comando *CONT*.

REINICIANDO COM GOTO

A instrução *GOTO* no modo direto também pode ser usada para reiniciar um programa interrompido, preservando, também, neste caso, os valores das variáveis.

Exemplo 10.6

```
100 LET A= 10
110 LET B= 30
120 STOP
130 PRINT A
140 PRINT B
150 END
```

Rodando, temos

```
RUN

BREAK IN 120
```

Para reiniciarmos a execução do programa, utilizando a instrução *GOTO*, devemos usá-la no modo direto, indicando a linha a partir da qual a execução do programa deve continuar.


```
GOTO 130
10
30
```

Uma vantagem da instrução *GOTO* sobre o comando *CONT* é que você pode reiniciar a execução a partir de qualquer linha do programa. A desvantagem é que o comando *STOP* é mais *"automático"*. Você não precisa informar em qual linha o programa deve continuar.

E AGORA AS FUNÇÕES NUMÉRICAS

A presença de cálculos matemáticos nos programas é uma constante. Em várias áreas de aplicação, o programa se fundamenta na solução de uma ou mais funções, as quais são, muitas vezes, utilizadas em vários pontos de um mesmo programa.

Para facilitar a *"MONTAGEM"* e manipulação dessas funções foi criado em *BASIC* uma instrução que permite a definição de uma função e cujo formato é:

número de linha *DEF FN* nome (argumento) = expressão

onde:

NOME - é nome que se quer atribuir a essa função. As regras de atribuição de nome para variáveis são aplicadas também aqui.

ARGUMENTO - também utiliza um nome de variável, a qual chamaremos de variável *"POSTIÇA"* por não representar uma variável, mas sim uma *"POSIÇÃO"* que será ocupada por um valor, variável ou até mesmo uma expressão.

EXPRESSÃO - é qualquer expressão matemática válida em *BASIC*. Se esta expressão contiver uma variável com o mesmo nome da variável *"POSTIÇA"* do argumento, também essa variável será *"POSTIÇA"*, automaticamente.

Exemplos:

a.) `DEF FN A (X)= 3`

A função de nome *FN A* de variável *"POSTIÇA" X* é definida como igual a 3. Qualquer que seja o valor, variável ou expressão a substituir o argumento, teremos como resultado 3.

Assim:

```
FN A (8)= 3
FN A (Y)= 3
FN A (3 * Y + 4)=3
```

b.) `DEF FN B (X)= X + 3`

A função de nome de *FN B (X)* de variável *"POSTIÇA" X* é definida como sendo igual a variável *"POSTIÇA" X* somada com 3. Como temos na expressão uma variável *"POSTIÇA"*, o valor, variável ou expressão que substituir o argumento, também substituirá essa variável *"POSTIÇA"* da expressão.

Assim:

```
FN B (8)= (8) + 3   OU  11
FN B (Y)= (Y) + 3   OU  Y + 3
FN B(3 * Y + 4)= (3 * Y + 4)+ 3  OU 3 * Y + 7
```

A primeira expressão terá como resultado o valor 11 e as duas outras, valores dependentes do valor da variável Y. Se Y tiver o valor 5, então, teremos como resultado 8 e 22 respectivamente.

c.) `DEF FN C (X)= X + Y`

A função de nome *FN C* de variável *"POSTIÇA" X* é definida como sendo igual a variável *"POSTIÇA" X*, somada com a variável Y.

Assim:

```
FN C (8)= (8) + Y   OU  8 + Y
FN C (Y)= (Y) + Y   OU  2 * Y
FN C (3 * Y + 4)= (3 * Y + 4)+ 4 OU  4 * Y + 4
FN C (T)= (T) + Y   OU  T + Y
```

Você pode usar num programa uma variável com mesmo nome da variável *"POSTIÇA"*, pois o computador trata os dois tipos de variáveis separadamente, sem criar *"CONFUSÕES"*.

Exemplo 10.7

```
100 LET X= 10
200 LET Y= 20
300 LET Z= 30
400 DEF FN T (X)= 2 * X + Y
500 PRINT FN T (100)
600 PRINT FN T (X)
700 PRINT FN T (Y)
800 PRINT FN T (Z)
900 PRINT FN T (X + Y +Z)
1000 END
```

Rodando, temos:

```
RUN

220
40
60
80
140
```

Analisando o programa linha a linha, temos:

100 - A variável X é igualada a 10.

200 - A variável Y é igualada a 20.

300 - A variável Z é igualada a 30.

400 - A função FN T é definida.

500 - Cálculo da expressão 2 * (100) + Y para Y = 20

600 - Cálculo da expressão 2 * (X) + Y para X = 10 e Y = 20

700 - Cálculo da expressão 2 * (Y) + Y para Y = 20.

800 - Cálculo da expressão 2 * (Z) + Y para Y = 20 e Z = 30

900 - Cálculo da expressão 2 * (X + Y + Z) + Y para X = 10, Y = 20 e Z = 30.

EXERCÍCIOS

EXERCÍCIO 10.1

Explique as diferenças de operação de tratamento das variáveis dos comandos *RUN*, *CONT*, *GOTO*.

EXERCÍCIO 10.2

Quais são as condições que levam um comando *CONT* não reiniciar a operação do programa?

EXERCÍCIO 10.3

Considerando a função DEF FN A(X) = X + Y + 12 e sendo X = 12, Y = -3, T = 6, determine o valor de

F N A (X)
F N A (Y)
F N A (X + T)
F N A (10)

CAPÍTULO 11

INSTRUÇÕES E FUNÇÕES AVANÇADAS

Neste capítulo, nós vamos apresentar algumas funções e instruções em *BASIC* as quais denominamos de *"AVANÇADAS"* pois, para serem utilizadas, requerem conhecimentos sobre o *"HARDWARE"* e *"SOFTWARE"* de seu computador.

Estas instruções não são necessárias para o desenvolvimento de programas de uso geral em *BASIC* e destinam-se a aqueles que querem adquirir conhecimentos sobre o computador especificamente, e/ou ao desenvolvimento de programas em linguagem de máquina.

Para a utilização plena destas instruções e funções, você deverá dispor de dados sobre o seu computador tais, como: "mapeamento da memória", endereços e descrição das principais subrotinas do programa operacional, tipo e código de instruções do microcomputador utilizado.

Mesmo que você não pretenda tornar-se um "especialista" em computador e/ou não disponha dos dados necessários, sugerimos a leitura "descomprometida" deste capítulo, pois os conhecimentos assim adquiridos poderão lhe ser úteis no futuro.

DEFININDO O NOSSO COMPUTADOR

Como para uso das funções e instruções que serão aqui apresentadas é indispensável o conhecimento do computador onde o programa será rodado, vamos apresentar alguns dados relativos a um computador *"FICTÍCIO"* no qual supostamente seriam rodados os exemplos a serem vistos.

Mapeamento da Memória:

0 a 10239	-	Programa operacional
10240 a 11039	-	Página do vídeo
11040 a 11263	-	Reservada para o usuário
11264 a 12227	-	Dados e *"FLAGS"* do programa operacional
12228 a 22467	-	Programa *BASIC*, variáveis e dados do mesmo

Alguns exemplos das subrotinas do programa operacional:

Posição	Descrição
8340	Limpa a tela do vídeo, colocando o cursor na posição inicial.
8680	Sobe o cursor (sem movê-lo horizontalmente). Se estiver na primeira linha, permanece nesta posição.
8660	Desce o cursor (sem movê-lo horizontalmente). Se estiver na última linha, permanece nesta posição.
8670	Move o cursor para a direita numa mesma linha. Se estiver no final da linha, permanece ali.
8680	Move o cursor para a esquerda numa mesma linha. Se estiver no início da linha, permanece ali.

Localização de alguns dados e *"FLAGS"*

Posição	Descrição
11264	Fornece o *"STATUS"* do teclado. Se contiver *"ZERO"* não existe tecla pressionada, qualquer outro número identifica uma determinada tecla.
11265	Fornece a codificação em *ASCII* da última tecla pressionada. (Veja apêndice A).

Microprocessador Z 80A

Agora que já definimos nosso computador, vamos ao estudo das funções e instruções avançadas.

O CONTEÚDO DA MEMÓRIA

Cada posição ou endereço da memória de um computador pessoal pode conter um número binário de 8 dígitos, cujo valor equivalente em decimal está no intervalo de 0 a 255. O significado deste número varia não só com o seu valor, mas também com a sua posição dentro da memória. Assim, por exemplo, o número 65 quando na área de memória da página de vídeo representa a letra A, na área do programa operacional pode representar um valor ou um comando de movimentação de dados dentro do microcomputador.

A área de memória mais fácil de ser entendida e analisada é a correspondente à página de vídeo. No nosso computador *"FICTÍCIO"* essa página é composta de 20 linhas com 40 caracteres cada uma, totalizando 800 caracteres por página. De um modo geral, a primeira posição no vídeo (primeira coluna, primeira linha) corresponde ao endereço inicial da *"MEMÓRIA DE VÍDEO"*. Assim esta posição equivale ao endereço 10240, como foi definido no *"MAPEAMENTO DA MEMÓRIA"*, a posição seguinte no vídeo corresponde ao endereço seguinte na memória e, assim por diante, até a última posição e último endereço, criando uma relação direta entre a posição no vídeo e endereço da memória.

Como a memória só armazena valores binários, uma tabela de conversão dos símbolos gráficos, em valores, torna-se necessária. A tabela universalmente adotada pelos computadores pessoais é a denominada *ASCII (AMERICAN STANDARD CODE FOR INFORMATION INTERCHANGE)* e está integralmente mostrada no apêndice A e parcialmente aqui, para facilitar a sua consulta.

CARACTERES	*CÓDIGO*	*CARACTERES*	*CÓDIGO*
Espaço	32	M	77
A	65	N	78
B	66	O	79
C	67	P	80
D	68	Q	81
E	69	R	82
F	70	S	83
G	71	T	84
H	72	U	85
I	73	V	86
J	74	W	87
K	75	X	88
L	76	Y	89
		Z	90

Deste modo, a palavra *BASIC* escrita na primeira linha do vídeo estaria na memória do texto representada pelos valores *ASCII* equivalentes, ou seja:

Endereço	10240	10241	10242	10243	10244
Código	66	65	83	73	67

LENDO A MEMÓRIA

O conteúdo de um endereço da memória do computador pode ser *"LIDO"* através da função de formato:

PEEK (argumento)

onde o valor do argumento fornece a posição da memória a ser lida. Você pode usar como argumento: um valor, uma variável ou uma função.

Exemplos:

```
PRINT PEEK (11265)
13
```

O endereço de memória que escolhemos corresponde em nosso computador *"FICTÍCIO"* ao valor em *ASCII* da última tecla pressionada. O valor 13 corresponde a *RETURN* que foi a última tecla que pressionamos.

```
PRINT PEEK (10240)
66
```

O endereço fornecido corresponde à primeira posição no vídeo e o valor 66 indica que nessa posição temos a letra B.

ESCREVENDO NA MEMÓRIA

Você pode *"ESCREVER"* um dado qualquer em qualquer endereço da memória do computador através da instrução de formato:

número da linha *POKE* complemento 1, complemento 2

Complemento 1 - representa o endereço da memória onde iremos escrever. Pode ser um valor, uma variável ou uma função e cujo valor deve estar no intervalo de 0 a 65535;

Complemento 2 - representa o dado que queremos escrever na posição definida pelo complemento 1. O valor deste dado, como vimos, deve estar no intervalo de 0 a 255.

Existem áreas do computador onde o dado constante na memória não pode ser mudado (por exemplo, programa operacional) e áreas onde não é aconselhável se escrever, a menos que se saiba exatamente o que se quer fazer (por exemplo, a área reservada ao programa *BASIC)*.

Assim, vamos escrever na área reservada ao vídeo, o que além de não criar nenhum tipo de problema, possibilita uma visualização imediata do resultado.

Exemplo 11.1

```
100 FOR I= 10240 TO 11039
200 POKE I, 65
300 NEXT I
400 END
```

Este programa preenche a área de memória da posição 10240 a 11039 com o valor 65. Como esta área de memória corresponde a toda a memória de vídeo, e como o valor 65 será traduzido como a letra A, o resultado final será a tela de vídeo totalmente preenchida por essa letra.

Para obtermos o mesmo resultado no seu computador, você deverá substituir os valores inicial e final do índice I na linha 100 pelos valores inicial e final do endereço da memória de vídeo.

OBTENDO OS VALORES ASCII

O valor *ASCII* correspondente a qualquer símbolo gráfico poderá ser obtido da tabela do apêndice A, ou através da função:

ASC (argumento)

onde o argumento é uma *"STRING"* ou uma variável alfanumérica.

O valor fornecido por essa função será sempre o correspondente *ASCII* do primeiro símbolo gráfico da *"STRING"* ou da variável.

Exemplos:

```
PRINT ASC ("A")

65

PRINT ASC ("AB")

65

PRINT ASC ("BASIC")

66
```

Vamos utilizar esta função para modificar o exemplo 11.1 para preencher o vídeo com o algarismo 8, sem recorrermos a tabela *ASCII*.

Exemplo 11.2

```
100 FOR I= 10240 TO 11039
200 POKE I, ASC ("8")
300 NEXT I
400 END
```

Rodando, teremos o vídeo totalmente preenchido pelo algarismo 8.

USANDO OS VALORES ASCII

Você pode também gerar um símbolo gráfico através de uma função em *BASIC*, inversa a que vimos e, cujo formato é:

CHR $ (argumento)

onde o argumento é um valor ou uma função numérica. O valor deste argumento deve estar no intervalo 0 a 127.

Exemplo:

```
PRINT CHR$ (65)

A

PRINT CHR$ (36)

$
```

CÓDIGO DE CONTROLE

Como tabela *ASCII* representa uma codificação de valores para troca de informações, inclui, por isso, além dos símbolos gráficos, alguns códigos de controles como por exemplo, o número 10 que corresponde à mudança de linha.

Exemplo 11.3

```
100 FOR I= 1 TO 10
200 PRINT I; CHR$ (10);
300 NEXT I
400 END
```

Rodando, temos:

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

A inclusão de um código de controle numa instrução de *PRINT* faz com que a ação determinada por esse código seja executada.

Se o seu computador possuir um dispositivo de sinalização auditiva, utilizado para chamar a sua atenção para uma mensagem de erro, ou qualquer outro evento, você poderá comandá-lo através do código de controle de número 7. Todas as ve

zes que você quiser enviar um sinal sonoro use a instrução:

PRINT CHR$ (7)

Exemplo 11.4

```
100 FOR I= 1 TO 10
200 PRINT CHR$ (7)
300 FOR Y= 1 TO 100
400 NEXT Y
500 NEXT I
600 END
```

Este programa enviará 10 sinais sonoros a espaços de tempos regulares. Para aumentar ou diminuir esse espaço de tempo, modifique o valor final do índice Y na linha 300 do programa.

SUBROTINAS DO PROGRAMA OPERACIONAL

Se você dispuser da relação das subrotinas do programa operacional, você poderá utilizá-las, como subrotinas de programas em *BASIC*, através da instrução de formato:

número da linha *CALL* complemento.

onde o complemento é um valor numérico ou uma expressão numérica. Esse complemento deve corresponder ao endereço da memória onde a subrotina esta localizada.

Quando no início do capítulo, definimos o nosso computador *"FICTÍCIO"*, apresentamos uma relação de algumas subrotinas do programa operacional. Vamos, agora, para exemplificarmos o uso desta instrução, utilizar três daquelas subrotinas.

8340	limpeza de tela
8660	desce o cursor
8670	move o cursor para a direita

Exemplo 11.5

```
100 REM A SUBROTINA 8340 LIMPA A TELA
110 CAL 8340
120 REM MOVENDO O CURSOR NA TELA
130 FOR I= 1 TO 20
140 REM DESCENDO O CURSOR
150 CALL 8660
160 REM MOVENDO PARA A DIREITA
170 CALL 8670
180 FOR X= 1 TO 100
190 NEXT X
200 NEXT I
210 END
```

Rodando este programa, teremos como resultado inicial a limpeza da tela do vídeo e a colocação do cursor na primeira posição, após teremos o deslocamento, em diagonal, do cursor na tela do vídeo. As linhas do programa de número 160 e 170 produzem um atraso na execução, possibilitando a visualização deste deslocamento.

EXERCÍCIOS

EXERCÍCIO 11.1

Utilizando a tabela ASCII, escreva um programa que sem o uso da instrução *PRINT* envie para o vídeo, na primeira linha, a frase *MEU PRIMEIRO PROGRAMA*.

EXERCÍCIO 11.2

Escreva um programa que resulte igual ao do Exercício 11.1, sem no entanto utilizar a tabela ASCII.

EXERCÍCIO 11.3

Escreva um programa que identifique se é letra ou número um caractere entrado via teclado.

EXERCÍCIO 11.4

Escreva um programa que coloque a letra A no centro da tela do vídeo e que permita o comando do deslocamento no vídeo desta letra para cima, para baixo, para direita ou para a esquerda, teclando-se as letras C, B, D e E, respectivamente.

APÊNDICE 1

PROGRAMAS E JOGOS

APÊNDICE 1

PROGRAMAS E JOGOS

DECOMPOSIÇÃO EM FATORES PRIMOS — I

Qualquer número inteiro e positivo pode ser igualado a um produto de números primos. Por exemplo, o número 6 pode ser obtido pelo produto dos números primos 2 e 3, o número 48 é resultante de 2x2x2x2x3, ou seja: 2 elevado a 4, vezes 3.

A operação matemática para obtenção desses números primos é denominada *DECOMPOSIÇÃO EM FATORES PRIMOS*, a qual todos nós vimos no Primeiro Grau.

Este programa realiza esta Decomposição através do método de divisões sucessivas pelos fatores primos divisores do número.

Rodando este programa, temos:

```
XXXXXX FATORES PRIMOS XXXXXX


ENTRE COM UM NUMERO INTEIRO
E POSITIVO E TECLE RETURN

?1200
2 ELEVADO A 4
3 ELEVADO A 1
5 ELEVADO A 2

NOVO NUMERO? (S OU N)

?S

ENTRE COM UM NUMERO INTEIRO
E POSITIVO E TECLE RETURN

?30630600
2 ELEVADO A 3
3 ELEVADO A 2
5 ELEVADO A 2
7 ELEVADO A 1
11 ELEVADO A 1
13 ELEVADO A 1
17 ELEVADO A 1

NOVO NUMERO? (S OU N)

?S

ENTRE COM UM NUMERO INTEIRO
E POSITIVO E TECLE RETURN

?5081436
2 ELEVADO A 2
3 ELEVADO A 2
17 ELEVADO A 1
19 ELEVADO A 2
23 ELEVADO A 1

NOVO NUMERO? (S OU N)

?N

FOI UM PRAZER. ATE A PROXIMA VEZ
```



```
1000  REM  FATORES PRIMOS DE UM NUMERO
1100  PRINT
1200  PRINT "XXXXXX FATORES PRIMOS XXXXXX"
1300  PRINT
1400  PRINT
1500  PRINT "ENTRE COM UM NUMERO INTEIRO"
1600  PRINT "E POSITIVO E TECLE RETURN"
1700  PRINT
1800  INPUT N
1900  REM  VERIFICACAO SE INTEIRO
2000  IF  INT (N) = N GOTO 2400
2100  PRINT
2200  PRINT "VAMOS LA MEU CHAPA... NUMERO INTEIRO!!"
2300  GOTO 1400
2400  REM  VERIFICACAO SE POSITIVO
2500  IF N > 0 GOTO 2900
2600  PRINT
2700  PRINT "QUE GRACINHA... NUMERO POSITIVO!!"
2800  GOTO 1400
2900  REM  CALCULOS DOS FATORES
2950  LET I = 2
3000  FOR I = I TO N
3100  J = 0
3200  IF N / I <  >  INT (N / I) GOTO 3600
3300  N = N / I
3400  J = J + 1
3500  GOTO 3200
3600  IF J = 0 GOTO 3800
3700  PRINT I;" ELEVADO A ";J
3750  GOTO 3000
3800  NEXT I
3900  PRINT
4000  PRINT "NOVO NUMERO? (S OU N)"
4100  PRINT
4200  INPUT R$
4300  IF R$ = "S" GOTO 1400
4400  IF R$ = "N" GOTO 4800
4500  PRINT
4600  PRINT "POR FAVOR, RESPONDA S OU N"
4700  GOTO 4100
4800  PRINT
4900  PRINT "FOI UM PRAZER. ATE A PROXIMA VEZ"
5000  END
```

DECOMPOSIÇÃO EM FATORES PRIMOS — II

Este programa também efetua a decomposição de um número inteiro e positivo em seus fatores primos. A diferença em relação ao programa anterior é que a apresentação do resultado é no formato tradicionalmente utilizado no Primeiro Grau.

Rodando este programa, temos:

```
DECOMPOSICAO EM FATORES PRIMOS

ENTRE COM UM NUMERO INTEIRO
E POSITIVO E TECLE RETURN

?1200


1200 ! 2
 600 ! 2
 300 ! 2
 150 ! 2
  75 ! 3
  25 ! 5
   5 ! 5
   1 !

NOVO NUMERO? (S OU N)

?S

ENTRE COM UM NUMERO INTEIRO
E POSITIVO E TECLE RETURN

?91520


91520 ! 2
45760 ! 2
22880 ! 2
11440 ! 2
 5720 ! 2
 2860 ! 2
 1430 ! 2
  715 ! 5
  143 ! 11
   13 ! 13
    1 !

NOVO NUMERO? (S OU N)

?N
OBRIGADO, FOI UM PRAZER SERVI-LO
```

```
1000  REM  FATORES PRIMOS DE UM NUMERO
1100  PRINT
1200  PRINT "DECOMPOSICAO EM FATORES PRIMOS"
1300  PRINT
1400  PRINT
1500  PRINT "ENTRE COM UM NUMERO INTEIRO"
1600  PRINT "E POSITIVO E TECLE RETURN"
1700  PRINT
1800  INPUT N
1850  PRINT
1900  REM  VERIFICACAO SE INTEIRO E POSITIVO
2000  IF  INT (N) = N AND N > 0 GOTO 2800
2100  REM  ADEQUAR O NUMERO
2200  LET N =  INT ( ABS (N))
2400  PRINT
2500  PRINT "COMO O NUMERO DEVE SER INTEIRO"
2600  PRINT "E POSITIVO, VOU EFETUAR A DECOMPOSICAO"
2700  PRINT " DO NUMERO ";N
2800  REM  DECOMPOSICAO DO NUMERO
2850  PRINT
2900  LET I = 2
2950  LET A =  LEN ( STR$ (N))
3000  FOR I = I TO N
3100  IF N / I <  > INT (N / I) GOTO 3500
3200  PRINT  SPC( A -  LEN ( STR$ (N)));N;" ! ";I
3300  N = N / I
3400  GOTO 3000
3500  NEXT I
3600  PRINT  SPC( A - 2);" 1 !"
3700  PRINT
3800  PRINT
3900  PRINT "NOVO NUMERO? (S OU N)"
4000  PRINT
4100  INPUT R$
4200  IF R$ = "SIM" OR R$ = "S" GOTO 1400
4300  IF R$ = "NAO" OR R$ = "N" GOTO 4700
4400  PRINT
4500  PRINT "RESPONDA S OU N"
4600  GOTO 4000
4700  PRINT "OBRIGADO, FOI UM PRAZER SERVI-LO"
4800  END
```


SISTEMA DE EQUAÇÕES DE PRIMEIRO GRAU COM DUAS INCÓGNITAS

A obtenção do ponto de intersecção de duas retas em um plano, equivale a achar os valores das coordenadas X e Y que satisfazem simultaneamente estas equações.

A equação genérica de uma reta num plano é

$$A.X + B.Y = C$$

Assim, obter o ponto de intersecção de duas retas num plano é resolver o sistema:

$$\text{RETA 1} \quad A_1.X + B_1.Y = C_1$$
$$\text{RETA 2} \quad A_2.X + B_2.Y = C_2$$

Os valores de X e Y obtidos representam as coordenadas do ponto de intersecção das retas.

Este programa resolve sistemas de equações a duas incógnitas, pelo método matricial.

Rodando o programa, temos:

```
******** EQUACOES PRIMEIRO GRAU ********
          COM DUAS INCOGNITAS

FORMATO:
        A1.X + B1.Y = C1
        A2.X + B2.Y = C2

ENTRE COM OS VALORES
VALOR DE A1
?2
VALOR DE B1
?1
VALOR DE C1
?3
VALOR DE A2
?1
VALOR DE B2
?1
VALOR DE C2
?1

VALOR DE X 2
VALOR DE Y -1

NOVO CALCULO ? ( S OU N ) ?S

ENTRE COM OS VALORES
VALOR DE A1
?5
VALOR DE B1
?-15
VALOR DE C1
?-250
VALOR DE A2
?-2
VALOR DE B2
?-5
VALOR DE C2
?-120

VALOR DE X 10
VALOR DE Y 20

NOVO CALCULO ? ( S OU N ) ?N
```



```
99  REM  SISTEMA DE DUAS EQUACOES
100  DATA  A1,B1,C1,A2,B2,C2
110  PRINT "******** EQUACOES PRIMEIRO GRAU ********"
120  PRINT "           COM DUAS INCOGNITAS "
130  PRINT
140  PRINT "FORMATO: "
150  PRINT "         A1.X + B1.Y = C1 "
160  PRINT "         A2.X + B2.Y = C2 "
170  PRINT
175  RESTORE
179  REM  ENTRADA DOS COEFICIENTES
180  PRINT "ENTRE COM OS VALORES"
190  FOR I = 1 TO 6
200  READ A$
210  PRINT "VALOR DE ";A$
220  INPUT A(I)
230  NEXT
239  REM  SOLUCAO
240  LET D = A(1) * A(5) - A(4) * A(2)
250  IF D = 0 THEN  GOTO 380
260  LET X = A(3) * A(5) - A(6) * A(2)
270  LET Y = A(1) * A(6) - A(4) * A(3)
280  PRINT
290  PRINT "VALOR DE X "; INT (100 * X / D) / 100
300  PRINT "VALOR DE Y "; INT (100 * Y / D) / 100
310  PRINT
320  PRINT "NOVO CALCULO ? ( S OU N ) ";
330  INPUT R$
340  IF R$ = "S" GOTO 170
360  GOTO 400
370  PRINT
380  PRINT "SISTEMA INDEFINIDO"
390  GOTO 310
400  END
```

EQUAÇÃO DO SEGUNDO GRAU

Este programa calcula os valores das raízes de uma equação do segundo grau de formato:

$$A.X^2 + B.X + C = 0$$

Você deverá fornecer os valores dos coeficientes e as raízes são calculadas, sendo informado, inclusive, se são raízes reais ou complexas.

Rodando o programa, temos:

```
EQUACAO  DO  SEGUNDO GRAU

SOLUCAO  DA  EQUACAO DA FORMA:

A.X^2 +B.X +C =0

(X^2 SIGNIFICA X AO QUADRADO)

ENTRE COM OS VALORES DOS COEFICIENTES

NO FORMATO: A, B, C

?6,-90,264
RAIZES REAIS
              4
              11

NOVO CALCULO? (S OU N)?S

ENTRE COM OS VALORES DOS COEFICIENTES

NO FORMATO: A, B, C

?1,1,-12
RAIZES REAIS
             -4
              3

NOVO CALCULO? (S OU N)?S

ENTRE COM OS VALORES DOS COEFICIENTES

NO FORMATO: A, B, C

?2,20,42
RAIZES REAIS
             -7
             -3

NOVO CALCULO? (S OU N)?S

ENTRE COM OS VALORES DOS COEFICIENTES

NO FORMATO: A, B, C

?8,-26,-169
RAIZES REAIS
             -3.25
              6.5

NOVO CALCULO? (S OU N)?S

ENTRE COM OS VALORES DOS COEFICIENTES

NO FORMATO: A, B, C
```

```
?4,4,-48
RAIZES REAIS
             -4
             3

NOVO CALCULO? (S OU N)?S

ENTRE COM OS VALORES DOS COEFICIENTES

NO FORMATO: A, B, C

?2,4,-6
RAIZES REAIS
             -3
             1

NOVO CALCULO? (S OU N)?N
```

```
100  PRINT "EQUACAO  DO  SEGUNDO GRAU"
110  PRINT
120  PRINT "SOLUCAO  DA  EQUACAO DA FORMA:"
130  PRINT
140  PRINT "A.X^2 +B.X +C =0"
150  PRINT
160  PRINT "(X^2 SIGNIFICA X AO QUADRADO)"
170  PRINT
180  PRINT "ENTRE COM OS VALORES DOS COEFICIENTES"
190  PRINT
200  PRINT "NO FORMATO: A, B, C"
210  PRINT
220  INPUT A,B,C
230  REM  CALCULO DO DETERMINANTE
240  LET D = B ^ 2 - 4 * A * C
250  IF D < 0 GOTO 320
260  REM  RAIZES REAIS
270  PRINT "RAIZES REAIS"
275  LET D =  SQR (D)
280  PRINT  SPC( 13);( - B - D) / (2 * A)
290  PRINT  SPC( 13);( - B + D) / (2 * A)
300  GOTO 360
310  REM    RAIZES IMAGINARIAS
320  PRINT "RAIZES COMPLEXAS"
325  LET D =  SQR ( - D)
330  PRINT  SPC( 17); - B / (2 * A);" + "D / (2 * A);"I"
340  PRINT  SPC( 17); - B / (2 * A);" - "D / (2 * A);"I"
350  REM  FINAL
360  PRINT
370  PRINT "NOVO CALCULO? (S OU N)";
380  INPUT R$
390  IF R$ = "S" GOTO 170
400  END
```


Dado dois números inteiros e positivos, qual é o maior número que os divide sem deixar resto ? Bem, como você sabe, este número é o chamado *Máximo Divisor Comum*.

Este programa calcula o *Máximo Divisor Comum* entre dois números, utilizando o *ALGORITMO DE EUCLIDES*.

Rodando o programa, temos:

```
XXXXXXXXXXXXXXXXXXXXXXX
X MAXIMO DIVISOR COMUM X
XXXXXXXXXXXXXXXXXXXXXXX

ENTRE COM  DOIS  NUMEROS
INTEIROS E POSITIVOS QUE
EU  CALCULAREI  O  M.D.C.

N1, N2?12400,2348

O M.D.C. E'= 4

NOVO CALCULO? (S OU N) ?S

ENTRE COM  DOIS  NUMEROS
INTEIROS E POSITIVOS QUE
EU  CALCULAREI  O  M.D.C.

N1, N2?1729,3059

O M.D.C. E'= 133

NOVO CALCULO? (S OU N) ?N
```

```
1000  PRINT "XXXXXXXXXXXXXXXXXXXXXXX"
1100  PRINT "X MAXIMO DIVISOR COMUM X"
1200  PRINT "XXXXXXXXXXXXXXXXXXXXXXX"
1300  PRINT
1400  PRINT "ENTRE COM  DOIS  NUMEROS"
1500  PRINT "INTEIROS E POSITIVOS QUE"
1600  PRINT "EU  CALCULAREI  O  M.D.C."
1650  PRINT
1700  PRINT "N1, N2";
1800  INPUT N1,N2
1900  IF N1 = 0 OR N2 = 0 GOTO 1300
2000  REM  ADEQUACAO DOS NUMEROS
2100  LET N1 =  INT ( ABS (N1))
2200  LET N2 =  INT ( ABS (N2))
2300  IF N1 > N2 GOTO 2800
2400  LET N = N1
2500  LET N1 = N2
2600  LET N2 = N
2700  REM   ALGORITMO DE EUCLIDES
2800  LET N = N1 - N2 *  INT (N1 / N2)
2900  IF N = 0 GOTO 3200
3000  GOTO 2500
3100  REM  PARTE FINAL
3200  PRINT
3300  PRINT "O M.D.C. E'= ";N2
3400  PRINT
3500  PRINT "NOVO CALCULO? (S OU N) ";
3600  INPUT R$
3700  IF R$ = "S" GOTO 1300
3800  END
```


CONVERSAO DE BASE

Nós utilizamos em nosso dia a dia o sistema decimal de numeração. A explicação histórica é que os homens pré-históricos utilizavam os dedos das mãos para fazerem cálculos, originando assim a base 10, permanecendo até hoje esse uso. (Não só da base 10, mas também cálculos com os dedos).

Junto a esse nosso dia a dia, existem outros sistemas de numeração que não fazem uso da base decimal, principalmente no mundo da computação, onde temos, por exemplo:

Sistema Binário - base 2
Sistema Octal - base 8
Sistema Hexadecimal - base 16

Este programa converte qualquer número decimal (positivo e inteiro) no número equivalente, em qualquer base de 2 a 16.

Rodando o programa, temos:

```
********* CONVERSAO DE BASE ********

  DE DECIMAL PARA QUALQUER OUTRA

         BASE ENTRE 2 E 16

     (NUMERO POSITIVO E INTEIRO)

************************************

ENTRE COM O NUMERO E A BASE

NO FORMATO N,B ?234561,2

EQUIVALENTE NA BASE 2

1110010100010000001

NOVA CONVERSAO ? (S OU N ) ?S

ENTRE COM O NUMERO E A BASE

NO FORMATO N,B ?25600024,2

EQUIVALENTE NA BASE 2

110000110101000000011000

NOVA CONVERSAO ? (S OU N ) ?N
```

```
100   PRINT "********* CONVERSAO DE BASE ********"
200   PRINT
300   PRINT "  DE DECIMAL PARA QUALQUER OUTRA"
400   PRINT
500   PRINT "          BASE ENTRE 2 E 16"
600   PRINT
700   PRINT "     (NUMERO POSITIVO E INTEIRO)"
800   PRINT
900   PRINT "************************************"
1000  REM  ENTRADA DO VALOR DO NUMERO E DA BASE
1100  PRINT
1200  PRINT "ENTRE COM O NUMERO E A BASE"
1300  PRINT
1400  PRINT "NO FORMATO N,B ";
1500  INPUT N,B
1600  REM  ADEQUAR VALORES
1700  LET N =  INT ( ABS (N))
1800  LET B =  INT ( ABS (B))
1900  IF B < 2 OR B > 16 THEN  GOTO 1200
2000  REM   CALCULO DO EXPOENTE MAXIMO
2100  LET E = 0
2200  IF B ^ E > N GOTO 2500
2300  LET E = E + 1
2400  GOTO 2200
2500  REM  ALGORITIMO DE CONVERSAO
```



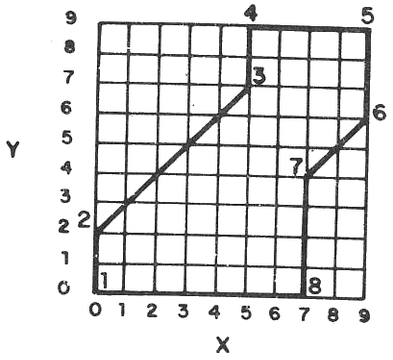
```
2510  PRINT
2520  PRINT "EQUIVALENTE NA BASE ";B
2530  PRINT
2550  FOR I = 1 TO E
2600  LET C =  INT (N / B ^ (E - I))
2700  LET N = N - (C * B ^ (E - I))
2800  IF C > 9 GOTO 3100
2900  PRINT C;
3000  GOTO 3200
3100  IF C = 10 THEN  PRINT "A";
3110  IF C = 11 THEN  PRINT "B";
3120  IF C = 12 THEN  PRINT "C";
3130  IF C = 13 THEN  PRINT "D";
3140  IF C = 14 THEN  PRINT "E";
3150  IF C = 15 THEN  PRINT "F";
3200  NEXT I
3300  PRINT
3400  PRINT
3500  PRINT "NOVA CONVERSAO ? (S OU N ) ";
3600  INPUT R$
3700  IF R$ = "S" THEN  GOTO 1100
3800  END
```

Este programa lhe permite calcular a área de uma figura poligonal de até 45 vértices. (Para um número maior de vértices, mude adequadamente a linha 10 do programa).

Os dados para obtenção deste cálculo são as coordenadas dos vértices do polígono, e o valor calculado da área é na unidade do sistema de coordenadas.

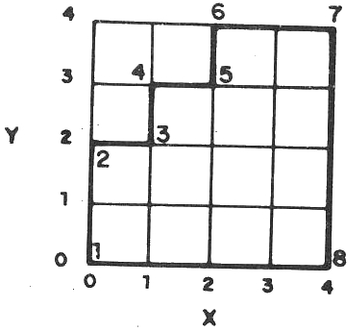
Para exemplificar e tornar mais claro o modo de apresentação dos dados, vamos utilizar o programa para cálculo das áreas das duas figuras abaixo.

Figura 1



Vértice 1 *(0,0)*
Vértice 2 *(0,2)*
Vértice 3 *(5,7)*
Vértice 4 *(5,9)*
Vértice 5 *(9,9)*
Vértice 6 *(9,6)*
Vértice 7 *(7,4)*
Vértice 8 *(7,0)*

Figura 2



Vértice 1 *(0,0)*
Vértice 2 *(0,2)*
Vértice 3 *(1,2)*
Vértice 4 *(1,3)*
Vértice 5 *(2,3)*
Vértice 6 *(2,4)*
Vértice 7 *(4,4)*
Vértice 8 *(4,0)*

Rodando o programa, temos:


```
********** AREA DE UM POLIGONO *********
           DE ATE 45 VERTICES

QUANTOS VERTICES?
?8
ENTRE AS COORDENADAS
NO FORMATO X,Y
VERTICE 1
?0,0
VERTICE 2
?0,2
VERTICE 3
?5,7
VERTICE 4
?5,9
VERTICE 5
?9,9
VERTICE 6
?9,6
VERTICE 7
?7,4
VERTICE 8
?7,0

AREA DO POLIGONO 48.5

NOVO CALCULO DE AREA ? (S OU N ) ?S
QUANTOS VERTICES?
?8
ENTRE AS COORDENADAS
NO FORMATO X,Y
VERTICE 1
?0,0
VERTICE 2
?0,2
VERTICE 3
?1,2
VERTICE 4
?1,3
VERTICE 5
?2,3
VERTICE 6
?2,4
VERTICE 7
?4,4
VERTICE 8
?4,0

AREA DO POLIGONO 13

NOVO CALCULO DE AREA ? (S OU N ) ?N
```

```
10   DIM X(46),Y(46)
100  PRINT "********** AREA DE UM POLIGONO *********"
110  PRINT "            DE ATE 45 VERTICES"
120  PRINT
130  PRINT "QUANTOS VERTICES?"
140  INPUT N
145  IF N < 3 GOTO 130
150  PRINT "ENTRE AS COORDENADAS"
155  PRINT "NO FORMATO X,Y"
160  FOR I = 0 TO N - 1
170  PRINT "VERTICE ";I + 1
180  INPUT X(I),Y(I)
190  NEXT I
200  REM  FECHANDO O POLIGONO
210  LET X(I) = X(0)
220  LET Y(I) = Y(0)
230  REM  CALCULO DA AREA
240  LET A = 0
250  FOR I = 0 TO N - 1
260  LET A = A + (X(I) + X(I + 1)) * (Y(I) - Y(I + 1))
270  NEXT I
280  PRINT
290  PRINT "AREA DO POLIGONO "; ABS (A / 2)
300  PRINT
310  PRINT "NOVO CALCULO DE AREA ? (S OU N ) ";
320  INPUT R$
330  IF R$ = "S" GOTO 130
340  END
```


FORCA

Este programa é o tradicional *Jogo da Forca*, só que ao invés de uma figura, a sua condição é descrita como por exemplo *"APROVEITE A ÚLTIMA VISÃO DO MUNDO"*.

A descrição de cada situação, evidentemente pode ser mudada para algo mais ao seu gosto, bastando para isso mudar as linhas da subrotina de 670 a 790.

Cada vez que o jogo inicia, a mesma seqüência de palavras é usada; e , para mudar essas palavras mude simplesmente as palavras constantes da instrução *DATA* da linha 810.

Rodando o programa, temos:

```
--- ENFORCADO ----
VOCE TEM 7 CHANCES

----------

QUAL LETRA ?A

----------

SUA HORA FINAL CHEGOU

QUAL LETRA ?E

-E-E--E--E

SUA HORA FINAL CHEGOU

QUAL LETRA ?O

-E-E--E--E

DESEJE UM BOM DIA AO CARRASCO

QUAL LETRA ?I

-E-E--E--E

CAPRICHARAM NA FORCA, NAO?

QUAL LETRA ?D

DE-E--E--E

CAPRICHARAM NA FORCA, NAO?

QUAL LETRA ?T

DETE--E-TE

CAPRICHARAM NA FORCA, NAO?

QUAL LETRA ?R

DETER-E-TE

CAPRICHARAM NA FORCA, NAO?

QUAL LETRA ?G

DETERGE-TE

CAPRICHARAM NA FORCA, NAO?

QUAL LETRA ?N
```



```
DETERGENTE

O ENFORCAMENTO FOI CANCELADO! PARABENS!!

NOVA PALAVRA? (S OU N) ?S

----------

QUAL LETRA ?A

------A-A---

QUAL LETRA ?E

-E----A-A--E

QUAL LETRA ?I

-E----A-A--E

SUA HORA FINAL CHEGOU

QUAL LETRA ?O

-E---OA-A--E

SUA HORA FINAL CHEGOU

QUAL LETRA ?U

-E---OA-A--E

DESEJE UM BOM DIA AO CARRASCO

QUAL LETRA ?T

-E-T-OA-A-TE

DESEJE UM BOM DIA AO CARRASCO

QUAL LETRA ?T

ESTA JA FOI
QUAL LETRA ?P

-E-T-OA-A-TE

CAPRICHARAM NA FORCA, NAO?

QUAL LETRA ?G

-E-T-OA-A-TE
```

```
APROVEITE A ULTIMA VISAO DO MUNDO

QUAL LETRA ?J

-E-T-OA-A-TE

O CAPUZ FOI COLOCADO

QUAL LETRA ?F

-E-T-OA-A-TE

A CORDA ESTA NO LUGAR...

QUAL LETRA ?V

-E-T-OAVA-TE

A CORDA ESTA NO LUGAR...

QUAL LETRA ?D

-E-T-OAVA-TE

VOCE FOI ENFORCADO!!

A PALAVRA E' CENTROAVANTE

NOVA PALAVRA? (S OU N) ?N
```

```
100  REM  JOGO DA FORCA
110  DIM L$(50)
120  REM  TITULOS
130  PRINT "--- ENFORCADO ----"
140  PRINT "VOCE TEM 7 CHANCES"
150  PRINT
160  REM  LER A PALAVRA
170  READ P$
180  REM  MOSTRAR NUMERO DE LETRAS
190  FOR I = 1 TO  LEN (P$)
200  PRINT "-";
210  NEXT I
220  PRINT
230  PRINT
240  REM  ENTRADA E CONTROLE DA LETRA
250  LET X = 0
255  LET A = 0
260  PRINT
270  PRINT "QUAL LETRA";" ";
280  INPUT L$
290  FOR I = 0 TO X
300  IF L$ = L$(I) GOTO 350
310  NEXT I
320  LET X = X + 1
330  LET L$(X) = L$
340  GOTO 380
350  PRINT
360  PRINT "ESTA JA FOI"
370  GOTO 270
380  REM  LETRA CORRETA?
390  FOR Y = 1 TO  LEN (P$)
400  IF L$ =  MID$ (P$,Y,1) GOTO 430
410  NEXT Y
420  GOTO 440
430  LET A = A + 1
440  REM  MOSTRAR AS LETRAS NA PALAVRA
442  LET L = 0
444  PRINT
450  FOR J = 1 TO  LEN (P$)
460  FOR I = 1 TO X
470  IF L$(I) =  MID$ (P$,J,1) GOTO 490
480  GOTO 510
490  PRINT L$(I);
495  LET L = L + 1
500  GOTO 530
510  NEXT I
520  PRINT "-";
530  NEXT J
540  PRINT
545  PRINT
550  REM  CONTROLES DE ERROS E FINAL
552  IF L =  LEN (P$) GOTO 560
554  IF X < A GOTO 260
556  GOTO 570
560  REM  PALAVRA CORRETA
564  PRINT
566  PRINT "O ENFORCAMENTO FOI CANCELADO! PARABENS!!"
568  GOTO 610
570  ON (X - A) GOSUB 670,690,710,730,750,770,790
580  IF (X - A) < 7 GOTO 260
590  PRINT
600  PRINT "A PALAVRA E' ";P$
610  PRINT
620  PRINT "NOVA PALAVRA? (S OU N) ";
630  INPUT R$
640  IF R$ = "S" GOTO 150
650  GOTO 900
660  REM  SUBROTINAS
670  PRINT "SUA HORA FINAL CHEGOU"
680  RETURN
690  PRINT "DESEJE UM BOM DIA AO CARRASCO"
700  RETURN
710  PRINT "CAPRICHARAM NA FORCA, NAO?"
720  RETURN
730  PRINT "APROVEITE A ULTIMA VISAO DO MUNDO"
740  RETURN
750  PRINT "O CAPUZ FOI COLOCADO"
760  RETURN
770  PRINT "A CORDA ESTA NO LUGAR..."
780  RETURN
```

```
790  PRINT "VOCE FOI ENFORCADO!!"
800  RETURN
810  DATA  DETERGENTE, CENTROAVANTE, GENGIBRE, JENIPAP
     O, PORCALHONA, QUEIXA, TERMONUCLEAR, EXADREZADO,
     HELIPORTO, MICROCIRURGIA, RUBIACEA, BIJUTERIA, TE
     STEMUNHA, BAMBUZINHO, VIPERINO, XINGATORIO, ZOMBE
     TEIRO, AJUIZADO.
900  END
```


Este programa simula a operação das famosas máquinas *"CAÇA-NÍQUEL"*, aqui apelidadas de *"PAPA-NÍQUEL"*.

Os prêmios pagos são:

TRÊS BARRAS	*SEUS 10 + 1000*
DUAS BARRAS	*SEUS 10 + 50*
QUALQUER TRIPLA	*SEUS 10 + 100*
QUALQUER DUPLA	*SEUS 10 + 20*

Qualquer dupla junto com *SINO*, não é paga.

Obs.:

O efeito deste jogo no vídeo é diferente do aqui mostrado, pois no vídeo a página permanece fixa, o que não pode ser feito com a impressora.

BOA SORTE

Rodando o programa, temos:

```
*** MAQUINA PAPA-NIQUEL ***

  10 CRUZEIROS CADA APOSTA

QUANTO DINHEIRO VOCE TEM?

?60
CUIDADO COM O LEITE DAS CRIANCAS

PARA RODAR A MAQUINA TECLE RETURN
PARA TERMINAR O JOGO TECLE N E RETURN

?
XXXXXXX XXXXXXX XXXXXXX
 AMEIXA XXXXXXX XXXXXXX
 AMEIXA  UVA XXXXXXX
 AMEIXA  UVA  LARANJA

NADA
SEU SALDO E' 50

?
XXXXXXX XXXXXXX XXXXXXX
 UVA XXXXXXX XXXXXXX
 UVA  AMEIXA XXXXXXX
 UVA  AMEIXA  AMEIXA

UMA DUPLINHA MAS VALE 30 MANGOS
SEU SALDO E' 80

?
XXXXXXX XXXXXXX XXXXXXX
 UVA XXXXXXX XXXXXXX
 UVA  BARRA XXXXXXX
 UVA  BARRA  LIMA

NADA
SEU SALDO E' 70

?
XXXXXXX XXXXXXX XXXXXXX
 AMEIXA XXXXXXX XXXXXXX
 AMEIXA  UVA XXXXXXX
 AMEIXA  UVA  UVA

UMA DUPLINHA MAS VALE 30 MANGOS
SEU SALDO E' 100

?
XXXXXXX XXXXXXX XXXXXXX
 CEREJA XXXXXXX XXXXXXX
 CEREJA  BARRA XXXXXXX
 CEREJA  BARRA  UVA

NADA
SEU SALDO E' 90

?N
```



```
5  DIM M$(10)
10  LET M$(0) = " BARRA"
11  LET M$(1) = " SINO"
12  LET M$(2) = " AMEIXA"
13  LET M$(3) = " LARANJA"
14  LET M$(4) = " LIMA"
15  LET M$(5) = " CEREJA"
16  LET M$(6) = " UVA"
17  LET M$(7) = " AMORA"
18  LET M$(8) = " LIMAO"
19  HOME
100  PRINT "*** MAQUINA PAPA-NIQUEL ***"
110  PRINT
120  PRINT "  10 CRUZEIROS CADA APOSTA"
130  PRINT
140  PRINT "QUANTO DINHEIRO VOCE TEM? "
150  PRINT
160  INPUT D
170  IF D < 20 GOTO 220
180  IF D < 200 GOTO 240
190  IF D < 5000 GOTO 260
200  PRINT "MILIONARIOS QUEIRAM DIRIGIR-SE AO POKER"
210  GOTO 820
220  PRINT "ACOMPANHE ESTE SENHOR A PORTA DE SAIDA"
230  GOTO 820
240  PRINT "CUIDADO COM O LEITE DAS CRIANCAS"
250  GOTO 270
260  PRINT "SEJA BEM VINDO! (HA! HA! MAIS UM PATO)"
270  PRINT
280  PRINT "PARA RODAR A MAQUINA TECLE RETURN"
290  PRINT "PARA TERMINAR O JOGO TECLE N E RETURN"
300  PRINT
310  INPUT R$
320  IF R$ = "N" GOTO 820
330  LET X =  INT (9 *  RND (1))
340  LET Y =  INT (9 *  RND (1))
350  LET Z =  INT (9 *  RND (1))
360  VTAB 15
370  PRINT "XXXXXXX XXXXXXX XXXXXXX"
380  GOSUB 900
390  VTAB 15
400  PRINT M$(X)$" XXXXXXX XXXXXXX"$ SPC( 3)$""
410  GOSUB 900
420  VTAB 15
430  PRINT M$(X)$" "$M$(Y)$" XXXXXXX"$ SPC( 5)$""
440  GOSUB 900
450  VTAB 15
460  PRINT M$(X)$" "$M$(Y)$" "$M$(Z)$ SPC( 12)$""
465  PRINT
470  IF X = 1 OR Y = 1 OR Z = 1 GOTO 730
480  IF X = Y AND Y = Z GOTO 580
490  IF X = Y OR X = Z GOTO 650
500  IF Y = Z GOTO 690
510  PRINT "NADA"$ SPC( 36)$""
520  LET D = D - 10
530  IF D < 10 GOTO 560
540  PRINT "SEU SALDO E' "$D$ SPC( 6)$""
```

```
550  GOTO 300
560  PRINT "PERDESTE TUDO MEU CHAPA, SAIA DE FININHO"
570  GOTO 820
580  IF X = 0 GOTO 620
590  PRINT "UMA TRIPLA - 110 MANGOS"; SPC( 17);""
600  LET D = 110 + D
610  GOTO 540
620  PRINT "OBA.PREMIO MAXIMO TRIPLA DE BARRAS +1010"
630  LET D = 1010 + D
640  GOTO 540
650  IF X = 0 GOTO 700
660  PRINT "UMA DUPLINHA MAS VALE 30 MANGOS " SPC( 8);""
670  LET D = 30 + D
680  GOTO 540
690  IF Y <  > 0 GOTO 660
700  PRINT "NADA MAL UMA DUPLA DE BARRAS. EMBOLSA 60 "
710  LET D = 60 + D
720  GOTO 540
730  PRINT "O SINO BADALA E VOCE NAO GANHA " SPC( 10);""
740  GOTO 520
820  END
900  FOR I = 1 TO 500
910  NEXT I
920  RETURN
```


A colocação de um grande número de palavras ou nomes em ordem alfabética é uma tarefa que, além de trabalhosa, requer sobretudo uma certa dose de paciência.

QUE TAL DEIXAR ESSE TRABALHO PARA O COMPUTADOR FAZER ?

Este programa aceita até 50 palavras ou nomes, e, por um *"TOQUE DE MÁGICA"*, os coloca em ordem alfabética em *"QUESTÃO DE SEGUNDOS"*.

Para um número maior de palavras, simplesmente mude o valor do dimensionamento na linha 110 do programa.

Rodando o programa, temos:

```
**********************
*  ORDEM ALFABETICA   *
**********************

ESTE PROGRAMA  ORGANIZA
EM ORDEM ALFABETICA ATE
50  PALAVRAS  OU  NOMES

QUANTOS NOMES/PALAVRAS?
?5
POR FAVOR, ENTRE COM OS
DADOS  LINHA  A  LINHA
?WERNER VON SIEMENS
?ALBERTO SANTOS DUMONT
?ALBERT EINSTEIN
?ROBERT FULTON
?ERNEST RUTHERFORD

A ORDEM ALFABETICA E':

ALBERT EINSTEIN
ALBERTO SANTOS DUMONT
ERNEST RUTHERFORD
ROBERT FULTON
WERNER VON SIEMENS
```

```
10  DIM A$(50)
99  REM  MENSAGENS INICIAIS
100  PRINT "**********************"
110  PRINT "*  ORDEM ALFABETICA   *"
120  PRINT "**********************"
130  PRINT
140  PRINT "ESTE PROGRAMA  ORGANIZA"
150  PRINT "EM ORDEM ALFABETICA ATE"
155  PRINT "50  PALAVRAS  OU  NOMES"
160  PRINT
169  REM  ENTRADA DE DADOS
170  PRINT "QUANTOS NOMES/PALAVRAS?"
180  INPUT N
185  IF N = 0 THEN  GOTO 500
190  PRINT "POR FAVOR, ENTRE COM OS"
200  PRINT "DADOS  LINHA  A  LINHA"
210  FOR I = 0 TO N - 1
230  INPUT A$(I)
240  NEXT I
250  REM  COLOCANDO EM ORDEM ALFABETICA
260  FOR I = 0 TO N - 2
270  FOR J = I + 1 TO N - 1
280  IF A$(I) < A$(J) GOTO 320
290  LET B$ = A$(I)
300  LET A$(I) = A$(J)
310  LET A$(J) = B$
320  NEXT J
330  NEXT I
340  REM  APRESENTANDO O RESULTADO
350  PRINT
360  PRINT "A ORDEM ALFABETICA E':"
370  PRINT
380  FOR I = 0 TO N - 1
390  PRINT A$(I)
400  NEXT I
500  END
```

O objetivo deste jogo é descobrir, através de dedução um número aleatório de 4 algarismos gerado pelo computador.

O computador lhe permitirá fazer até 18 tentativas antes de lhe revelar qual é o número e, a cada tentativa sua, irá lhe fornecer duas pistas, sobre as quais você deverá fazer sua dedução.

Estas duas pistas são:

MOSCA - diz quantos algarismos do número entrado por você coincidem em valor e posição com o número secreto.

Por exemplo:

número secreto 1532
você entrou 1672

Temos MOSCA 2

PERTO - diz quantos algarismos do número entrado por você coincidem em valor, mas estão em posição errada em relação ao número secreto.

Por exemplo:

número secreto 1532
você entrou 0121

Temos PERTO 2

Mais alguns exemplos:

número secreto 1321
você entrou 1111

MOSCA 2 PERTO 0

número secreto 1234
você entrou 1343

MOSCA 2 PERTO 2

As 18 tentativas que lhe são oferecidas são mais do que suficientes, você deverá se esforçar para deduzir qual é o número no menor número de tentativas possível.

```
###############
# DEDUZINDO #
###############

VOCE TEM 18 TENTATIVAS PARA
DESCOBRIR O NUMERO  SECRETO
QUAL O NUMERO  ?1234
MOSCA 2      PERTO 0
QUAL O NUMERO  ?5678
MOSCA 0      PERTO 0
QUAL O NUMERO  ?1290
MOSCA 1      PERTO 0
QUAL O NUMERO  ?3490
MOSCA 0      PERTO 1
QUAL O NUMERO  ?1133
MOSCA 2      PERTO 2
QUAL O NUMERO  ?1331

PARABENS. VOCE CONSEGUIU EM 6 TENTATIVAS

NOVO NUMERO ? (S OU N ) ?S

VOCE TEM 18 TENTATIVAS PARA
DESCOBRIR O NUMERO  SECRETO
QUAL O NUMERO  ?1234
MOSCA 0      PERTO 1
QUAL O NUMERO  ?5678
MOSCA 0      PERTO 1
QUAL O NUMERO  ?9012
MOSCA 0      PERTO 1
QUAL O NUMERO  ?2259
MOSCA 1      PERTO 0
QUAL O NUMERO  ?1122
MOSCA 0      PERTO 0
QUAL O NUMERO  ?5599
MOSCA 1      PERTO 1
QUAL O NUMERO  ?3599
MOSCA 2      PERTO 1
QUAL O NUMERO  ?4599
MOSCA 1      PERTO 1
QUAL O NUMERO  ?3559
MOSCA 2      PERTO 0
QUAL O NUMERO  ?3589
MOSCA 3      PERTO 0
QUAL O NUMERO  ?3569
MOSCA 2      PERTO 0
QUAL O NUMERO  ?3579
MOSCA 2      PERTO 0
QUAL O NUMERO  ?3559
MOSCA 2      PERTO 0
QUAL O NUMERO  ?3598
MOSCA 1      PERTO 2
QUAL O NUMERO  ?3689
MOSCA 3      PERTO 0
QUAL O NUMERO  ?3989

PARABENS. VOCE CONSEGUIU EM 16 TENTATIVAS

NOVO NUMERO ? (S OU N ) ?N
```



```
100  PRINT "###############"
110  PRINT "# DEDUZINDO #"
120  PRINT "###############"
130  PRINT
140  PRINT "VOCE TEM 18 TENTATIVAS PARA"
150  PRINT "DESCOBRIR O NUMERO  SECRETO"
160  REM  TRUQUE PARA OBTENCAO DO NUMERO
170  LET N =  INT (10000 *  RND (1)) + 10000
180  REM  INICIO DAS TENTATIVAS
190  FOR I = 1 TO 18
230  PRINT "QUAL O NUMERO  ";
240  INPUT A
250  REM  ADAPTANDO O NUMERO ENTRADO
260  LET A = A + 10000
270  REM  TRANSFORMACAO EM STRING
280  LET N$ =  STR$ (N)
290  LET A$ =  STR$ (A)
292  LET C = 0
294  LET P = 0
300  REM  POSICAO CORRETA
305  LET X = 2
310  IF  MID$ (A$,X,1) <  > MID$ (N$,X,1) GOTO 335
312  IF X =  LEN (A$) GOTO 322
315  LET A$ =  MID$ (A$,1,X - 1) +  MID$ (A$,X + 1,( LEN
     (A$) - X))
320  LET N$ =  MID$ (N$,1,X - 1) +  MID$ (N$,X + 1,( LEN
     (N$) - X))
321  GOTO 327
322  LET A$ =  MID$ (A$,1,X - 1)
323  LET N$ =  MID$ (N$,1,X - 1)
324  LET P = P + 1
325  IF P = 4 GOTO 530
326  GOTO 350
327  LET P = P + 1
328  IF P = 4 GOTO 530
330  GOTO 310
335  LET X = X + 1
340  IF X <  =  LEN (A$) GOTO 310
350  REM  POSICAO INCORRETA
352  FOR Y = 2 TO  LEN (N$)
354  FOR X = 2 TO  LEN (N$)
358  IF  MID$ (A$,Y,1) <  > MID$ (N$,X,1) GOTO 370
360  LET C = C + 1
362  LET N$ =  MID$ (N$,1,X - 1) + "X" +  MID$ (N$,X +
     1,( LEN (N$) - X))
364  GOTO 380
370  NEXT X
380  NEXT Y
400  REM   APRESENTACAO DO RESULTADO
410  PRINT "MOSCA ";P;"     PERTO ";C
440  NEXT I
450  REM  MENSAGENS FINAIS
460  PRINT
470  PRINT "VOCE NAO CONSEGUIU. O NUMERO E' "; RIGHT$
     ( STR$ (N),4)
480  PRINT
```

```
490  PRINT "NOVO NUMERO ? (S OU N ) ";
500  INPUT R$
510  IF R$ = "S" GOTO 130
520  GOTO 560
530  PRINT
540  PRINT "PARABENS. VOCE CONSEGUIU EM ";I;" TENTATIV
     AS"
550  GOTO 480
560  END
```

APÊNDICE 2

FUNÇÕES, INSTRUÇÕES E COMANDOS EM LINGUAGEM BASIC

APÊNDICE 2

FUNÇÕES, INSTRUÇÕES E COMANDOS EM LINGUAGEM BASIC

ABS (exp.num.)	100 PRINT ABS(-3.5) 200 LET X = ABS(X) 300 IF ABS(3-X)> 3 THEN	Fornece o valor absoluto do argumento.
AND	100 IF A$ = "N" AND A = 3 THEN	Ambas as condições ligadas pelo operador AND têm que ser verdadeiras, para que a condição total seja verdadeira.
ASC (exp.alf.)	100 PRINT ASC("BASIC") 200 PRINT ASC(A$)	Fornece o valor decimal do código ASCII do primeiro caractere do argumento.
ATN (exp.num.)	100 PRINT ATN(1.5) 200 PRINT ATN(X)	Fornece o arcotangente em radianos, do argumento.
AUTO num.1, num.2	AUTO 100,10	Este comando inicia uma numeração automática das linhas do programa que for entrado. O número 1 fornece o valor da primeira linha do programa e o número 2 fornece o incremento entre uma linha e a seguinte.
CALL (exp.num.)	100 CALL 3280 200 CALL X	Executa uma subrotina em linguagem de máquina, com endereço fornecido pelo valor da expressão numérica.
CHR$ (exp.num.)	100 PRINT CHR$ (65) 200 PRINT CHR$ (X) 300 LET A$ = CHR$(60+ I)	Fornece o caractere equivalente ao valor do argumento de acordo com a tabela ASCII.

CLEAR	100 CLEAR	Zera todas as variáveis numéricas e "limpa" todas as variáveis alfanuméricas do programa.
CLOAD	CLOAD	Recebe um programa de um gravador de fita acoplado ao computador e o envia para a memória. Equivalente a LOAD em algumas versões.
CLS	100 CLS	Limpa a tela do vídeo e coloca o cursor no início da primeira linha. Equivalente a HOME em algumas versões.
CONT	CONT	Reinicia a execução de um programa interrompido por um STOP ou control C, a partir do ponto onde ocorreu a interrupção.
COS (exp.num.)	100 LET A = COS (1.5) 200 PRINT COS (X)	Fornece o cosseno do ângulo em radianos dado pelo argumento.
CSAVE	CSAVE	Envia o programa atualmente na memória para um gravador de fita acoplado ao computador. Equivalente à SAVE em algumas versões.
CURSOR(exp.num.1, exp.num.2)	100 CURSOR (10,20) 200 CURSOR (X,Y)	Coloca o cursor na linha de número dado pela exp.num.1 na coluna dada pela exp.num.2.
DATA lista de dados	100 DATA 1,2,3,5 200 DATA MEU, SEU 300 DATA "JAN", 31, "FEV", 28, "MAR", 31	Define a lista de dados a ser lida pela instrução READ. Esses dados devem ser valores numéricos e/ou "STRINGS".

DEF FN nome (var.num.) = exp.num.	100 DEF FN A(X)=X+3 200 DEF FN B(Y)=A+ATN (Y)/COS(Y)	Define uma função para uso no programa. O nome da função pode ser qualquer nome válido para variáveis. A variável desta função é "postiça" (ver texto p.167).
DEL num.1, num.2	DEL 100 DEL 100,1000	Retira a linha do programa de número num.1 ou todas as linhas de num.1 a num.2. DELETE em algumas versões.
DELETE num.1, num.2	DELETE 100 DELETE 100,1000	Idêntico a DEL, sendo usado no seu lugar em algumas versões.
DIM	100 DIM A(20) 200 DIM X$ (25) 300 DIM A (10),X$(25)	Reserva espaços na memória para as variáveis numéricas e alfanúmericas, indexadas.
END	100 END	Termina a execução do programa e leva o computador para o modo de Comando e Edição.
EXP (exp.num.)	100 LET T = EXP(20) 200 PRINT EXP (X)	Fornece o valor de "e" elevado ao argumento (e = 2.718289).
FLASH	100 FLASH	Faz com que todas as mensagens enviadas pelo computador fiquem piscando no vídeo.

FOR var.num. = exp.num. 1 *TO* exp.num. 2 *STEP* exp.num.2	100 FOR I = 0 TO 100 200 FOR A =X+3 TO Y-4 300 FOR C = 1 TO 1000 STEP 10	Inicia um "loop" controlado pelo valor da variável. Seu valor inicial é definido pela exp. num.1 e o final pela exp. num.2, sendo incrementado a cada ciclo de "loop", unitariamente ou, em degraus, quando STEP e exp.num. 3 forem utilizadas. O "loop" termina quando o valor da variável ultrapassar ao valor dado pela exp.num.2.
GOSUB exp.num.	100 GOSUB 1200 200 GOSUB X + 3000	Desvia a execução do programa para a subrotina iniciada na linha de número igual ao valor da exp.num. Em algumas versões a exp.num. deve obrigatoriamente ser uma constante.
GOTO exp.num.	100 GOTO 500 200 GOTO Y + 250	Desvia a execução do programa para a linha do programa de número igual ao valor da exp.num. Em algumas versões, a exp.num. deve obrigatoriamente ser uma constante.
HOME	100 HOME	Limpa a tela do vídeo e coloca o cursor na primeira linha.
IF condição *THEN* ação 1 *ELSE* ação 2	100 IF A > 10 OR B$ = "H" THEN PRINT "QUASE" ELSE PRINT "MUITO LONGE" 200 IF A$ = "S" GOTO 300	Se a condição for válida, então a ação 1 será executada, caso contrário será a ação 2. Em algumas versões, a palavra ELSE e a ação 2 não podem ser usadas e, neste caso, quando a condição não for válida, a próxima linha do programa será executada.

INPUT (var.)	100 INPUT A 200 INPUT B$ 300 INPUT A, B$ 400 INPUT "VALOR ?";A 500 INPUT "NOME"; B$	Permite a entrada de dados via teclado, os quais são associados às variáveis presentes na instrução. Algumas versões permitem a inclusão de uma "STRING" após INPUT a qual é enviada para o vídeo.
INT (exp.num.)	100 LET X = INT (3.1) 200 PRINT INT(X*3.517)	Despreza a parte decimal do número, criando assim um número inteiro.
INVERSE	100 INVERSE	Inverte a apresentação dos caracteres enviados pelo computador, no vídeo, para letra preta em fundo branco.
LEFTH$ (exp.alf., exp.num.)	100 LET T$ = LEFTH$ ("ALFA",3) 200 PRINT LEFTH$ (X$,A)	Cria uma "STRING" formada pelos caracteres da esquerda da "STRING" constante da exp. alf. O número de caracteres retirados para formação da nova "STRING" é dado pelo valor da exp.num.
LEN (exp.alf.)	100 PRINT LEN (A$) 200 LET A=LEN("ABCD")	Fornece o comprimento da exp.alf. em número de caracteres.
LET var.= exp.	100 LET A$ = "ABCD" 200 LET A = 23.34	Associa um valor a uma variável ou uma "STRING" a uma variável alfanumérica. A palavra LET é opcional na maioria das versões.

LIST	LIST LIST 10 LIST 10,100	Lista o programa atualmente na memória do computador. Quando a instrução contiver um número, somente a linha do programa com esse número será listada. Se forem dois números separados por vírgula, o trecho do programa definido por esses dois números de linha, será listado.
LOAD nome	LOAD LOAD SOMANDO	Recebe um programa de um gravador de fita acoplado ao computador e o envia para a memória. Neste caso, equivalente a CLOAD de algumas versões. Se um nome seguir a LOAD, o programa será buscado de uma unidade de disco acoplada ao computador.
MID$ (exp.alf., exp.num.1, exp.num.2)	100 LET X$ = MID$ (A$,3,2) 200 PRINT MID$ ("ALFA",1,1) 300 PRINT MID$ (C$,X, X + I)	Cria uma "STRING" formada com os caracteres da "STRING" constante da exp.alf., a partir do caractere dado pela exp.num. 1, com comprimento dado pela exp.num.2.
exp.num.1 *MOD* exp.num.2	100 LET X = A MOD B 200 PRINT X MOD 2	Fornece o "resto" da divisão da exp.num.1 pela exp.num.2.
NEW	NEW	Apaga o programa que está atualmente na memória e limpa todas as variáveis. Deve ser usado sempre que se for iniciar a entrada de um novo programa.

NEXT var.num.	100 NEXT I 200 NEXT J 300 NEXT I,J 400 NEXT	Define a instrução final do "loop" iniciado pela instrução FOR ... TO. Algumas versões permitem a colocação de duas ou mais variáveis numa mesma instrução desde que sejam separadas por vírgula e estejam na ordem correta. Quando apenas um "loop" estiver sendo realizado a omissão do nome da variável é facultativo em algumas versões.
NORMAL	100 NORMAL	Reestabelece a condição normal do envio de mensagens do computador para o vídeo, ou seja, caracteres brancos sobre o fundo preto. Veja FLASH e INVERSE.
NOTRACE	NOTRACE	Interrompe a condição TRACE. Em algumas versões é denominado de TROFF.
ON exp.num. *GOSUB* num., num.	100 ON X GOSUB 500, 600, 700	Desvia a execução do programa para a subrotina cujo número de linha corresponde ao número de ordem dado pelo valor da exp.num. Sendo esse valor 1, a primeira subrotina será executada; 2, a segunda; e assim por diante. Se o valor for zero ou se ultrapassar o número de subrotinas especificadas, a linha seguinte do programa será executada.
ON exp.num. *GOTO* num., num.	100 ON X GOTO 500,600,700	Operação semelhante a ON GOSUB, sendo executado um GOTO ao invés de um GOSUB.

OR	100 IF A$ ="S" OR V = 100 THEN	Se qualquer uma das condições ligadas pelo operador OR for verdadeira, a condição total será verdadeira.
PEEK (exp.num.)	100 LET A = PEEK (X) 200 PRINT PEEK(10240)	Fornece o valor decimal do conteúdo da posição da memória dada pela exp.num.
POKE exp.num.1, exp.num.2	100 POKE 10340,Ø 200 POKE X, Y	Carrega a posição da memória dada pela exp.num.1, com o valor da exp.num.2.
POP	100 POP	Limpa o endereço atual de retorno de subrotina, substituindo-o pelo anterior. Equivale a um RETURN, só que o retorno não é executado, transformando um GOSUB num GOTO.
POS (num.)	100 LET X = POS(0) 200 PRINT POS (0)	Fornece o valor da coluna onde o cursor se encontra. O valor fornecido para primeira coluna é zero. Como o valor fornecido independe do valor do argumento, este pode ser qualquer um, sendo normalmente utilizado o valor zero.
PRINT exp.	100 PRINT "ME" 200 PRINT 3 300 PRINT A$ 400 PRINT A 500 PRINT A, A$ 600 PRINT A; "MEU" 700 PRINT A + B 800 PRINT 3 + 4 900 PRINT A$ + B$ 1000 PRINT	Envia para o vídeo o dado fornecido pela exp., podendo ser um valor ou uma "STRING". Quando o complemento não for seguido por ";" ou "," a próxima mensagem será enviada para a linha seguinte. Um ponto e vírgula concatena o envio de mensagens e a vírgula organiza em colunas. Quando utilizada sem complemento envia uma linha em branco.

READ var.,var.	100 READ A 200 READ A$ 300 READ A, A$, B	Associa sequencialmente os dados constantes da tabela definida pela instrução DATA à variável ou variáveis presentes. A leitura de dados além da tabela causa interrupção do programa por erro.
REM	100 REM COMENTÁRIO	Permite a inclusão de comentários em um programa. O computador ignora totalmente esta linha do programa.
RETURN	100 RETURN	Última instrução de uma subrotina. Desvia o fluxo do programa para a linha seguinte a instrução GOSUB que iniciou a execução da subrotina. O seu uso sem que tenha havido uma GOSUB causa a interrupção do programa por erro.
RIGHT$ (exp.alf. exp.num.)	100 LET X$ = RIGHT$ ("ALFA",3) 200 PRINT RIGHT$ (X$; A)	Cria uma "STRING" formada pelos caracteres da direita da "STRING" constante da exp.alf. O número dos caracteres retirados para formação da nova "STRING" é dado pelo valor da exp.num.
RND (num.)	100 LET X = RND(-1) 200 PRINT RND (1) 300 IF X < RND (0) THEN	Fornece um número aleatório entre 0 e 1. Se o argumento for negativo uma mesma seqüência de número aleatório é fornecida. O argumento sendo positivo, uma nova seqüência é iniciada. O argumento zero fornece o último valor dado.

RUN num.	RUN RUN 100	Inicia a execução do programa arquivado na memória do computador. Quando o valor de uma linha é fornecido o programa é executado a partir desta linha.
SAVE nome	SAVE SAVE SOMANDO	Envia o programa atualmente na memória para um gravador de fita acoplado ao computador. Neste caso, equivalente a CSAVE de algumas versões. Se um nome seguir a SAVE, o programa será enviado a uma unidade de disco acoplado ao computador.
SGN (exp.num.)	100 PRINT SGN(X) 200 IF SGN(Y) < 0 THEN 300 LET A = SGN (-3)	Fornece o valor -1, se o argumento for negativo, zero, se for zero; e 1, se for positivo.
SIN (exp.num.)	100 PRINT SIN (.05) 200 LETX = SIN (X)	Fornece o valor do seno do argumento. O argumento é considerado como um valor de ângulo em radianos.
SPACE $ (exp.num.)	100 PRINT SPACE $ (X) 200 PRINT SPACE $(10)	Envia para o vídeo, espaços equivalentes ao valor do argumento. Equivalente a SPC em algumas versões.
SPC (exp.num.)	100 PRINT SPC (10) 200 PRINT SPC (X)	Envia para o vídeo espaços equivalentes ao valor do argumento. Equivalente a SPACE$ em algumas versões.

SQR (exp.num.)	100 PRINT SQR (10) 200 LETX = SQR (20) 300 IF X < SQR (Y) THEN	Fornece o valor da raiz quadrada do argumento que deve ser positivo.
STOP	100 STOP	Interrompe a execução do programa e leva o computador para o modo Comando e Edição. O número da linha na qual ocorreu a interrupção é enviada para o vídeo.
STR$ (exp.num.)	100 LET X$ = STR$ (-12,3) 200 LET X1 = STR$ (X)	Converte o valor da exp.num. em uma "STRING".
TAB (exp.num.)	100 PRINT TAB (8);"A" 200 PRINT TAB (I);I	Inicia a escrever no vídeo na coluna correspondente ao valor dado pela exp.num. Deve, obrigatoriamente, ser usado em uma instrução de PRINT.
TAN (exp.num.)	100 PRINT TAN (1.5) 200 LET Y = TAN (X)	Fornece o valor da tangente do argumento. O argumento é considerado como um valor de ângulo em radianos.
TRACE	TRACE	Envia para o vídeo o número das linhas do programa que está sendo executado, automaticamente. Este tipo de operação visa a simplificação e análise de programas para a descoberta de "BUGS". Equivalente a TRON de algumas versões.

TROFF	TROFF	Equivalente a NOTRACE.
TRON	TRON	Equivalente a TRACE.
USR (exp.num.)	100 USR (8) 200 USR (X)	Executa uma subrotina em linguagem de máquina, colocada em um endereço específico da memória. O valor do argumento é utilizado como dado para essa subrotina.
VAL (exp.num.)	100 PRINT VAL ("100") 200 PRINT VAL (X$) 300 LET X = VAL (X$)	Fornece o valor numérico de uma "STRING" até o primeiro caractere não numérico.
VTAB exp.num.	100 VTAB 100 200 VTAB X	Inicia a escrever no vídeo na linha correspondente ao valor dado pela exp.num.
XOR	100 IF A$ = "N" XOR B$ = "N" THEN	Operador ou-exclusivo. A condição total só é válida quando uma condição for falsa e a outra verdadeira, e vice-versa. Se as condições forem ambas falsas ou verdadeiras simultaneamente, a condição total será falsa.

CONVENÇÕES UTILIZADAS

exp.num. → pode ser um número, uma variável numérica ou uma expressão numérica.

var.num. → deve ser exclusivamente uma variável numérica.

num. → deve ser exclusivamente um número.

exp.alf. → pode ser uma "STRING", uma variável alfanumérica, ou uma expressão alfanumérica.

exp. → pode ser uma expressão numérica ou alfanumérica.

var. → pode ser uma variável número ou alfanumérica.

APÊNDICE 3

TABELA ASCII

APÊNDICE 3

TABELA ASCII

DECIMAL	CARACTERE	DECIMAL	CARACTERE	DECIMAL	CARACTERE
000	NUL	043	+	086	V
001	SCH	044	,	087	W
002	STX	045	-	088	X
003	ETX	046	.	089	Y
004	EOT	047	/	090	Z
005	ENQ	048	0	091	[
006	ACK	049	1	092	\
007	BEL	050	2	093	]
008	BS	051	3	094	^
009	HT	052	4	095	_
010	LF	053	5	096	`
011	VT	054	6	097	a
012	FF	055	7	098	b
013	CR	056	8	099	c
014	SO	057	9	100	d
015	SI	058	:	101	e
016	DLE	059	;	102	f
017	DC1	060	<	103	g
018	DC2	061	=	104	h
019	DC3	062	>	105	i
020	DC4	063	?	106	j
021	NARK	064	@	107	k
022	SYN	065	A	108	l
023	ETB	066	B	109	m
024	CAN	067	C	110	n
025	EM	068	D	111	o
026	CONTROL	069	E	112	p
027	ESCAPE	070	F	113	q
028	FS	071	G	114	r
029	GS	072	H	115	s
030	RS	073	I	116	t
031	US	074	J	117	u
032	SPACE	075	K	118	v
033	!	076	L	119	w
034	"	077	M	120	x
035	#	078	N	121	y
036	$	079	O	122	z
037	%	080	P	123	{
038	&	081	Q	124	|
039	'	082	R	125	}
040	(	083	S	126	~
041	)	084	T	127	DEL
042	*	085	U		

BEL - CAMPAINHA

EOT - FINAL DO TEXTO

LF - PULA LINHA

CR - VOLTA O CARRO

APÊNDICE 4

RESOLUÇÃO DOS EXERCÍCIOS PROPOSTOS

APÊNDICE 4

RESOLUÇAO DOS EXERCICIOS PROPOSTOS

Este apêndice contém as respostas aos exercícios propostos no final de cada capítulo. Procure somente recorrer a ele para comprovar sua resposta, e não para saber a solução do problema.

EXERCÍCIOS DO CAPÍTULO 1

EXERCÍCIO 1.1

a.) Não. O programa será listado na ordem seqüencial das linhas.

b.) O programa será executado na ordem seqüencial das linhas, ou seja, 10, 20, 30, 40.

```
10  PRINT "UM"
20  PRINT "DOIS"
30  PRINT "FEIJAO COM ARROZ"
40  END
```

c.) Rodando o programa, obteremos:

```
UM
DOIS
FEIJAO COM ARROZ
```

EXERCÍCIO 1.2

Envie a nova linha

9Ø END

e suprima a linha 40, enviando somente o número 40.

4Ø

Listando o programa, temos:

```
10  PRINT "UM"
20  PRINT "DOIS"
30  PRINT "FEIJAO COM ARROZ"
90  END
```

EXERCÍCIO 1.3

Essas novas linhas serão acrescentadas ao programa e, listando-o, teremos:

```
10  PRINT "UM"
20  PRINT "DOIS"
30  PRINT "FEIJAO COM ARROZ"
40  PRINT "TRES"
50  PRINT "QUATRO"
60  PRINT "FAZER O PRATO"
90  END
```

Rodando temos:

```
UM
DOIS
FEIJAO COM ARROZ
TRES
QUATRO
FAZER O PRATO
```

EXERCÍCIO 1.4

Como queremos entrar um novo programa, antes de mais nada, devemos limpar a memória do computador com o comando *NEW* e, então, entrarmos com o programa, e após, rodá-lo.

Esta seqüência, então, será:

```
100  PRINT "CINCO"
200  PRINT "SEIS"
300  PRINT "OUTRA VEZ"
400  END
```

Rodando temos:

```
CINCO
SEIS
OUTRA VEZ
```


EXERCÍCIOS DO CAPÍTULO 2

EXERCÍCIO 2.1

```
10  PRINT "         D"
20  PRINT "        O  E"
30  PRINT "       D    S"
40  PRINT "      N      C"
50  PRINT "     I        E"
60  PRINT "    B          N"
70  PRINT "   U            D"
80  PRINT "  S              O"
90  END
```

Rodando temos:

```
     D
    O  E
   D    S
  N      C
 I        E
B          N
 U          D
  S          O
```

EXERCÍCIO 2.2

```
100  PRINT "ANTONIMOS"
200  PRINT
300  PRINT "DIA","NOITE"
400  PRINT "CARO","BARATO"
500  PRINT "FRIO","QUENTE"
600  PRINT "LONGE","PERTO"
700  PRINT "FACIL","DIFICIL"
800  END
```

Rodando temos:

```
ANTONIMOS

DIA             NOITE
CARO            BARATO
FRIO            QUENTE
LONGE           PERTO
FACIL           DIFICIL
```

EXERCÍCIO 2.3

```
100  PRINT "NUMERO DE SEGUNDOS EM:"
110  PRINT
120  PRINT "1 MINUTO",60 * 1
130  PRINT "1 HORA",60 * 60 * 1
140  PRINT "1 DIA",60 * 60 * 24 * 1
150  PRINT "1 ANO",60 * 60 * 24 * 365 * 1
160  END
```

Rodando temos:

```
NUMERO DE SEGUNDOS EM:

1 MINUTO        60
1 HORA          3600
1 DIA           86400
1 ANO           31536000
```

EXERCÍCIO 2.4

```
100  PRINT "CALCULO DE AREA"
105  PRINT
110  PRINT "UM QUADRADO DE 52 METROS DE LADO"
115  PRINT "TEM ";52 * 52;" METROS QUADRADOS DE AREA"
120  PRINT "UM CIRCULO COM 133 CENTIMETROS DE"
125  PRINT "DIAMETRO TEM ";((133 ^ 2) * 3.14) / 4;" CENTIMETROS"
130  PRINT "QUADRADOS DE AREA"
140  END
```

Rodando temos:

```
CALCULO DE AREA

UM QUADRADO DE 52 METROS DE LADO
TEM 2704 METROS QUADRADOS DE AREA
UM CIRCULO COM 133 CENTIMETROS DE
DIAMETRO TEM 13885.865 CENTIMETROS
QUADRADOS DE AREA
```


EXERCÍCIOS DO CAPÍTULO 3

EXERCÍCIOS 3.1

```
100  PRINT "QUAL E' O SEU NOME?"
110  PRINT
120  INPUT N$
130  PRINT
140  PRINT "QUANDO VOCE NASCEU?"
150  PRINT
160  INPUT A
170  PRINT
180  PRINT "MEUS PARABENS, ";N$;". ESTE ANO VOCE COMPL
     ETA ";1982 - A;" ANOS"
190  END
```

Rodando temos:

```
QUAL E' O SEU NOME?

?CARLOS   (NOME ENTRADO VIA TECLADO)

QUANDO VOCE NASCEU?

?1952     ( ANO  ENTRADO VIA TECLADO)

MEUS PARABENS, CARLOS.  ESTE ANO VOCE COMPLETA 30
ANOS
```

EXERCÍCIO 3.2

```
100  PRINT "ENTRE COM UM NUMERO?"
110  PRINT
120  INPUT A
130  PRINT
140  PRINT "O DOBRO DE ";A;" MAIS 16, MULTIPLICADO"
150  PRINT "POR 18 E DIVIDIDO POR 6, RESULTA EM "
160  PRINT ((A * 2 + 16) * 18) / 6;". DUVIDA?"
170  END
```

Rodando temos:

```
ENTRE COM UM NUMERO?

?36

O DOBRO DE 36 MAIS 16, MULTIPLICADO
POR 18 E DIVIDIDO POR 6, RESULTA EM
264. DUVIDA?
```

EXERCÍCIO 3.3

```
100  PRINT "QUAL SUA IDADE?"
110  PRINT
120  INPUT A
130  PRINT
140  PRINT "PUXA! VOCE TEM MAIS QUE ";A * 365 * 24 * 6
     0;" MINUTOS DE IDADE"
```

Rodando temos:

```
QUAL SUA IDADE?

?21

PUXA! VOCE TEM MAIS QUE 11037600 MINUTOS DE IDADE
```

EXERCÍCIOS DO CAPÍTULO 4

EXERCÍCIO 4.1

A seqüência de execução é 10, 100, 40, 20, 30, 50, 90, 70, 80, 60, 110.

EXERCÍCIO 4.2

Rodando temos:

```
MENSAGEM SURPRESA
ENTRE COM 2 NUMEROS SEPARADOS POR
VIRGULA ENTRE 1 E 3, PODENDO SEREM
REPETIDOS
?1,2
VOCE ME AMA?
```


EXERCÍCIO 4.3

```
10   PRINT "MENSAGEM SURPRESA"
20   PRINT "ENTRE COM 2 NUMEROS SEPARADOS POR"
22   PRINT "VIRGULA ENTRE 1 E 3, PODENDO SEREM"
23   PRINT "REPETIDOS"
25   LET M$ = "OS NUMEROS DEVEM ESTAR ENTRE 1 E 3"
30   INPUT A,B
40   ON A GOSUB 100,200,300
50   PRINT M$
70   GOTO 1200
100   ON B GOSUB 110,120,130
105   RETURN
110   LET M$ = "EU TE AMO"
115   RETURN
120   LET M$ = "VOCE ME AMA?"
125   RETURN
130   LET M$ = "NOS NOS AMAMOS"
135   RETURN
200   ON B GOSUB 210,220,230
205   RETURN
210   LET M$ = "EU TE ODEIO"
215   RETURN
220   LET M$ = "VOCE ME ODEIA?"
225   RETURN
230   LET M$ = "NOS NOS ODIAMOS"
235   RETURN
300   ON B GOSUB 310,320,330
305   RETURN
310   LET M$ = "EU QUERO CASAR-ME COM VOCE"
315   RETURN
320   LET M$ = "VOCE QUER CASAR-SE COMIGO?"
325   RETURN
330   LET M$ = "NOS QUEREMOS NOS CASAR"
335   RETURN
1200   END
```

Rodando temos:

```
MENSAGEM SURPRESA
ENTRE COM 2 NUMEROS SEPARADOS POR
VIRGULA ENTRE 1 E 3, PODENDO SEREM
REPETIDOS
?1,2
VOCE ME AMA?
```

EXERCÍCIOS DO CAPÍTULO 5

EXERCÍCIO 5.1

a.) IF A < 10 GOTO 100

b.) IF A$ = "SIM" THEN B = 100

c.) IF R1$ = "PAGO" AND R2$ = "PASSO" THEN PRINT "FIM DO JOGO".

EXERCÍCIO 5.2

a.) Para qualquer valor de T

b.) Sempre que J$ não for nem "NÃO" e nem "SIM".

EXERCÍCIO 5.3

a.) A condição sempre será válida.

b.) A condição nunca será válida.

EXERCÍCIOS DO CAPÍTULO 6

EXERCÍCIO 6.1

```
100  PRINT "FATORIAL DE UM NUMERO"
110  PRINT
120  PRINT "QUAL NUMERO?"
130  INPUT A
140  F = 1
150  FOR I = 1 TO A
160  F = F * I
170  NEXT I
180  PRINT
190  PRINT "O FATORIAL DE ";A;" E' ";F
200  END
```

Rodando temos:

```
FATORIAL DE UM NUMERO

QUAL NUMERO?
?10

O FATORIAL DE 10 E' 3628800
```


EXERCÍCIO 6.2

```
100  FOR A = 1 TO 5
200  PRINT A * 2
300  NEXT A
400  END
```

Rodando, temos:

```
2
4
6
8
10
```

EXERCÍCIO 6.3

```
100  PRINT "TABUADA"
120  PRINT
130  PRINT "1- MULTIPLICACAO"
140  PRINT "2- SOMA"
150  PRINT "QUAL? 1 OU 2"
160  INPUT Q
170  PRINT
180  PRINT "DE QUAL NUMERO?"
190  INPUT N
200  PRINT
210  ON Q GOTO 220,300
220  PRINT "TABUADA DE MULTIPLICACAO DO NUMERO ";N
230  FOR I = 1 TO 10
240  PRINT N;" X ";I" = ";N * I
250  NEXT I
260  GOTO 400
300  PRINT "TABUADA DE SOMA DO NUMERO ";N
310  FOR I = 1 TO 10
320  PRINT N;" + ";I" = ";N + I
330  NEXT I
340  GOTO 400
400  END
```

Rodando, temos:

```
TABUADA

1- MULTIPLICACAO
2- SOMA
QUAL? 1 OU 2
?2

DE QUAL NUMERO?
?7

TABUADA DE SOMA DO NUMERO 7
7 + 1 = 8
7 + 2 = 9
7 + 3 = 10
7 + 4 = 11
7 + 5 = 12
7 + 6 = 13
7 + 7 = 14
7 + 8 = 15
7 + 9 = 16
7 + 10 = 17
```

EXERCÍCIO 6.4

```
100  PRINT "X","Y"
200  FOR I =  - 10 TO 10
300  PRINT I,3 * I ^ 2 - 2 * I -
     6
400  NEXT I
500  END
```

Rodando temos:

X	Y
-10	314
-9	255
-8	202
-7	155
-6	114
-5	79
-4	50
-3	27
-2	10
-1	-1
0	-6
1	-5
2	2
3	15
4	34
5	59
6	90
7	127
8	170
9	219
10	274

EXERCÍCIOS DO CAPÍTULO 7

EXERCÍCIO 7.1

A variável está dimensionada incorretamente.

```
10  DIM A$(11)
20  FOR I = 1 TO 10
30  INPUT A$(I)
40  NEXT I
50  END
```

EXERCÍCIO 7.2

```
10  DIM A(10)
20  FOR I = 10 TO 20
30  INPUT A(I)
40  PRINT B + A(I)
50  NEXT I
60  END
```

Rodando, temos:

```
?10
10
?20

?BAD SUBSCRIPT ERROR IN 30
```

EXERCÍCIO 7.3

```
100  DIM N$(4),V$(13)
110  LET N$(0) = "COPAS"
111  LET N$(1) = "PAUS"
112  LET N$(2) = "OUROS"
113  LET N$(3) = "ESPADA"
114  LET V$(0) = "AS"
115  LET V$(1) = "2"
116  LET V$(2) = "3"
117  LET V$(3) = "4"
118  LET V$(4) = "5"
119  LET V$(5) = "6"
120  LET V$(6) = "7"
121  LET V$(7) = "8"
122  LET V$(8) = "9"
123  LET V$(9) = "10"
124  LET V$(10) = "VALETE"
125  LET V$(11) = "DAMA"
126  LET V$(12) = "REI"
130  PRINT "EIS UMA CARTA DE UM BARALHO"
140  LET X =  INT (4 *  RND (1))
150  LET Y =  INT (13 *  RND (1))
155  PRINT
160  PRINT V$(Y);" DE ";N$(X)
165  PRINT
170  PRINT "NOVA CARTA?"
180  INPUT R$
190  IF R$ = "S" GOTO 140
200  END
```

Rodando temos:

```
EIS UMA CARTA DE UM BARALHO

6 DE COPAS

NOVA CARTA?
?S

4 DE PAUS

NOVA CARTA?
?S

10 DE ESPADA

NOVA CARTA?
?N
```

EXERCÍCIOS DO CAPÍTULO 8

EXERCÍCIOS 8.1

```
100  PRINT "FERIADOS OFICIAIS DE 1984"
120  PRINT
130  READ A$
140  IF A$ = "FIM" GOTO 500
150  PRINT A$
160  GOTO 130
170  DATA  1 DE JANEIRO - ANO NOVO,20 DE ABRIL - PAIXA
     O,21 DE ABRIL - TIRADENTES,1 DE MAIO - DIA DO TRA
     BALHO,21 JUNHO - CORPUS CHRISTI,7 DE SETEMBRO - I
     NDEPENDENCIA DO BRASIL,12 DE OUTUBRO - NOSSA SRA.
      APARECIDA
180  DATA  2 DE NOVEMBRO - FINADOS,15 NOVEMBRO - PROCL
     . DA REPUBLICA,25 DE DEZEMBRO - NATAL,FIM
500  END
```

Rodando, temos:

```
FERIADOS OFICIAIS DE 1984

1 DE JANEIRO - ANO NOVO
20 DE ABRIL - PAIXAO
21 DE ABRIL - TIRADENTES
1 DE MAIO - DIA DO TRABALHO
21 JUNHO - CORPUS CHRISTI
7 DE SETEMBRO - INDEPENDENCIA DO BRASIL
12 DE OUTUBRO - NOSSA SRA. APARECIDA
2 DE NOVEMBRO - FINADOS
15 NOVEMBRO - PROCL. DA REPUBLICA
25 DE DEZEMBRO - NATAL
```


EXERCÍCIO 8.2

```
100  PRINT "CUSTO MEDIO"
110  PRINT
115  FOR I = 1 TO 3
120  RESTORE
130  PRINT "ENTRE COM  O CUSTO"
140  PRINT "DO SUPERMERCADO ",I
150  FOR J = 0 TO 4
160  READ A$
170  PRINT " CUSTO DE ";A$
180  INPUT A
190  LET S(I) = S(I) + A
200  NEXT J
205  NEXT I
210  PRINT
215  PRINT "CUSTO MEDIO"
220  FOR I = 1 TO 3
225  PRINT
230  PRINT "SUPERMERCADO ";I;" ";S(I) / 5
235  NEXT I
240  GOTO 300
250  DATA  QUILO DE ARROZ,QUILO DE BATATA,QUILO DE FEI
     JAO,DUZIA DE OVOS,LITRO DE LEITE
300  END
```

EXERCÍCIO 8.3

a.)

```
90  REM  EXERCICIO 8.1/A
100  PRINT "FERIADOS OFICIAIS DE 1984"
120  PRINT
125  REM  LEITURA DA TABELA
130  READ A$
135  REM  VERIFICACAO DO FIM DA TABELA
140  IF A$ = "FIM" GOTO 500
145  REM  APRESENTACAO DOS DADOS
150  PRINT A$
160  GOTO 130
165  REM  TABELA DE DADOS
170  DATA  1 DE JANEIRO - ANO NOVO,20 DE ABRIL - PAIXA
     O,21 DE ABRIL - TIRADENTES,1 DE MAIO - DIA DO TRA
     BALHO,21 JUNHO - CORPUS CHRISTI,7 DE SETEMBRO - I
     NDEPENDENCIA DO BRASIL,12 DE OUTUBRO - NOSSA SRA.
      APARECIDA
175  REM  CONTINUACAO DA TABELA
180  DATA  2 DE NOVEMBRO - FINADOS,15 NOVEMBRO - PROCL
     . DA REPUBLICA,25 DE DEZEMBRO - NATAL,FIM
500  END
```

Rodando temos:

```
FERIADOS OFICIAIS DE 1984

1 DE JANEIRO - ANO NOVO
20 DE ABRIL - PAIXAO
21 DE ABRIL - TIRADENTES
1 DE MAIO - DIA DO TRABALHO
21 JUNHO - CORPUS CHRISTI
7 DE SETEMBRO - INDEPENDENCIA DO BRASIL
12 DE OUTUBRO - NOSSA SRA. APARECIDA
2 DE NOVEMBRO - FINADOS
15 NOVEMBRO - PROCL. DA REPUBLICA
25 DE DEZEMBRO - NATAL
```

EXERCÍCIO 8.3

b.)

```
90  REM   EXERCICIO 8.2/A
100  PRINT "CUSTO MEDIO"
110  PRINT
111  REM  ENTRANDO DADOS
115  FOR I = 1 TO 3
119  REM  VOLTANDO AO INICIO DA TABELA
120  RESTORE
130  PRINT "ENTRE COM  O CUSTO"
140  PRINT "DO SUPERMERCADO ",I
145  REM  VALORES DAS MERCADORIAS
150  FOR J = 0 TO 4
160  READ A$
165  REM  CALCULO DO CUSTO MEDIO
170  PRINT " CUSTO DE ";A$
180  INPUT A
190  LET S(I) = S(I) + A
200  NEXT J
205  NEXT I
209  REM  APRESENTACAO DOS RESULTADOS
210  PRINT
215  PRINT "CUSTO MEDIO"
220  FOR I = 1 TO 3
225  PRINT
230  PRINT "SUPERMERCADO ";I;" ";S(I) / 5
235  NEXT I
240  GOTO 300
249  REM  LISTA DAS MERCADORIAS
250  DATA  QUILO DE ARROZ,QUILO DE BATATA,QUILO DE FEI
     JAO,DUZIA DE OVOS,LITRO DE LEITE
300  END
```


EXERCÍCIOS DO CAPÍTULO 9

EXERCÍCIO 9.1

```
100  PRINT "CONVERSAO DE NOTACAO"
110  PRINT
120  PRINT "ENTRE COM UM NUMERO"
130  INPUT N
135  IF N =  INT (N) GOTO 300
140  LET N$ =  STR$ (N)
150  FOR I = 1 TO  LEN (N$)
160  IF  MID$ (N$,I,1) = "." GOTO 175
170  NEXT I
175  LET F$ =  RIGHT$ (N$,( LEN (N$) - I))
180  IF I = 1 GOTO 250
190  LET M$ =  MID$ (N$,1,(I - 1))
200  PRINT M$;",";F$
210  GOTO 500
250  PRINT "0,";F$
260  GOTO 500
300  PRINT N
500  END
```

Rodando, temos:

```
CONVERSAO DE NOTACAO

ENTRE COM UM NUMERO
?123.4567
123,4567
```

EXERCÍCIO 9.2

```
90  DIM X$(40)
100  PRINT "INVERTENDO"
110  PRINT
120  PRINT "ENTRE COM UMA PALAVRA"
130  PRINT "QUE  IREI  ESCREVE-LA"
140  PRINT "DE TRAS PARA A FRENTE"
150  PRINT
160  INPUT A$
170  FOR I = 1 TO  LEN (A$)
180  LET X$(I - 1) =  MID$ (A$,I,1)
190  NEXT I
200  FOR I =  LEN (A$) - 1 TO 0 STEP  - 1
210  PRINT X$(I);
220  NEXT I
225  PRINT
230  PRINT "NOVA PALAVRA (S OU N) "
240  INPUT R$
250  IF R$ = "S" GOTO 150
300  END
```

Rodando, temos:

```
INVERTENDO

ENTRE COM UMA PALAVRA
QUE  IREI  ESCREVE-LA
DE TRAS PARA A FRENTE

?LINGUAGEM BASIC
CISAB MEGAUGNIL
NOVA PALAVRA (S OU N)
?S

?CAFE COM LEITE
ETIEL MOC EFAC
NOVA PALAVRA (S OU N)
?S

?BASIC PARA NEOFITOS
SOTIFOEN ARAP CISAB
NOVA PALAVRA (S OU N)
?S

?COFFEA ARABICA
ACIBARA AEFFOC
NOVA PALAVRA (S OU N)
?S

?COMO VAI
IAV OMOC
NOVA PALAVRA (S OU N)
?N
```

EXERCÍCIO 9.3

```
100  PRINT "CODIFICANDO"
110  PRINT
120  PRINT "ENTRE O NUMERO A"
130  PRINT "SER   CODIFICADO"
140  PRINT
150  INPUT N
155  LET N =  INT (N)
160  LET N$ =  STR$ (N)
170  FOR I = 1 TO  LEN (N$)
180  PRINT  MID$ ("PERNAMBUCO",( VAL ( MID$ (N$,I,1))),1);
190  NEXT I
200  END
```

Rodando, temos:

```
CODIFICANDO

ENTRE O NUMERO A
SER   CODIFICADO

?23185
ERPUA
```


EXERCÍCIO 9.4

```
100  PRINT "ZENIT POLAR"
110  PRINT
120  PRINT "ENTRE A  PALAVRA"
130  PRINT "A SER CODIFICADA"
140  PRINT
150  INPUT A$
160  FOR I = 1 TO  LEN (A$)
170  LET X$ =  MID$ (A$,I,1)
180  LET Y$ = X$
185  GOSUB 300
190  PRINT Y$;
200  NEXT I
210  PRINT
215  PRINT
220  PRINT "NOVA PALAVRA? (S OU N)"
230  INPUT R$
240  IF R$ = "S" GOTO 110
250  GOTO 400
300  IF X$ = "P" THEN Y$ = "Z"
301  IF X$ = "Z" THEN Y$ = "P"
302  IF X$ = "O" THEN Y$ = "E"
303  IF X$ = "E" THEN Y$ = "O"
304  IF X$ = "L" THEN Y$ = "N"
305  IF X$ = "N" THEN Y$ = "L"
306  IF X$ = "I" THEN Y$ = "A"
307  IF X$ = "A" THEN Y$ = "I"
308  IF X$ = "T" THEN Y$ = "R"
309  IF X$ = "R" THEN Y$ = "T"
310  RETURN
400  END
```

Rodando, temos:

```
ZENIT POLAR

ENTRE A  PALAVRA
A SER CODIFICADA

?ZENIT
POLAR

NOVA PALAVRA? (S OU N)
?S

ENTRE A  PALAVRA
A SER CODIFICADA

?PALMEIRAS
ZINMOATIS

NOVA PALAVRA? (S OU N)
?S
```

```
ENTRE A  PALAVRA
A SER CODIFICADA

?BASIC
BISAC

NOVA PALAVRA? (S OU N)
?N
```

EXERCÍCIOS DO CAPÍTULO 10

EXERCÍCIO 10.1

Comandos:

RUN: Limpa todas as variáveis, sempre;

CONT: Mantém os valores das variáveis, sempre;

GOTO: Mantém os valores das variáveis, se não existiu mensagem de erro.

EXERCÍCIO 10.2

a.) Existência de uma mensagem de erro.

b.) Modificação do programa.

EXERCÍCIO 10.3

```
FNA (X)     =    - 48
FNA (Y)     =    -  3
FNA (X + T)=     - 66
FNA (10)    =    - 42
```


EXERCÍCIOS DO CAPÍTULO 11

EXERCÍCIO 11.1

```
100  REM  ESCREVENDO DIRETAMENTE
110  REM  LIMPANDO A TELA
120  CALL 8340
130  REM  ENDERECO DO VIDEO
140  LET I = 10240
150  REM  ESCREVENDO
160  READ X
170  IF X = 0 GOTO 300
180  POKE I,X
190  LET I = I + 1
200  GOTO 160
210  DATA  77,69,85,32,80,82,73,77,69,73,82,79,32,80,8
     2,79,71,82,65,77,65,0
300  END
```

EXERCÍCIO 11.2

```
100  REM  ESCREVENDO DIRETAMENTE
110  REM  LIMPANDO A TELA
120  CALL 8340
130  REM  ENDERECO DO VIDEO
140  LET I = 10239
150  REM  ESCREVENDO
160  LET X$ = "MEU PRIMEIRO PROGRAMA"
170  FOR J = 1 TO  LEN (X$)
180  LET X =  ASC ( MID$ (X$,J,1))
190  POKE I + J,X
200  NEXT J
210  END
```

EXERCÍCIO 11.3

```
100  PRINT "LETRA OU NUMERO?"
110  PRINT
120  PRINT "ENTRE COM UM CARACTERE"
130  PRINT "E EU DIREI O QUE E'"
140  PRINT
150  INPUT A$
160  IF  ASC (A$) > 64 AND  ASC (A$) < 91 GOTO 300
180  IF  ASC (A$) > 47 AND  ASC (A$) < 58 GOTO 340
200  PRINT "NEM NUMERO NEM LETRA"
210  GOTO 400
300  PRINT "LETRA"
310  GOTO 400
340  PRINT "NUMERO"
400  PRINT
410  PRINT "NOVO CARACTERE? (S OU N) "
420  INPUT R$
430  IF R$ = "S" GOTO 110
450  END
```

Rodando, temos:

```
LETRA OU NUMERO?

ENTRE COM UM CARACTERE
E EU DIREI O QUE E'

?#
NEM NUMERO NEM LETRA

NOVO CARACTERE? (S OU N)
?S

ENTRE COM UM CARACTERE
E EU DIREI O QUE E'

?A
LETRA

NOVO CARACTERE? (S OU N)
?S

ENTRE COM UM CARACTERE
E EU DIREI O QUE E'

?3
NUMERO

NOVO CARACTERE? (S OU N)
?N

]RUN
LETRA OU NUMERO?

ENTRE COM UM CARACTERE
E EU DIREI O QUE E'

?#
NEM NUMERO NEM LETRA

NOVO CARACTERE? (S OU N)
?S

ENTRE COM UM CARACTERE
E EU DIREI O QUE E'

?A
LETRA

NOVO CARACTERE? (S OU N)
?S

ENTRE COM UM CARACTERE
E EU DIREI O QUE E'

?3
NUMERO

NOVO CARACTERE? (S OU N)
?N
```

EXERCÍCIO 11.4

```
100  REM  LIMPANDO A TELA
105  GOSUB 500
110  REM  POSICAO INICIAL
115  LET X = 10640
120  POKE X, ASC ("A")
130  REM  LEITURA DO TECLADO
140  LET Y =  PEEK (11264)
150  IF Y = 0 GOTO 140
160  LET Z =  PEEK (11265)
165  REM  PREPAR. DO MOVIMENTO
170  IF Z = 66 THEN W = 40
180  IF Z = 67 THEN W =  - 40
190  IF Z = 68 THEN W = 1
200  IF Z = 69 THEN W =  - 1
210  REM  LIMITANDO O MOVIMENTO
220  IF X + W < 10240 THEN X = 10240
230  IF X + W > 11039 THEN X = 11039
240  REM  LIMPAR A TELA
245  GOSUB 500
250  REM  NOVA POSICAO
260  POKE X + W, ASC ("A")
270  GOTO 140
500  REM   SUB LIMPAR A TELA
502  FOR I = 10240 TO 11039
504  POKE I,32
506  NEXT I
508  RETURN
600  END
```

ÍNDICE

BIBLIOGRAFIA

ALBRECHT, FINKLE AND BROWN - *BASIC* - Peoples Computer Company, 1969.

BASIC - Apple Computer Inc. - Order nº A2L ØØØ6, 1978.

B 7000/B 6000 SERIES BASIC REFERENCE MANUAL - Burroughs Corporation - Order nº 05001407-001, 1978.

COAN, SAMES S. - *ADVANCE BASIC* - Hayden Book Company.

FARINA, MÁRIO V. - *PROGRAMMING IN BASIC* - Pretince-Hall, Inc., 1969.

PEGELS, C. - *BASIC: A COMPUTER PROGRAMMING LANGUAGE* - Holden-Day, Inc. , 1973.

Impresso nas Oficinas da

RUMO GRÁFICA EDITORA LTDA

Rua Dr. Horácio da Costa n.º 1-A

C.G.C. 46.295.564/0001.08 — S. Paulo

OUTROS LANÇAMENTOS

Teoria e Desenvolvimento de Projetos de Circuitos Eletrônicos

Diodos, Transistores de Junção, FET, MOS, UJT, LDR, NTC, PTC, SCR, Transformadores, Amplificadores Operacionais e suas aplicações em Projetos de Fontes de Alimentação, Amplificadores, Osciladores, Osciladores de Relaxação e outras.
7.a Edição, 580 páginas.
Autores: Engos. Cipelli / Sandrini

Teoria e Processo de Desenvolvimento em Eletrônica

Estudo e Associação de Bipolos Passivos e Ativos, Circuitos Ressonantes, Aparelhos de Medidas, Válvulas, Semicondutores, Leis de Kirchhoff, Teoremas de Thevenin e Norton, Osciloscópio e outros. 2.a Edição, 259 páginas.
Autor: Eng.o Sidnei David

Microprocessadores 8080 e 8085 - Hardware - Vol. 1

Memórias RAM, ROM, PROM e EPROM, o 8224, 8228, 8080, 8085, 8255 e 8253, suas aplicações e montagem de um microprocessador. 3.a Edição, 138 páginas
Autor: Eng.º Antonio Carlos José Franceschini Visconti

Microprocessadores 8080 e 8085 - Software - Vol. 2

Estudo das instruções dos microprocessadores 8080 e 8085, Fluxogramas, iniciação a programação e desenvolvimento de programas com a utilização dos microprocessadores 8080 e 8085. 3.a Edição, 202 páginas.
Autor: Eng.º Antonio Carlos José Franceschini Visconti

Elementos de Eletrônica Digital

Iniciação a Eletrônica Digital, Álgebra de Boole, Minimização de Funções Booleanas, Circuitos Contadores, Decodificadores, Multiplex, Demultiplex, Display, Registradores de Deslocamento, Desenvolvimento de Circuitos Lógicos, Circuitos Somadores/Subtratores e outros. 5.a Edição, 504 páginas.
Autores: Capuano / Idoeta.