PROGRAMAÇÃO COM TK 2000

ALOISIO PINTO ALVES

Compatível com TK 2000/II
e micros da linha Apple

atlas

Programação com
TK 2000

Compatível com TK 2000/II
e micros da linha Apple

ALOISIO PINTO ALVES

Programação com TK 2000

Compatível com TK 2000/II e micros da linha Apple

2ª Edição

EDITORA ATLAS S.A.
Rua Conselheiro Nébias, 1384 (Campos Elísios)
Caixa Postal 7186 — Tel.: (011) 221-9144 (PABX)
01203 São Paulo (SP)

SÃO PAULO
EDITORA ATLAS S.A. — 1986

(c) 1985 by EDITORA ATLAS S.A.
Rua Conselheiro Nébias, 1384 (Campos Elísios)
Caixa Postal 7186 — Tel.: (011) 221-9144 (PABX)
01203 São Paulo (SP)

1ª Edição 1985; 2.ª Edição — Abril — 1986

ISBN 85-224-0106-3

Impresso no Brasil/**Printed in Brazil**

Depósito legal na Biblioteca Nacional, conforme Decreto nº 1.825, de 20 de dezembro de 1907.

TODOS OS DIREITOS RESERVADOS — É proibida a reprodução total ou parcial, de qualquer forma ou por qualquer meio, salvo com autorização, por escrito, do Editor.

Capa
Paulo Ferreira Leite

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

Alves, Aloísio Pinto.
A477p Programação com TK 2000 / Aloisio Pinto Alves. -- São Paulo : Atlas, 1986.

ISBN 85-224-0106-3

1. TK 2000 (Computador) — Programação I. Título.

17. CDD-651.8
85-1485 18. -001.6425

Índices para catálogo sistemático:

1. Programação : TK 2000 : Computadores : Processamento de dados 651.8 (17.) 001.6425 (18.)
2. TK 2000 : Computadores : Programação : Processamento de dados 651.8 (17.) 001.6425 (18.)

A minha esposa e filhos

SUMÁRIO

INTRODUÇÃO

Este livro é o resultado do trabalho de toda uma equipe envolvida no ensino de programação para microcomputadores. Essa equipe integra o corpo de profissionais do Fundo de Pesquisa do Instituto de Administração - FUNAD - órgão complementar de ensino e pesquisa da FACULDADE DE ECONOMIA E ADMINISTRAÇÃO DA UNIVERSIDADE DE SÃO PAULO.

Como um dos objetivos da Universidade é o apoio à comunidade, iniciou-se em 1984 uma série de três cursos voltados exclusivamente para o uso de microcomputadores. Esses cursos têm finalidades distintas e devem, por esse motivo, atingir uma população diferenciada. Neste livro, os três cursos aparecem como partes.

A Parte I, Introdução à Programação, com o uso da linguagem BASIC, é dirigida àqueles que nunca tiveram contato com o microcomputador. Tem como objetivo apresentar as noções básicas de programação, permitindo que se faça a transposição da solução de um problema para a linguagem do computador. É muito mais um curso para aprender a raciocinar em termos dos recursos do equipamento, do que o ensino de uma linguagem.

Na Parte I são apresentados apenas problemas que poderíamos classificar como lineares, isto é, o desenvolvimento do programa é totalmente linear no sentido de que não é permitido nenhum comando de desvio condicional, do tipo "se ocorrer esta condição, então..., caso contrário...". Com esta restrição têm-se problemas simplificados, o que permite ao aluno concentrar-se apenas na solução de problemas e no domínio dos recursos do equipamento.

A Parte II, Linguagem BASIC, tem por objetivo a solução de problemas mais complexos, com a inclusão do comando de desvio condicional e demais recursos da linguagem BASIC, exceto a manipulação de arquivos, objeto da Parte III.

Com o domínio da Parte II consegue-se resolver vários tipos de problemas, com diversos graus de dificuldade, estando a eficiência do resultado dependendo diretamente do tempo dedicado, principalmente na resolução dos exercícios propostos.

A Parte III tem como objetivo apresentar os conceitos básicos de manipulação de arquivos, sendo que algumas noções sobre a montagem de Sistemas de Processamento também são apresentadas.

Da maneira como foram preparadas, é possível estudar só a Parte I, ou I e II ou as três. Estudar só a Parte I significa querer ter um primeiro contato com o computador, conhecer sua estrutura de funcionamento.

O estudo das Partes I e II significa o desejo de realizar atividades com o uso do computador, as quais, sem o uso deste recurso, seriam extremamente trabalhosas.

O estudo das três partes deve ser o objetivo de pessoas que realmente desejam desfrutar de todo o potencial de recursos de um microcomputador e que tenham tempo e disposição para isto, pois a última parte exige muito mais trabalho que as anteriores, bem có-

mo maior investimento em equipamento, devendo ser adquirido um "disk drive" que custa tanto quanto ou mais que o microcomputador.

O sucesso no aprendizado de qualquer das partes depende do empenho com que se tenta solucionar os exercícios propostos. Somente a leitura do livro não é suficiente para o domínio do conteúdo apresentado.

Convém ressaltar que este livro não substitui o Manual do Fabricante, pois cada um tem sua finalidade.

O Manual do Fabricante apresenta em detalhe todos os comandos existentes no equipamento, o que não é a finalidade deste livro, que se concentra na atividade de programação.

Para encerrar esta parte introdutória, gostaria de apresentar meus agradecimentos à equipe que tornou possível este livro, devendo-se destacar o Prof. Nicolau Reinhard e o Prof. Hiroo Takaoka, companheiros de longos anos, os quais absorveram muitas das minhas atividades de maneira que pudesse dedicar com mais intensidade à montagem deste livro. Um agradecimento especial ao Sr. Luiz Álvaro Leão Gil, que supervisionou a execução destes cursos na FEA, USP. Aos instrutores Janio Jiro Kobori, Clodoaldo Tosi e Sergio Muller pela valiosa cooperação no aprimoramento do material didático. Ao Sergio devo ainda a incumbência, muito bem desempenhada, de adaptar o curso ao equipamento TK2000 color. Finalmente não poderia deixar de mencionar a equipe de datilógrafas, Srtas. Ida Cambi, Dirce Rodrigues Soares, Marta Toshie Ishiy, Celia Satie Shirai e Lizete Capano, as quais pacientemente produziram as diversas versões deste texto.

ALOISIO PINTO ALVES

PARTE I

INTRODUÇÃO A PROGRAMAÇÃO

CAPITULO 1

O COMPUTADOR E A PROGRAMAÇÃO

1.1 CONCEITOS SOBRE COMPUTADOR

Para se resolver um problema através do uso do computador é preciso que as instruções de como solucionar este problema já estejam previamente armazenadas na **m e m ó r i a** da máquina. Além disso, os valores numéricos ou os dados do nosso problema também devem ser armazenados ou então os valores deverão ser fornecidos durante a execução do programa.

Como se observa, quem resolveu o problema foi o programador, o qual decompôs o problema em instruções ou comandos reconhecíveis pelo computador. Este nada mais fez que executar as instruções num tempo muito pequeno. Frações de segundo.

A grande dificuldade reside em saber como resolver um problema qualquer, usando somente as instruções existentes na máquina.

Como foi comentado, na **m e m ó r i a** do computador devem estar presentes, ao mesmo tempo, as instruções e os dados necessários para a execução.

Ao conjunto de instruções preparadas pelo programador do computador dá-se o nome de **PROGRAMA.**

PROGRAMA { INSTRUÇÃO 1, INSTRUÇÃO 2, INSTRUÇÃO 3

Se pudéssemos representar a **MEMÓRIA** do computador de forma física e simplificada, teríamos alguma coisa como:

00001	00002	00003	00004
INSTRUÇÃO 1	INSTRUÇÃO 2	INSTRUÇÃO 3	INSTRUÇÃO 4
00084	00085	00086	00087

Os números 00001, 00002, 00003 são conhecidos como endereços das respectivas memórias.

O programa estaria armazenado em algumas partes da memória e em outras os dados numéricos necessários.

As posições de memória possuem uma identificação 0001,0002 etc., até o limite máximo da capacidade de memória do computador. Os microcomputadores encontrados em casas comerciais têm capacidade de memória que varia de 16.000 a 64.000 posições de memória.

Acima desse limite, as máquinas são mais para uso profissional, dispondo-se hoje de máquinas com capacidade além de 10.000.000 de posições de memória principal. Como veremos adiante, existem grandes memórias de armazenamento auxiliar com capacidade de bilhões de posições.

Para ilustrar o funcionamento geral do computador, vamos simular a execução de um programa armazenado na memória.

Deseja-se resolver o seguinte problema:

Para cada aluno de uma classe, calcular a sua média e apresentá-la juntamente com seu número no vídeo. Os alunos possuem duas notas.

As instruções para resolver este problema são:

```
INSTRUÇÃO  1  - LER NÚMERO DO ALUNO
INSTRUÇÃO  2  - LER NOTA1 DO ALUNO
INSTRUÇÃO  3  - LER NOTA2 DO ALUNO
INSTRUÇÃO  4  - CALCULAR A MÉDIA
INSTRUÇÃO  5  - APRESENTAR NO VÍDEO NÚMERO E MÉDIA DO
                ALUNO
INSTRUÇÃO  6  - VOLTAR A INSTRUÇÃO 1
```

Após a colocação deste programa na memória do computador, tem-se:

00001	00002	00003	00004
LER NÚMERO DO ALUNO	LER NOTA1 DO ALUNO	LER NOTA2 DO ALUNO	CALCULAR A MÉDIA DO ALUNO
00005	00006		
APRESENTAR NO VÍDEO O Nº E A MÉDIA DO ALUNO	VOLTAR A INSTRUÇÃO 1		
00049	00050	00051	00052
NÚMERO	NOTA1	NOTA2	MÉDIA

Neste ponto convém ressaltar que:

1. O computador executa uma única instrução de cada vez.

2. É possível atribuir **NOME** aos endereços de memória, conforme ocorreu com:

00049 - é reconhecido por NÚMERO
00050 - é reconhecido por NOTA1
00051 - é reconhecido por NOTA2
00052 - é reconhecido por MÉDIA

3. Após executar uma instrução, automaticamente o computador se prepara para executar a instrução contida no endereço seguinte.

4. Uma posição de memória tem um endereço de localização e um conteúdo, o qual pode ser um número, uma letra ou um caractere especial como (.,/? etc.)

00129
35

neste caso, o endereço 00129 contém o valor 35

Supondo-se que o primeiro aluno, número 130, tenha notas 4 e 6, tem-se a seguinte configuração nos endereços:
00049-NÚMERO,000050-NOTA1, 00051-NOTA2 após a leitura destes dados:

Memória após execução das instruções 00001 00002 00003

00049	00050	00051	00052
NÚMERO 130	NOTA1 4	NOTA2 6	MÉDIA

Somente após executar a instrução colocada em 00004: calcular a média do aluno e colocar na posição MÉDIA o valor 5.

Memória após execução das instruções 00001 00002 00003

00049	00050	00051	00052
NÚMERO 130	NOTA1 4	NOTA2 6	MÉDIA

Somente após executar a instrução coloçada em 00004: calcular a média do aluno e colocar na posição MÉDIA o valor 5.

Memória após execução da instrução 00004

00049	00050	00051	00052
NÚMERO 130	NOTA1 4	NOTA2 6	MÉDIA 5

A execução das instruções 00005 e 00006 não altera em nada os valores de NÚMERO, NOTA1, NOTA2 e MÉDIA.

Somente quando for executada a instrução 00001 novamente, é que será substituído o valor 130 pelo número do próximo aluno.

Com a execução das instruções 00002 e 00003 serão substituídas as notas do aluno antigo pelo novo.

ALUNO 2

NÚMERO 150
NOTA1 2
NOTA2 4

Posições de memória após execução das instruções 00001 00002 00003 para aluno número 150

00049	00050	00051	00052
NÚMERO 150	NOTA1 2	NOTA2 4	MÉDIA 5

Pode-se perceber que a posição MÉDIA ainda contém o valor referente ao aluno de número 130. Este valor só será alterado quando for executada a instrução contida em 00004, que é o cálculo da nova média.

Memória após execução instrução 00004

00049	00050	00051	00052
NÚMERO 150	NOTA1 2	NOTA2 4	MÉDIA 3

A seguir apresentamos um quadro do que foi explicado:

INSTRUÇÃO EXECUTADA	POSIÇÃO MEMÓRIA 00049-NUMERO	POSIÇÃO MEMÓRIA 00050-NOTA1	POSIÇÃO MEMÓRIA 00051-NOTA2	POSIÇÃO MEMÓRIA 00052-MÉDIA
00001	(130)	_	_	_
00002	130	(4)	_	_
00003	130	4	(6)	_
00004	130	4	6	(5)
00005	130	4	6	5
00006	130	4	6	5
00001	(150)	4	6	5
00002	150	(2)	6	5
00003	150	2	(4)	5
00004	150	2	4	(3)
00005	150	2	4	3
00006	150	2	4	3
00001	( )	2	4	3

Este processo continua enquanto forem fornecidos dados ao programa.

EXERCÍCIOS

Preparar as instruções para resolver o seguinte problema:

1. Dado o ano de nascimento, calcular a idade desta pessoa.

2. Simular a execução para os seguintes dados: 1965, 1972.
   Apresentar o quadro-resumo.

1.2 NOÇÕES DE PROGRAMAÇÃO

Para colocar-se um programa na memória RAM do MC é preciso que o problema a ser resolvido seja subdividido em instruções disponíveis no computador que vai ser utilizado.

Ao realizar o processamento de um programa contido na memória, o computador executa sequencialmente uma instrução após a outra.

Está subentendido pelo que foi dito que se deve armazenar as instruções numa sequência que resolve o problema desejado.

Esta é a grande dificuldade da programação: transformar certo problema num conjunto sequencial de instruções.

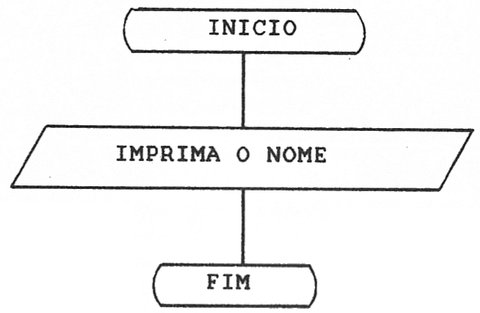
EXERCÍCIO 1

Deseja-se ver um nome impresso no vídeo.

Dado o problema, é preciso que se construa um conjunto de instruções que o resolva.

Resolução:

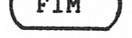
No caso deste problema tão simples, basta colocar uma única instrução:



Pelo que foi colocado, pode-se concluir que:

1. A solução completa de um problema se inicia com duas etapas preliminares:
   - Enunciado (do problema);
   - Resolução (na forma de símbolos gráficos).

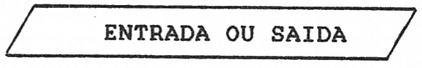
2. A representação gráfica da solução de um problema, que é chamada FLUXOGRAMA é composta de símbolos predefinidos:



Todo problema tem um início...

... e um fim

A representação daquilo que sai no vídeo, ou que entra para a memória do computador, é feita por um paralelogramo.

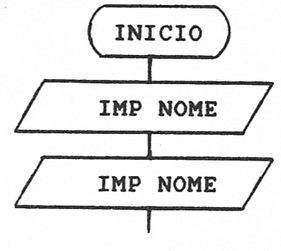


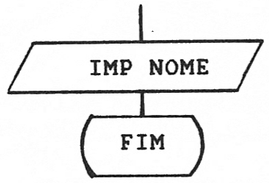
Dentro do paralelogramo coloca-se a ação desejada: imprimir NOME; ler uma nota; ler um nome.

EXERCÍCIO 2

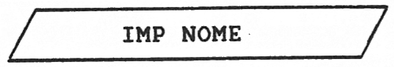
Imprimir um nome três vezes.

FLUXOGRAMA



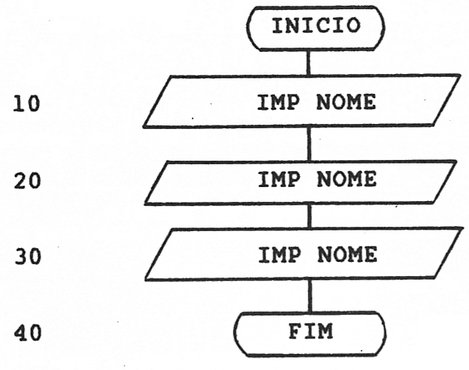


É possível perceber que este processo de resolução não é nada prático, pois, se quisermos imprimir o NOME um número maior de vezes, teríamos de codificar:

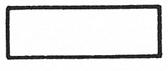


diversas vezes

Cada instrução de um programa fica armazenada numa posição de memória. Geralmente, estas posições são numeradas automaticamente a partir de 10 e em variações de 10.
Desta forma, o exercício 2 estaria armazenado em:

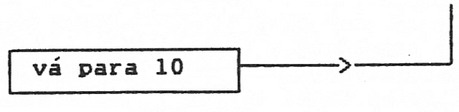


Neste exercício não vamos utilizar instruções iniciais, razão pela qual o programa se inicia na instrução 10.
Para simplificar este problema bastaria introduzir uma nova instrução, que terá uma representação gráfica diferente:

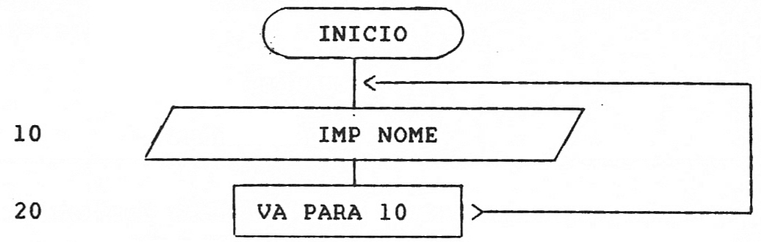


símbolo para um procedimento diferente de entrada e saída.

A instrução que resolve o nosso problema é:



EXERCÍCIO 2A



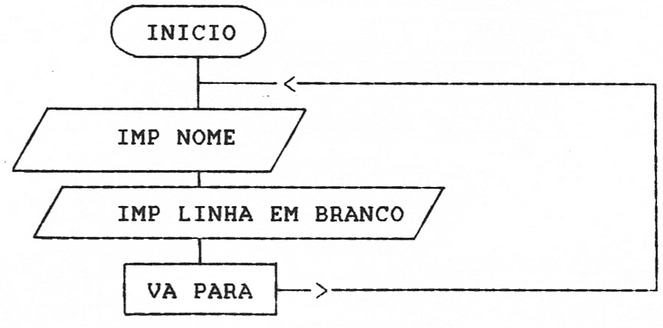
Notar que este problema não tem FIM. O computador irá imprimir um NOME, depois a próxima instrução indica que deve ser executada novamente IMP NOME, e assim indefinidamente ou até que se interrompa mecanicamente a execução deste programa.

Como resultado da execução, temos no vídeo:

```
JOSE CARLOS
JOSE CARLOS
JOSE CARLOS
JOSE CARLOS
```

EXERCÍCIO 3
Imprimir um nome e saltar uma linha. Repetir este processo várias vezes.

FLUXOGRAMA



SAÍDA NO VÍDEO

```
JOSE CARLOS
JOSE CARLOS
JOSE CARLOS
```

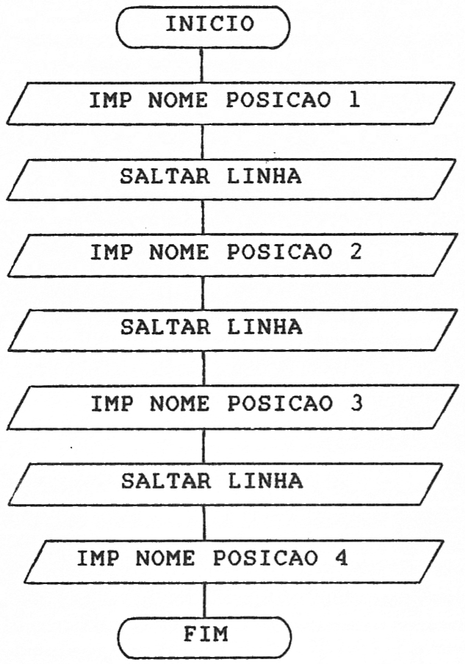
EXERCÍCIO 4
Imprimir um NOME e uma linha em branco várias vezes. O nome deverá ser deslocado para a direita em cada nova impressão.

SAÍDA NO VÍDEO

```
JOSE CARLOS
     JOSE CARLOS
          JOSE CARLOS
               JOSE CARLOS
```

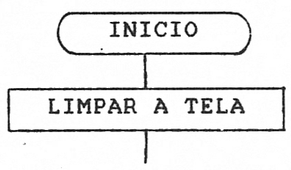
Neste exercício temos quatro linhas distintas, que devem ser representadas no fluxograma:

FLUXOGRAMA



Ao executarmos os programas anteriores pode-se constatar quais saídas não se iniciarão sempre na mesma posição, o que é muito desagradável para um bom planejamento. Para garantir que isto aconteça é necessário limpar a tela ao iniciar o processamento.

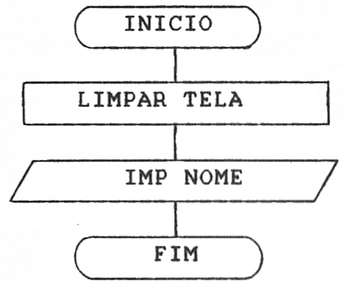
Em todo problema, como instrução inicial teremos a limpeza da tela:



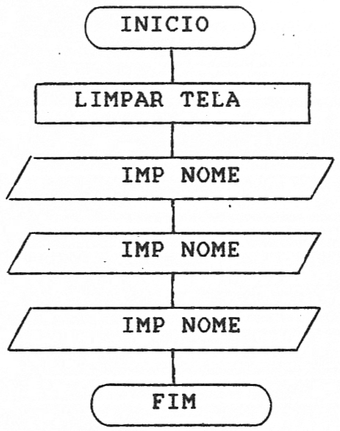
EXERCÍCIO 5

Refazer os Exercícios 1 a 4 com limpeza inicial da tela (tempo 10 minutos).

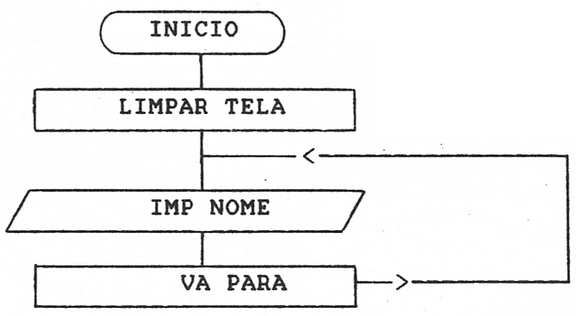
FLUXOGRAMA
EX. 1



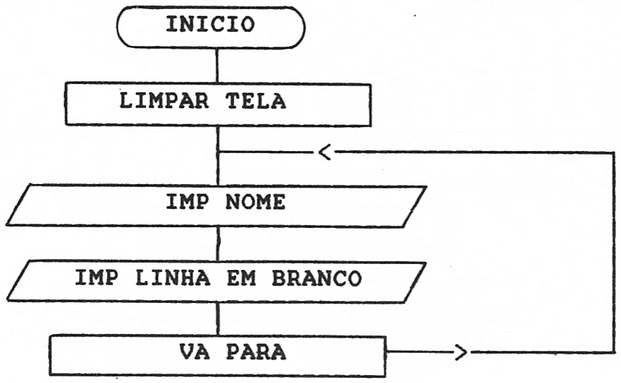
FLUXOGRAMA
EX. 2



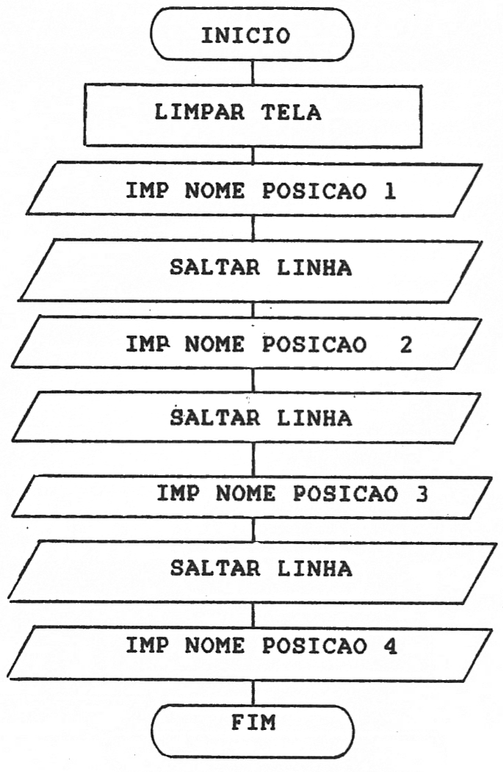
FLUXOGRAMA
EX. 3A



FLUXOGRAMA
Ex. 3



FLUXOGRAMA
EX.4



A terceira etapa da solução de um problema é planejar em detalhe as saídas do programa. Para isto utiliza-se uma folha especial, na qual temos traçado um conjunto de linhas, cada uma delas com um conjunto de colunas.

No curso que utiliza o TK-2000 deve-se planejar saídas com um máximo de 24 linhas com 40 colunas cada linha.

Este planejamento da saída tem o nome técnico de "LAYOUT" de saída.

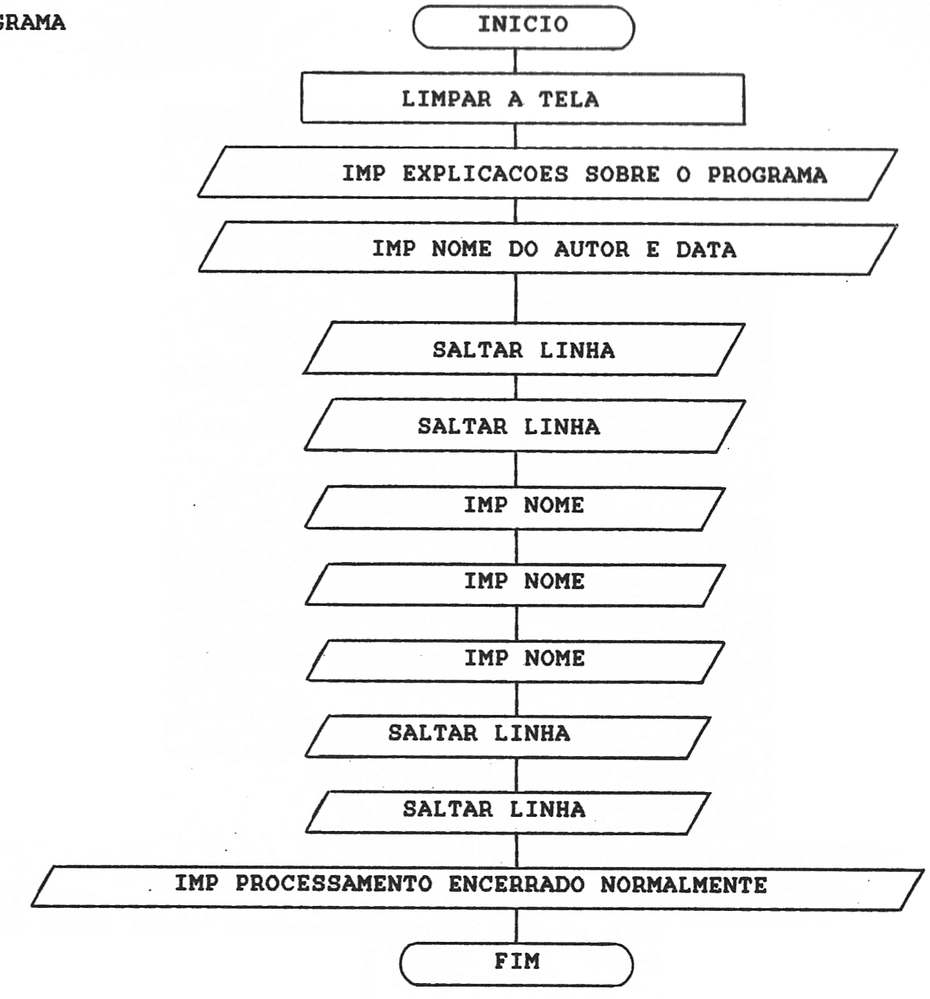
É conveniente que todo programa mantenha comunicação com o seu usuário e o informe sobre o que o programa faz, por quem e quando foi escrito, assim como quando o processamento se encerra normalmente.

Essa comunicação é feita através de saídas no vídeo ao iniciar o programa e no seu encerramento.

. ETAPA 1 _ Definição do problema
. ETAPA 2 _ Fluxograma
. ETAPA 3 _ "Layout"

Para exemplificar o que foi dito, vamos introduzir comentários no Exercício 2.

FLUXOGRAMA
EX. 2



A folha de "Layout" de saída do Exercício 2 está representada a seguir. Fazer o "Layout" de saída para os exercícios 1,2A,3 e 4.

EXERCICIOS SUPLEMENTARES

EXERCICIO 1

Fazer o FLUXOGRAMA do exercício a seguir e, numa folha de codificação, fazer o "layout" de saída.

O programa deve imprimir os dados de identificação de um funcionário.

```
FICHA DE IDENTIFICAÇÃO DE FUNCIONÁRIO
**************************************
*                                    *
*NOME................................*
*                                    *
*DATA NASCIMENTO.....................*
*                                    *
*R.G.................................*
*                                    *
*CIC.................................*
*                                    *
*ENDERECO............................*
*                                    *
**************************************
```

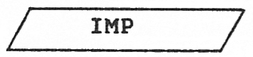
EXERCÍCIO 2

- Propor o enunciado de um exercício.
- Fazer o fluxograma do mesmo.
- Preparar o "Layout" de saída em folha de codificação.

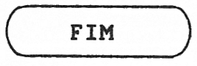
1.3 CODIFICAÇÃO EM BASIC DE UM FLUXOGRAMA (ETAPA 4 NA RESOLUÇÃO DE UM PROBLEMA)

A última etapa na solução do problema é transpor o problema da linguagem de fluxograma para a linguagem entendida pelo microcomputador - BASIC.

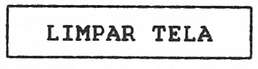
No tópico 1.2 foram apresentados quatro comandos da linguagem BASIC, que serão formalizados a seguir:



PRINT



END



HOME

Estes três comandos aparecem no Exercício 1.

O comando PRINT deve ser seguido do conteúdo que se deseja imprimir colocado entre aspas.

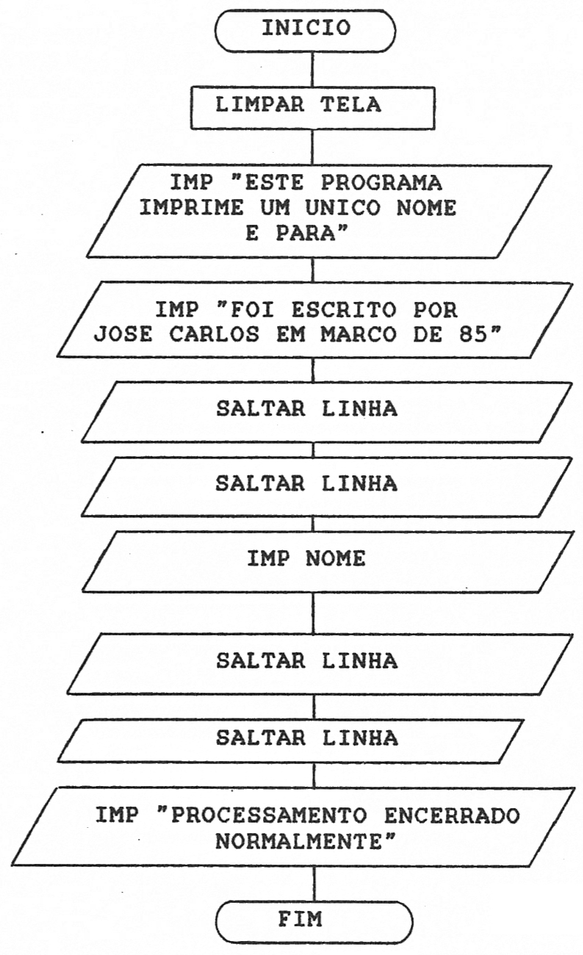
```
PRINT "MEU NOME E "
```

Sairá no vídeo durante a execução:

```
MEU NOME E
```

Conforme mencionado anteriormente, um programa é armazenado através de instruções sequenciais, numeradas geralmente de 10 em 10.

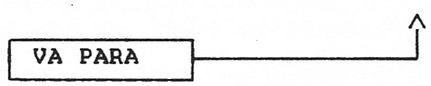
Exemplo da codificação do Exercício 1:




```
10 HOME
20 PRINT "ESTE PROGRAMA IMPRIME"
25 PRINT "UM UNICO NOME E PARA"
30 PRINT "FOI ESCRITO POR JOSE"
35 PRINT "EM MARCO DE 85"
```

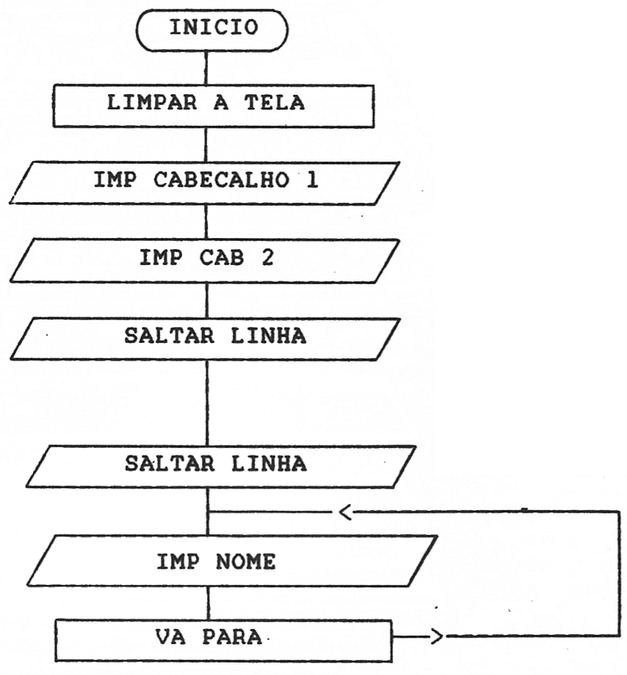
```
 40 PRINT
 50 PRINT
 60 PRINT " J O S E   C A R L O S "
 70 PRINT
 80 PRINT
 90 PRINT "PROCESSAMENTO ENCERRADO NORMALMENTE"
100 END
```

O último comando visto foi o desvio incondicional:



Na linguagem BASIC é o comando:
GOTO

Vamos codificar o Exercício 2A:




```
10 HOME
20 PRINT "PROGRAMA PARA IMPRIMIR"
25 PRINT "UM NOME DIVERSAS VEZES"
30 PRINT "ESCRITO POR ANTONIO BOA VIDA"
35 PRINT "EM MARCO DE 85"
40 PRINT
50 PRINT
60 PRINT "A N T O N I O  B O A  V I D A"
70 GO TO  60
```


Fazer a codificação BASIC para os exercícios 2,3,4 (tempo 15 minutos). Fazer o que puder neste tempo.

1.3.1 Comandos para introduzir e operar um programa Basic no TK-2000

RUN - <CONTROL><C> - LIST

Após ligar o computador, podemos imediatamente iniciar a digitação das instruções de um programa.

Vale lembrar que cada instrução para ser aceita pelo TK-2000 deve ser seguida da tecla RETURN.

Exemplo de aplicação (exercício 2A)

```
10 HOME
20 PRINT "PROGRAMA PARA IMPRIMIR UM NOME"
25 PRINT "DIVERSAS VEZES"
30 PRINT "ESCRITO POR ANTONIO BOA VIDA"
35 PRINT "EM MARCO DE 85"
40 PRINT
50 PRINT
60 PRINT "A N T O N I O  B O A  V I D A"
70 GOTO 60
```

Para executar um programa previamente armazenado na memória deve ser digitado o comando RUN.

Se na memória do computador estiver armazenado o exercício 2A (por exemplo), verificaremos que o computador repetirá indefinidamente a impressão do nome. Para interromper a execução do programa devemos pressionar simultaneamente as teclas <CONTROL> e <C>.

Após a execução, para verificarmos o conteúdo do programa (que ainda está na memória) basta digitar o comando LIST.

Aparecerá na tela a listagem do programa.

Sempre que se desejar, basta teclar LIST e a listagem do programa aparecerá na tela.

Deve-se tentar colocar os programas apresentados no computador. O leitor deve escolher um dos exercícios apresentados.

1.3.2 Comandos STOP e CONT

A apresentação de um programa pode ser melhorada com a introdução de dois novos comandos:

STOP - este comando interrompe a execução do programa.
CONT - reinicia a execução do programa.

Estes comandos podem ser inseridos após as mensagens iniciais e antes da final. Consegue-se manter no vídeo só a mensagem principal do programa, apagando-se as mensagens iniciais.

```
10 HOME
20 PRINT "ESTE PROGRAMA IMPRIME"
25 PRINT "UM UNICO NOME E PARA"
30 PRINT "FOI ESCRITO POR JOSE CARLOS"
35 PRINT "EM MARCO DE 85"
40 PRINT
50 PRINT
60 PRINT "DIGITE A PALAVRA CONT E APERTE"
65 PRINT "A TECLA RETURN PARA CONTINUAR"
```

```
 70 STOP
 80 HOME
 90 PRINT "J O S E   C A R L O S   E  O  M A I O R"
100 PRINT
110 PRINT
120 PRINT "PROCESSAMENTO ENCERRADO NORMALMENTE"
130 END
```

Alterar os exercícios 2,2a para incluir estes novos comandos.

(tempo 15 minutos)

O tempo estimado para este exercício é de 15 minutos.

1.4 VARIAVEL NUMÉRICA

Para trabalhar com números no computador é preciso que sejam identificados, pois só assim será possível distinguir um valor de outro.

A	Z	N1		

A cada valor que se deseja armazenar na memória do computador é preciso associar um nome.

1.4.1 Nome de uma variável numérica

- Inicia-se por uma letra.
- Máximo de dois caracteres, que podem ser duas letras ou letra e número.

Exemplo: A, B, AB, BA, N1, NZ, CP, X9

Para colocar um valor numa posição de memória temos duas opções:

Entrar com o valor através do COMANDO INPUT ou definir o valor da variável.

1.4.2 Comando INPUT

Este comando permite transferir um valor do teclado para a memória.

INPUT ⊢——————⊣ colocar nesta posição o nome da variável que se deseja armazenar.

Exemplo:

```
INPUT NM
```

Supondo que NM seja o número do aluno, ao ser executado este comando, na posição de memória rotulada de NM será armazenado o valor digitado no teclado.

Exemplo:


```
 5 HOME
10 PRINT "ENTRE COM SUA IDADE"
20 INPUT ID
30 HOME
40 PRINT "ENTRE COM SEU NUMERO DE SORTE"
50 INPUT NS
60 HOME
70 PRINT "VOCE ESTA COM ";ID;" ANOS E"
80 PRINT
90 PRINT "SEU NUMERO DE SORTE E ";NS
100 PRINT
110 PRINT "PROGRAMA ENCERRADO NORMALMENTE"
120 END
```

1.4.3 Definição do valor de uma variável numérica

Outra maneira de atribuirmos valor a uma variável é através de uma definição de valor:

```
10 A = 1985
20 M = 2
30 AL= 4
40 HOME
50 PRINT "ESTAMOS NO MES "; M; " DE "; A
60 PRINT
70 PRINT "ESTA E A AULA DE NUMERO "; AL
80 END
```

Fazer um programa que troque entre si as idades de duas pessoas. Na variável:

IP armazena a Idade da Primeira pessoa.

IS armazena a Idade da Segunda pessoa.

```
 10 HOME
 20 PRINT "QUAL SUA IDADE"
 30 INPUT IP
 40 HOME
 50 PRINT "QUAL A IDADE DA SEGUNDA PESSOA"
 60 INPUT IS
 70 IT = IP
 80 IP = IS
 90 IS = IT
100 PRINT "A IDADE DA PRIMEIRA PESSOA E "; IP
110 PRINT
120 PRINT "A IDADE DA SEGUNDA PESSOA E "; IS
130 PRINT
140 PRINT "VOCE ACHA QUE ESTA CORRETO?"
150 PRINT "NAO?, ENTAO VAMOS CONSERTAR."
155 PRINT "DIGITE CONT E TECLE RETURN"
160 PRINT "PARA CONTINUAR"
165 STOP
170 IT = IP
180 IP = IS
190 IS = IT
200 HOME
210 PRINT "A IDADE DA PRIMEIRA PESSOA E "; IP
220 PRINT
230 PRINT "A IDADE DA SEGUNDA PESSOA E "; IS
235 PRINT
240 PRINT "DIGITE CONT E TECLE RETURN"
```

```
245 PRINT "PARA CONTINUAR"
250 STOP
260 PRINT "AGORA ESTA CORRETO. DESCULPE MEU ERRO"
270 END
```

EXERCÍCIO

Fazer um programa para emitir uma lista de presença para o dia desejado.

SAÍDA NO VÍDEO

```
LISTA DE PRESENÇA DO DIA ----------------
        CURSO INTRODUÇÃO AO BASIC
   1.....................................
   2.....................................
   3.....................................
   4.....................................
   5.....................................
   6.....................................
   7.....................................
   8.....................................
   9.....................................
   10....................................
```

Este programa terá uma saída pela impressora.

1.5 EXPRESSÕES ARITMÉTICAS

Com frequência deseja-se transformar os dados de entrada, criando-se novas variáveis dentro de um programa. Por exemplo, a média final de um aluno em decorrência de suas notas parciais.

$$\text{MÉDIA FINAL} = \frac{\text{NOTA1} + \text{NOTA2} + \text{NOTA3}}{3}$$

O programa teve como dados iniciais as três notas, e o computador deve calcular a média final. Isto é conseguido através de uma expressão aritmética na linguagem BASIC.

```
MF = (N1 + N2 + N3) / 3
```

As expressões aritméticas são formadas por símbolos que indicam a operação desejada. Existem regras que estabelecem a prioridade na sequência de execução. Os símbolos e as prioridades estão colocados no quadro a seguir:

SIMBOLO	SIGNIFICADO	PRIORIDADE
+ -	soma subtração	4
* /	multiplicação divisão	3
^	exponenciação	2
( )	alterar prioridades	1

A maior prioridade é a 1 e a menor a 4.

Isso significa que, se numa expressão tivermos uma soma e uma multiplicação, a multiplicação será efetuada em primeiro lugar.

```
MF = N1*0.3 + N2*0.4 + N3*0.3
```

A expressão acima calcula a média final onde as notas têm peso 30%, 40% e 30%. Em primeiro lugar serão feitas as multiplicações e só depois as adições.

Supondo N1= 8 N2=7 N3=5, teremos como sequência interna de resolução:

```
MF = 8*0.3  +  7*0.4  +  5*0.3
MF = 2.4    +  2.8    +  1.5
MF = 6.7
```

A expressão acima poderia ser calculada de outra maneira:

$$\text{MF} = \frac{\text{N1*30} + \text{N2*40} + \text{N3*30}}{100}$$

Neste caso, a ordem das operações deve ser:

1. Realizar as multiplicações pelos pesos 30,40 e 30.
2. Realizar as somas.
3. Por último dividir por 100.

Existe uma inversão na prioridade predefinida, pois deseja-se realizar uma soma antes de efetuar a divisão.

Esta é uma situação bastante comum.

Nestes casos, o uso dos parênteses permite alterar a prioridade predefinida, pois tudo o que estiver dentro dos parênteses será feito em primeiro lugar.

Dentro dos parênteses as operações seguem as prioridades normais.

```
MF = (N1*30 + N2*40 + N3*30) / 100
```

O Resultado será:

```
MF = (8*30 + 7*40 + 5*30) / 100
MF = (240 + 280 + 150) / 100
MF = (670) / 100
MF = 6.7
```

Algumas vezes têm-se duas operações de mesma prioridade numa expressão. Neste caso, a ordem de operação é:

EXPRESSÃO ARITMÉTICA	BASIC
$A = \frac{B \times C}{D}$	A= B * C / D
$A = \frac{B}{\frac{C}{D}}$	A= B / (C/D)

Supondo B=12 C=6 D=2

$$A = \frac{12}{\frac{6}{2}} = \frac{12}{3} = 4$$ 4 é a resposta correta.

Se não for dada atenção às regras de execução, a tendência é fazermos em BASIC:

A = B / C / D o que resulta

A = 12/ 6 / 2 como a execução é da esquerda para a direita tem-se:

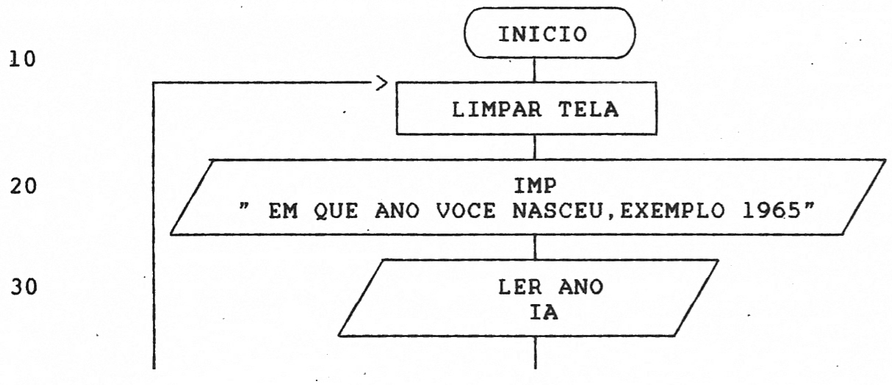
A = 2/2

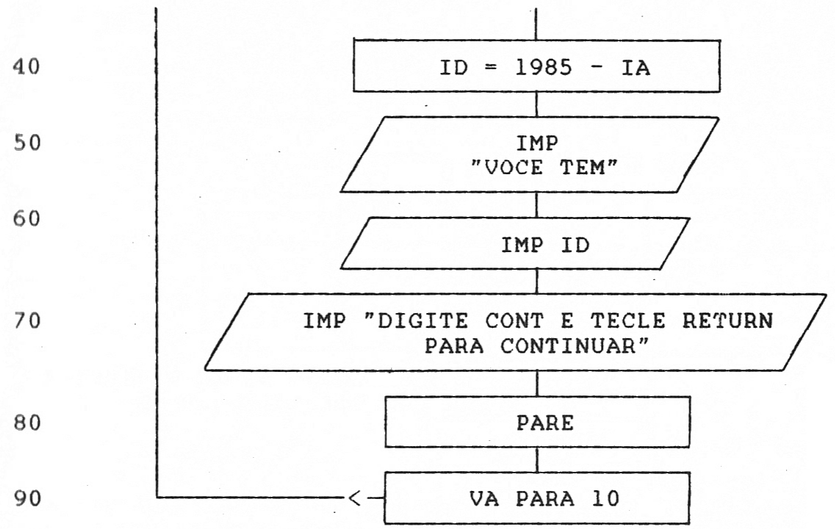
A = 1 com resultado evidentemente errado.

Exemplo:

Fazer um programa para calcular a idade de uma pessoa.
É fornecido o ano de nascimento.

FLUXOGRAMA



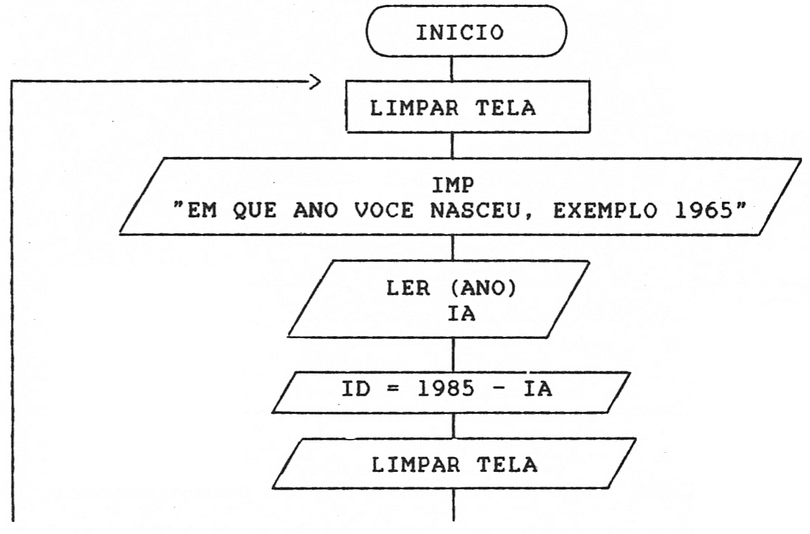


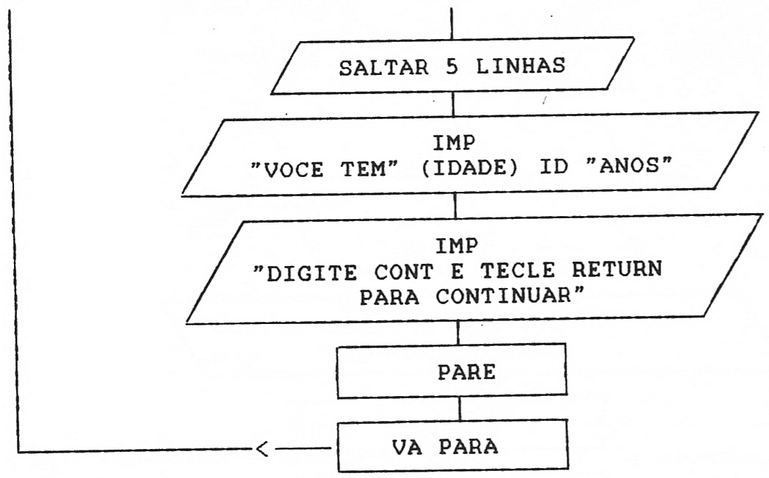
OBSERVAÇÃO: Em alguns microcomputadores é possível colocar num mesmo comando as linhas 20 e 30.
Em todos é possível colocar num único comando as linhas 50 e 60.

Neste ponto convém verificar com qual microcomputador se está trabalhando.

O TK - 85 não permite juntar 20 e 30
O TK - 2000, APPLE, CP-500, permitem juntar 20 e 30

FLUXOGRAMA SIMPLIFICADO





CODIFICAÇAO BASIC - FLUXOGRAMA SIMPLIFICADO

```
 10 HOME
 20 PRINT "EM QUE ANO VOCE NASCEU, EXEMPLO 1965"
 30 INPUT IA
 40 ID = 1985 - IA
 50 HOME
 60 PRINT
 70 PRINT
 80 PRINT
 90 PRINT
100 PRINT
110 PRINT "VOCE TEM " ; ID; " ANOS "
120 PRINT
130 PRINT "DIGITE CONT E TECLE RETURN"
135 PRINT "PARA CONTINUAR"
140 STOP
150 GO TO 10
```

EXERCÍCIO

Ler o número de um aluno, suas três notas e calcular sua média. Imprimir o número, as três notas e a média.

Repetir este processo para diversos alunos.

NUMERO	NOTA1	NOTA2	NOTA3	MEDIA
1901	5	6	7	6

OBSERVAÇAO: Cada aluno terá uma saída como foi exemplificado acima.

1.6 CONTADORES

Os exercícios propostos no Capítulo 2 utilizam um conceito muito usado em programação, que é a noção de CONTADOR.

Supondo-se que na memória do computador tenhamos definido uma variável S:

		S 0	

na posição S temos armazenado o valor 0.

Se for colocado num programa

S = S + 1,

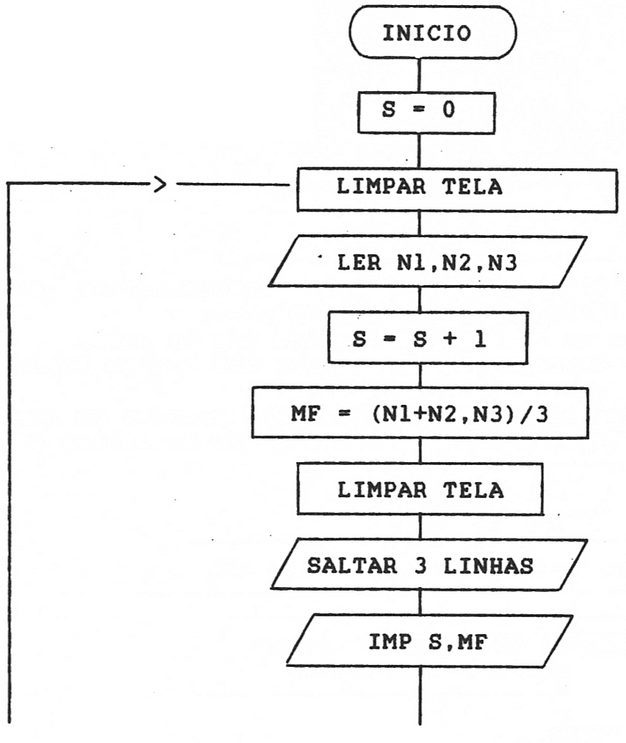
toda vez que esta instrução for executada teremos o contador S "contando" quantas vezes a instrução foi executada.

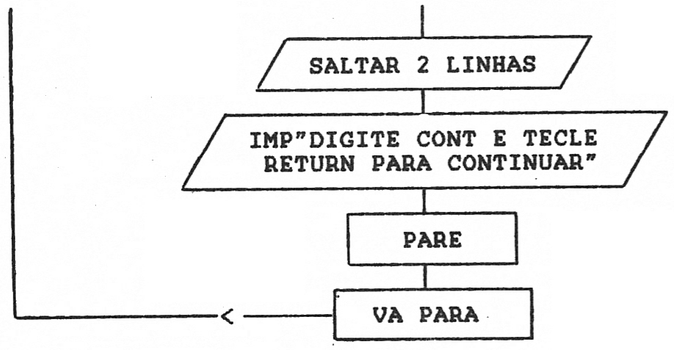
Se esta instrução é executada para cada aluno de uma classe, tem-se S contando o número de alunos da classe.

Exemplo:

Deseja-se calcular a média do aluno e contar o número de alunos da classe.

Imprimir a média e o número para cada aluno.






```
10  S = 0
15  HOME
20  PRINT "ENTRE COM AS 3 NOTAS "
30  INPUT "NOTA1 = "; N1
40  INPUT "NOTA2 = "; N2
50  INPUT "NOTA3 = "; N3
60  S = S + 1
70  MF = (N1 + N2 + N3) /3
75  HOME
80  PRINT
90  PRINT
100 PRINT
110 PRINT "ALUNO NUMERO "; S;" MEDIA = "; MF
120 PRINT
130 PRINT
140 PRINT "DIGITE CONT E TECLE RETURN"
145 PRINT "PARA CONTINUAR"
150 STOP
160 GOTO 15
```

Observar que o contador só pode ser "zerado" uma única vez. Por isso, a volta é para o comando 15.

1.7 VARIÁVEL ALFANUMÉRICA

Um problema torna-se muito mais compreensível se puder trabalhar com nomes em lugar de somente números.
Um aluno tem um número, mas também tem um nome.
Uma peça de estoque tem um número, mas também tem um nome.

Para reservar espaço na memória para guardar um nome, basta indicar que a variável é alfanumérica, o que se indica pelo sinal $.

Exemplo:

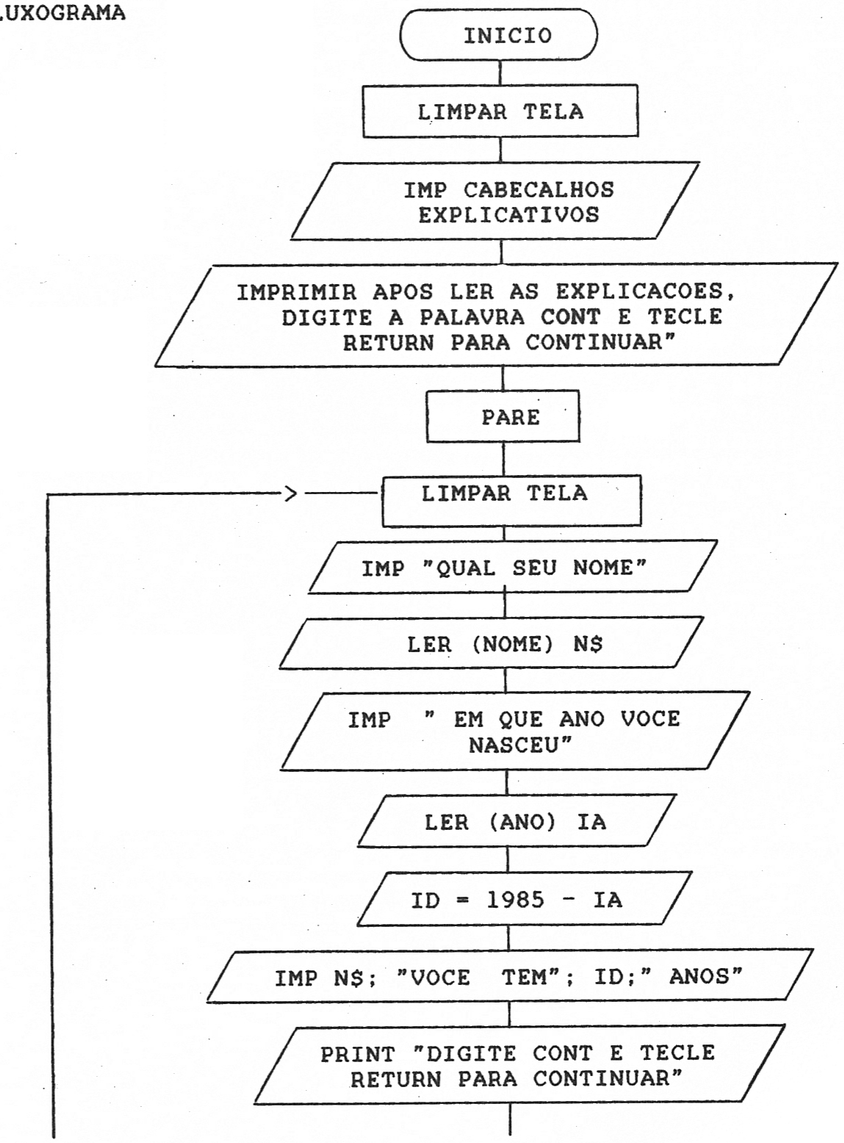
```
A  =  3
B$ =  "PARAFUSO"
C  =  4
D$ =  "PORCA"
```


Na memória do computador será armazenado:

A 3	B$ PARAFUSO	C 4
D$ PORCA		

Exemplo:
Ler o nome e o ano de nascimento de uma pessoa e imprimir seu nome e sua idade.

FLUXOGRAMA



CODIFICAÇÃO

```
 10  HOME
 20  PRINT "ESTE PROGRAMA IRA CALCULAR"
 25  PRINT "A IDADE DAS PESSOAS"
 30  PRINT
 40  PRINT "DIGITAR SEU NOME"
 45  PRINT "E O ANO DO SEU NASCIMENTO"
 50  PRINT
 60  PRINT "O ANO DO NASCIMENTO DEVERA"
 65  PRINT "SER COMPLETO, 1965,1972,ETC"
 70  PRINT
 75  PRINT "APOS LER AS INSTRUCOES"
 80  PRINT "DIGITE A PALAVRA CONT E"
 85  PRINT "TECLE RETURN PARA CONTINUAR"
 90  STOP
100  HOME
110  PRINT "QUAL O SEU NOME"
120  INPUT N$
130  PRINT "EM QUE ANO VOCE NASCEU"
140  INPUT IA
150  ID  = 1984 - IA
160  PRINT N$;" VOCE TEM ";ID ;" ANOS"
170  PRINT "DIGITE CONT E TECLE RETURN PARA CONTINUAR"
175  PRINT "PARA CONTINUAR"
180  STOP
190  GO TO 100
```

EXERCÍCIO

Fazer uma lista de presença contendo o dia, mês, disciplina, o número e nome de cinco alunos.

LISTA DE PRESENCA DO DIA.....MES.....
MATERIA.....

ORDEM	NUMERO	NOME	ASSINATURA
1	.....	.....	.....
2	.....	.....	.....
3	.....	.....	.....
4	.....	.....	.....
5	.....	.....	.....

CAPITULO 2

COMANDOS

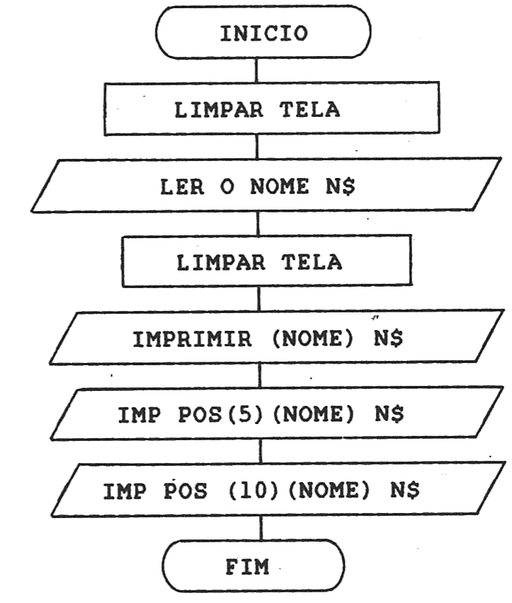
2.1 COMANDO TAB (N)

Este comando permite dar n-1 espaços antes de imprimir o conteúdo desejado.

Exemplo:
Imprimir um nome conforme quadro abaixo:

```
ANTONIO
        ANTONIO
                ANTONIO
```

FLUXOGRAMA

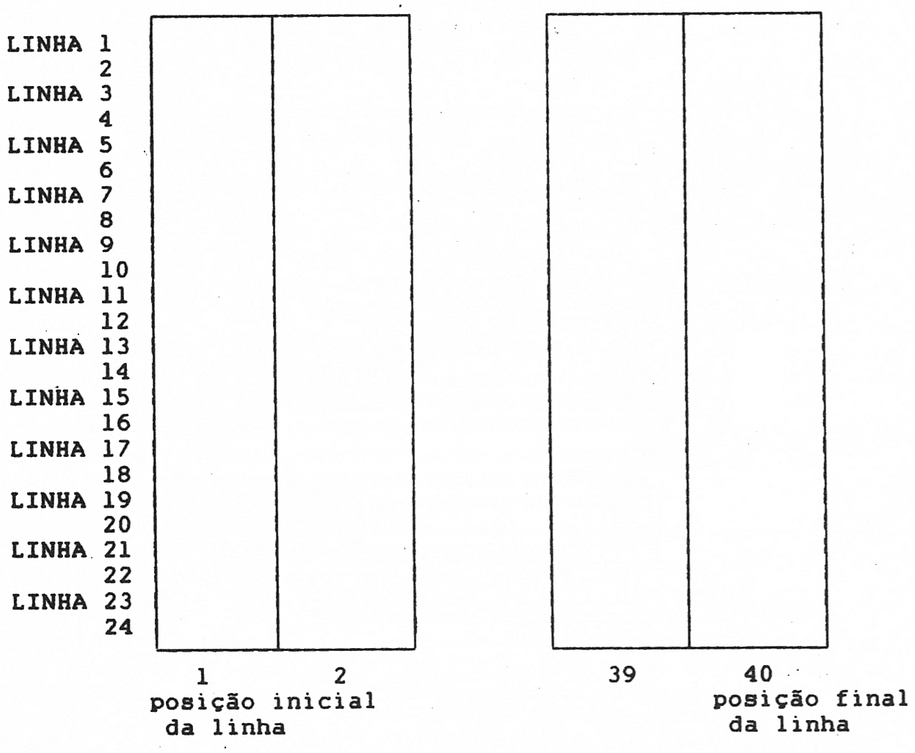


CODIFICAÇAO

```
 10  HOME
 20  PRINT "QUAL O SEU NOME"
 30  INPUT N$
 40  HOME
 50  PRINT N$
 60  PRINT
 70  PRINT TAB(5);N$
 80  PRINT
 90  PRINT TAB(10);N$
100  END
```

2.2 COMANDOS VTAB e HTAB

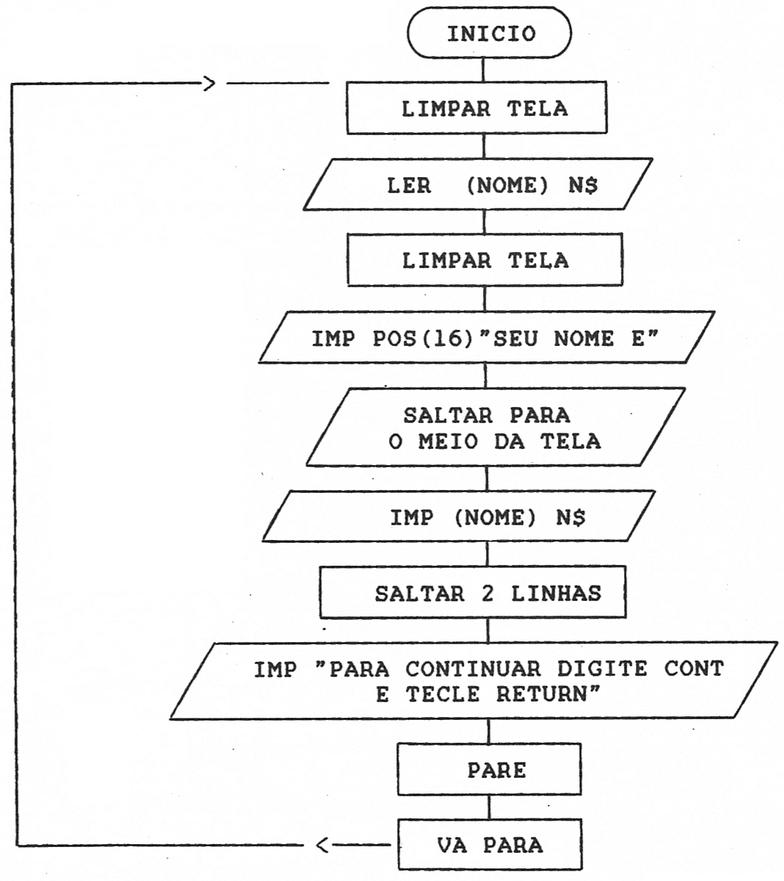
Estes comandos permitem deslocar o cursor para uma posição determinada da tela. Para isto é preciso saber como é a divisão da tela do TK - 2000. Ela tem 24 linhas e cada linha tem 40 posições.



O comando **VTAB** permite a movimentação vertical do cursor para qualquer linha da tela. O comando **HTAB** permite que o cursor se movimente horizontalmente em uma linha qualquer da tela.

Exemplo:
Imprimir um nome na linha central do vídeo.

FLUXOGRAMA



CODIFICAÇAO

```
 10  HOME
 20  PRINT "QUAL SEU NOME"
 30  INPUT N$
 40  HOME
 50  PRINT TAB(16); "SEU NOME E"
 60  VTAB 13
 70  HTAB 17
 80  PRINT N$
 90  PRINT
100  PRINT
110  PRINT "PARA CONTINUAR DIGITE CONT"
115  PRINT "E TECLE RETURN"
```

```
120  STOP
130  GOTO 10
```

EXERCICIOS

1. Fazer o nome piscar no meio da tela.

```
MEU NOME E

J O S E
```

Fazer piscar o nome JOSE, ou seu nome.

2.

```
E U   S O U   O   M A I O R
* * * * * * * * * * * * * * * * * *
*                                 *
* C   A   R   L   O   S           *
*                                 *
* * * * * * * * * * * * * * * * * *
```

Piscar o nome CARLOS, ou seu nome. As demais posições devem permanecer fixas.

3. Propor um exercício e resolver usando os recursos vistos até este capítulo.

2.3 COMANDO SPC(n)

Este comando imprime determinado número de espaços, de acordo com o parâmetro entre parênteses. Na verdade, o valor de n na instrução SPC(n) indica quantos espaços devem ser impressos.

Exemplo:

```
10 HOME
20 PRINT SPC(10);"MICROCOMPUTADOR"
30 PRINT SPC(15);"MICROCOMPUTADOR"
40 END
```


EXERCICIO

Imprimir o seu nome em colunas variadas, como no exemplo seguinte:

```
JOSE
  JOSE
    JOSE
      JOSE
        JOSE
          JOSE
            JOSE
              .
                .
                  .
```

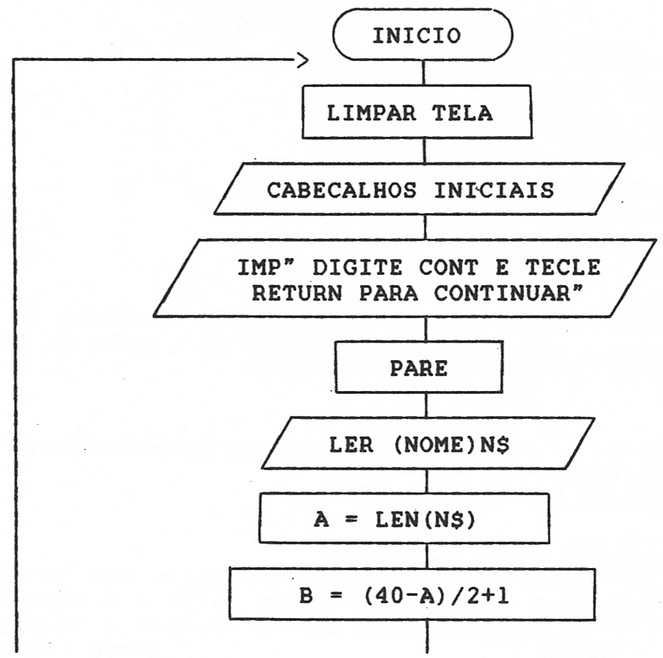
Utilize a instrução SPC(n) e um contador.

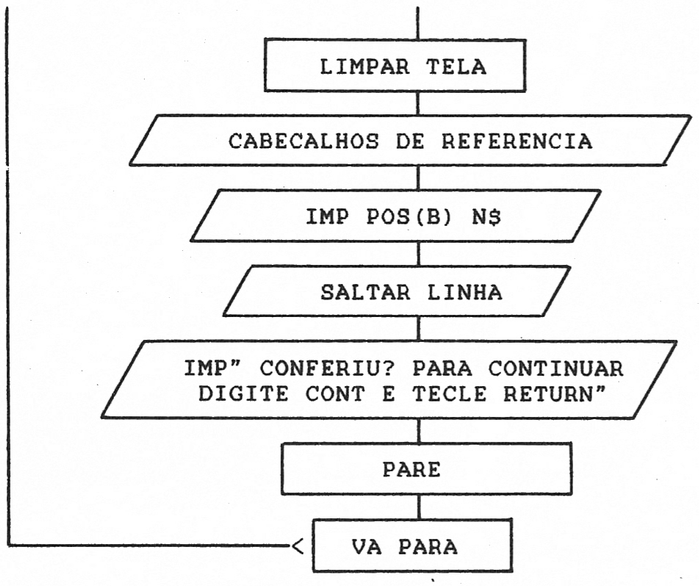
2.4 COMANDO LEN(A$)

Este comando permite calcular o número de caracteres de uma variável alfanumérica. Com isto é fácil centralizar um nome no vídeo. Em cursos mais avançados de BASIC é possível verificar se uma palavra fornecida excedeu certo valor predeterminado. Por exemplo: o nome é um campo com 30 caracteres. Será que certo nome ultrapassou este limite?

Exemplo:

Imprimir nomes centralizados no vídeo.






```
TESTE DE CENTRALIZACAO

        1         2         3         4
1234567890123456789012345678901234567890

NOME
```

```
 10 HOME
 20 B$ = "1234567890"
 30 PRINT "ENTRE COM UM NOME E"
 40 PRINT "VEJA A IMPRESSAO SEMPRE"
 50 PRINT "CENTRALIZADA"
 60 PRINT
 70 PRINT "DIGITE CONT E TECLE RETURN PARA CONTINUAR"
 75 PRINT "RETURN PARA CONTINUAR"
 80 STOP
 90 INPUT "ENTRE COM UM NOME "; N$
100 A = LEN (N$)
110 B=((40-A)/2)+1
120 HOME
130 PRINT TAB(10)"TESTE DE CENTRALIZACAO"
140 VTAB 10
150 HTAB 1
155 PRINT TAB(10);"1";TAB(20);"2";TAB(30);"3";TAB(40);"4"
160 PRINT B$;B$;B$;B$
170 PRINT TAB(B);N$
180 PRINT
190 PRINT "CONFERIU; PARA CONTINUAR"
195 PRINT "DIGITE CONT E TECLE RETURN"
200 STOP
210 GOTO 10
```


EXERCÍCIOS

1. Ler um nome (com até 12 posições) e imprimir dentro de uma moldura como abaixo.

```
************************************************************
*                                                          *
*                  N O M E                                 *
*                                                          *
*                       N O M E                            *
*                                                          *
*                            N O M E                       *
*                                                          *
*                                                          *
************************************************************
```

OBSERVAÇÕES:

- A última letra do nome da linha 1 está na mesma coluna da primeira letra do nome na linha 2.
- A última letra do nome da linha 2 está na mesma coluna da primeira letra do nome da linha 3.

- Fazer o fluxograma.
- Codificar.

2. Cada item do almoxarifado de certa empresa tem sete informações.

N. item	Número das informações
1	1 2 3 4 5 6 7
2	8 9 10 11 12 13 14
3	15 16 17 18 19 20 21
.	. . . .
.	. . . .
.	. . . .
.	. . . .

O problema consiste em, dado o número do item, imprimir a sequência do número de informações associada a este item. Por exemplo:

Para o item 3 , imprimir:

3 15 16 17 18 19 20 21

3. Sabe-se que a terceira informação de cada item do exercício anterior é o saldo atual em estoque. Dado o número do item, imprimir a posição da variável estoque deste item. Veja o exemplo abaixo:

Item	Posicao da variavel saldo
2	10
3	17

4. O custo da tarifa da energia elétrica é função do tipo do consumidor.

Sendo:

Código usuário	Tipo usuário	Custo KW/h
1	Residencial	150
2	Comercial	100
3	Industrial	50

Tendo o usuário de código 1, 2 ou 3 consumido n KW/h, achar o custo de sua conta.

```
CUSTO = CONSUMO * (4 - TIPO) * 50
```

2.5 COMANDO REM

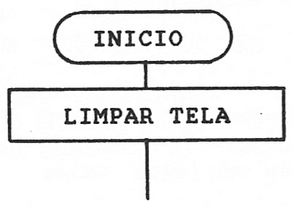
Este comando permite a colocação de mensagens explicativas no programa, mas que não são impressas durante a execução. Somente facilitam a compreensão do programa. São úteis para se compreender cada etapa do programa.

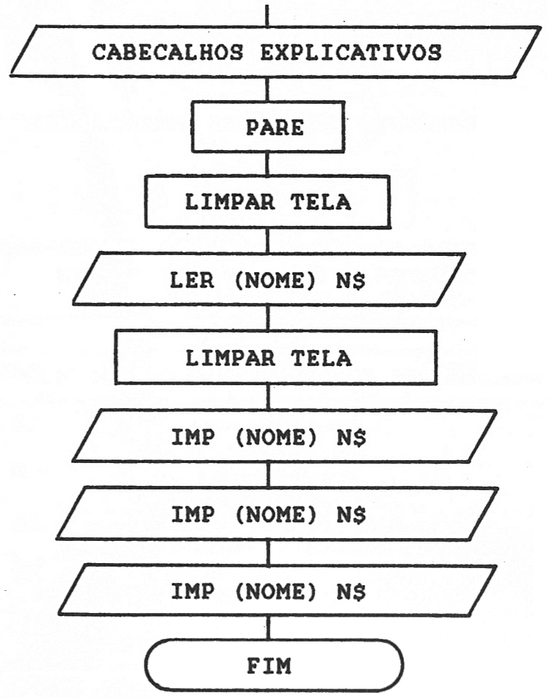
Num problema complexo, é conveniente colocar diversas mensagens com o REM. Isto permite que outras pessoas possam decifrar o significado do programa. Ver o exercício da folha de estoque ao final deste capítulo para melhor esclarecer o que foi dito.

Exemplo:

Imprimir um nome três vezes, mas em posições diferentes.

FLUXOGRAMA





CODIFICAÇÃO

```
 10 HOME
 20 REM COLOCANDO EXPLICACOES SOBRE O PROGRAMA
 30 REM
 40 PRINT "ESTE PROGRAMA IMPRIME 3 VEZES"
 50 PRINT "UM MESMO NOME MAS EM POSICOES DISTINTAS"
 60 REM
 70 REM COLOCANDO UMA PAUSA PARA COMPREENSAO DO TEXTO
 80 REM
 90 PRINT
100 PRINT "DIGITE CONT E TECLE"
105 PRINT "RETURN PARA CONTINUAR"
110 STOP
120 HOME
130 REM
140 REM ARMAZENANDO O NOME A SER IMPRESSO
150 REM
160 PRINT "QUAL O SEU NOME "
170 INPUT N$
180 HOME
190 REM
200 REM IMPRIMINDO O NOME 3 VEZES
210 REM
220 VTAB 10
225 HTAB 1
230 PRINT N$
235 PRINT
240 PRINT SPC(5); N$
245 PRINT
250 PRINT SPC(10); N$
260 REM
```

```
270 REM IMPRIMINDO MENSAGEM FINAL
280 VTAB 23
294 HTAB 4
297 PRINT "PROCESSAMENTO ENCERRADO NORMALMENTE"
300 END
```

EXERCICIOS

1. Imprimir o calendário do mês de junho, com um espaço abaixo dos números para anotações.

SEG	TER	QUA	QUI	SEX	SAB	DOM
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	

2. Fazer um programa para emitir fichas de estoque que contém até quatro itens (ver modelo para dois itens ao final deste capítulo).

FICHA DE ESTOQUE DATA.....

CODIGO	NOME	ESTOQUE	SAIDA 1	SAÍDA 2
_____	____	_____	_____	_____

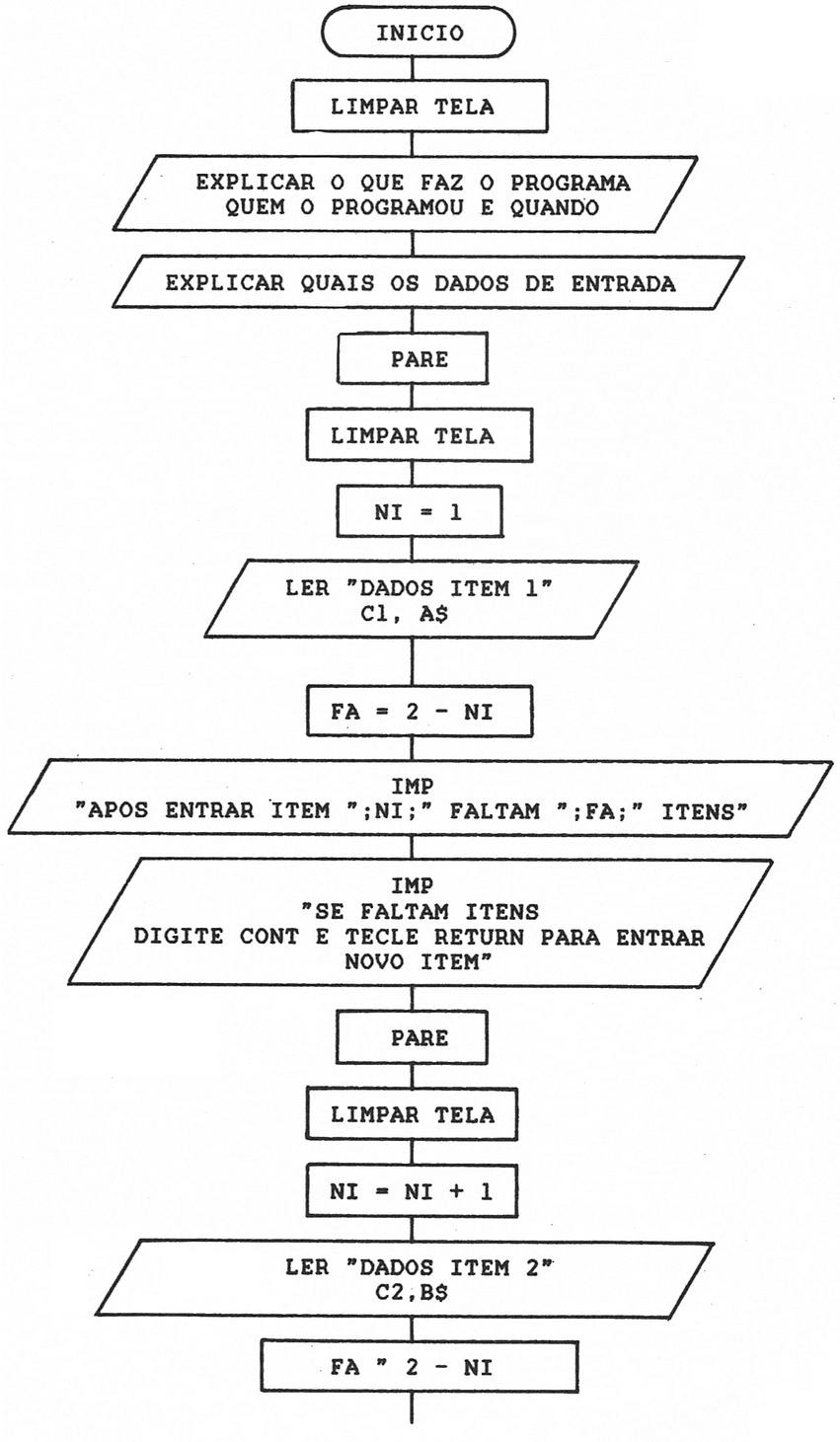
MENSAGENS INICIAIS

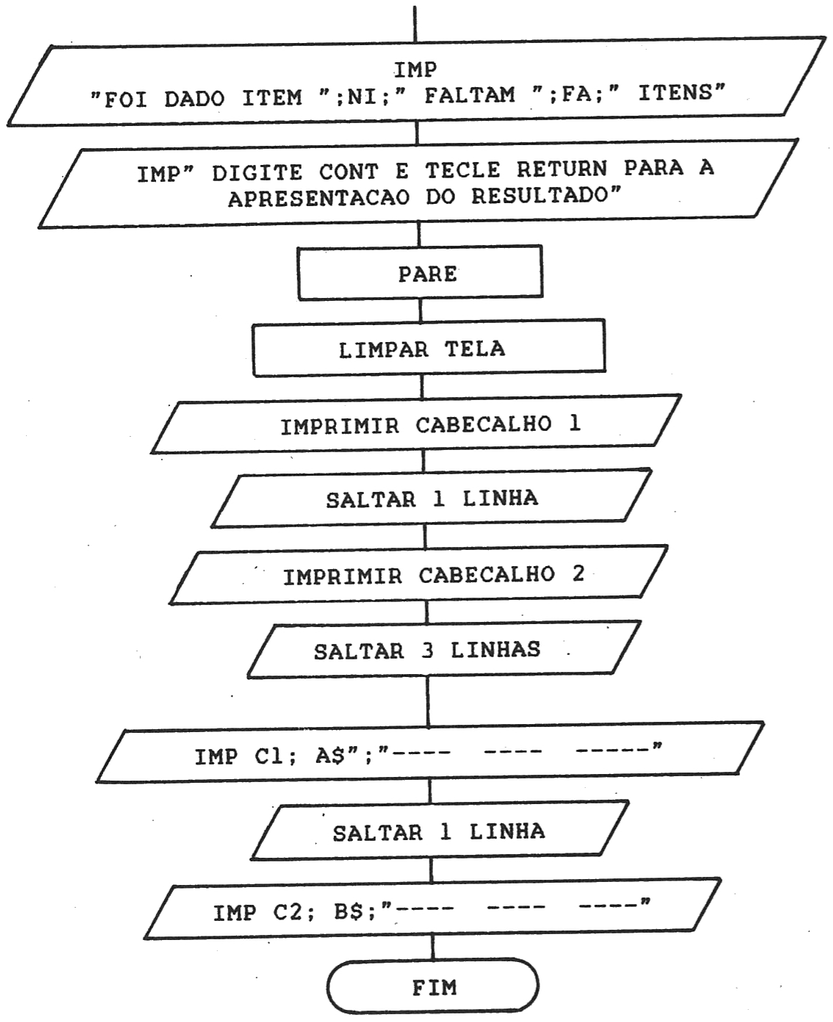
ESTE PROGRAMA PERMITE QUE SE UTILIZE O COMPUTADOR PARA ELABORAR FICHAS DE ESTOQUE.

FOI PROGRAMADO POR.....
EM.....

DEVEM SER FORNECIDOS OS CODIGOS E OS NOMES DE 2 ITENS DE CADA VEZ.

FLUXOGRAMA DO EXERCÍCIO FICHA DE ESTOQUE PARA 2 ITENS:





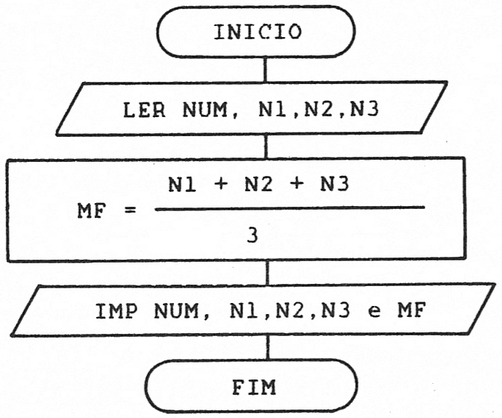
CODIFICAÇAO

```
 10 HOME
 20 REM MENSAGENS EXPLICATIVAS INICIAIS
 30 REM * * * * * * * * * * * * * * * *
 40 PRINT "ESTE PROGRAMA UTILIZA O COMPUTADOR"
 50 PRINT "PARA MONTAR FICHAS DE ESTOQUE"
 60 PRINT
 70 PRINT "FOI ESCRITO POR ANTONIO C. SILVA EM 15/03/84"
 80 PRINT
 90 PRINT "ESTE PROGRAMA PERMITE UM MAXIMO"
100 PRINT "DE DOIS ITENS DE CADA VEZ"
110 PRINT
120 PRINT "APOS LER AS INSTRUCOES DIGITE"
130 PRINT "CONT E TECLE RETURN PARA CONTINUAR"
135 STOP
140 REM * * * * * * * * * * * * * * * * * * * * * * * * *
```

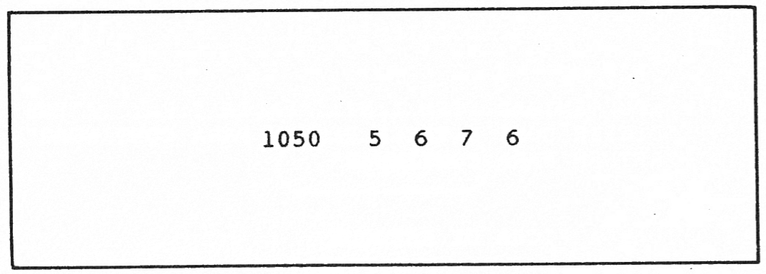


```
150 REM
160 REM FIM DAS MENSAGENS INICIAIS
170 REM
180 REM VARIAVEIS UTILIZADAS NO PROGRAMA
190 REM
200 REM C1, C2  GUARDAM OS CODIGOS DOS ITENS
210 REM A$, B$  GUARDAM OS NOMES DOS ITENS
220 REM NI      GUARDA O NUMERO DE ITENS JA LIDOS
230 REM FA      GUARDA O NUMERO DE ITENS QUE RESTA PARA LER
240 HOME
245 NI = 1
250 REM INICIO DA LEITURA DOS DADOS
260 REM LENDO DADOS ITEM 1
270 RINT "QUAL O CODIGO DO ITEM "; NI
280 INPUT C1
290 PRINT "QUAL O NOME DO ITEM "; NI
300 INPUT A$
310 FA = 2 - NI
320 REM
330 REM SE O NUMERO DE ITENS FOR
340 REM MAIOR QUE 2, SUBSTITUIR O 2.
350 REM NOS COMANDOS 310 E 480 PELO NOVO VALOR.
360 REM
370 PRINT "APOS ENTRAR ITEM "; NI;" FALTA ";FA;" ITEM "
380 PRINT "SE FALTAM ITENS, DIGITE CONT"
385 PRINT "TECLE RETURN PARA ENTRAR NOVO ITEM"
390 STOP
400 HOME
410 REM LENDO DADOS DO ITEM 2
420 REM
430 NI = NI + 1
440 PRINT "QUAL O CODIGO DO ITEM "; NI
450 INPUT C2
460 PRINT "QUAL O NOME DO ITEM "; NI
470 INPUT B$
480 FA = 2 - NI
490 PRINT "APOS ENTRAR ITEM ";NI;" FALTA ";FA;" ITENS"
500 PRINT "DIGITE CONT E TECLE RETURN PARA CONTINUAR"
505 PRINT "RETURN PARA CONTINUAR"
510 STOP
520 REM
530 REM SAIDA DA FICHA DE ESTOQUE
540 REM
550 HOME
560 PRINT TAB(5)"FICHA DE ESTOQUE DATA....."
570 PRINT
580 PRINT "CODIGO  NOME  ESTOQUE  SAIDA1  SAIDA2"
590 PRINT
600 PRINT
610 PRINT C1;" ";A$;" ----  ----  ----"
620 PRINT C2;" ";B$;" ----  ----  ----"
630 REM
640 REM
650 END
```

3. Ler o número e as três notas de um aluno. Calcular sua média e imprimir estes dados.



Se for codificado um programa em BASIC baseado no fluxograma apresentado, o operador do programa ou seu usuário não saberá que valor deverá entrar pelo teclado e terá dificuldade em identificar os valores numéricos que irão aparecer no vídeo.



Para melhorar a comunicação computador-operador é importante colocar mensagens no programa sempre que for solicitado um dado ou for fornecido um valor pelo vídeo. Além disso, é importante que o próprio programa se "apresente", ou seja, qual sua finalidade, por quem foi escrito e quando.

Ao final da execução é interessante dar uma mensagem avisando este fato.

Em resumo, um programa para se relacionar bem com o seu usuário deve conter:

1. Mensagens Iniciais:

   - Qual a sua utilidade.
   - Quem o programou.
   - Quando.
   - Quais os dados necessários para a execução.

2. Mensagens durante a Execução:

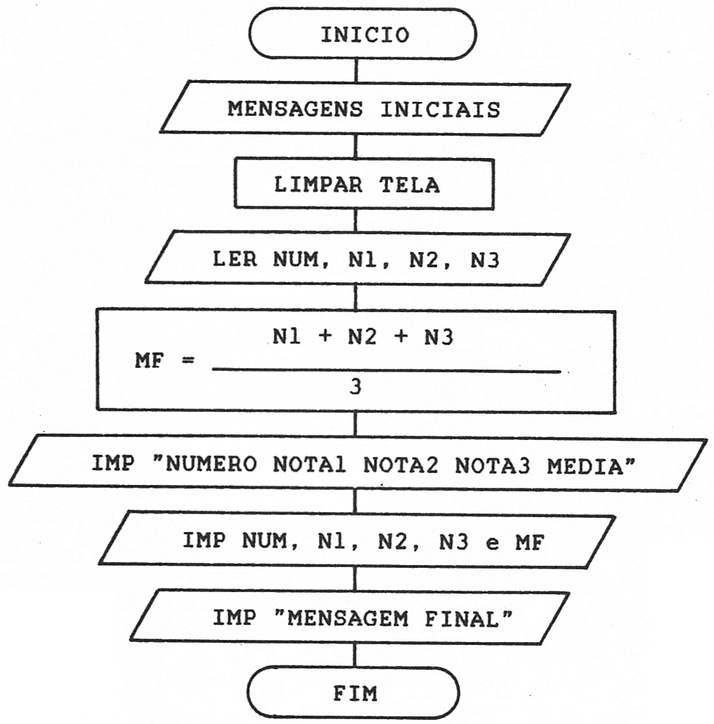
   - Identificar quais são os valores que aparecem no vídeo.

3. Mensagens Finais:

   - Em alguns programas são apresentados resultados estatísticos sobre os dados: número de informações, valores máximos, mínimos, média.

   - Mensagem acusando fim normal do processamento.

O fluxograma assumirá o seguinte aspecto:



Durante a execução do programa teríamos duas telas diferentes:

TELA 1

```
ESTE PROGRAMA CALCULA
A MEDIA DE UM ALUNO
PROGRAMADO POR AIRTON BELUCCI
EM 20-03-85


ATENCAO _ APOS DIGITAR
UM NUMERO, APERTE A
TECLA RETURN
```

TELA 2

```
ENTRE COM O NUMERO DO ALUNO  - 150
ENTRE COM A NOTA1  -  3
ENTRE COM A NOTA2  -  4
ENTRE COM A NOTA3  -  5

NUMERO   NOTA1   NOTA2   NOTA3   MEDIA
 150       3       4       5       4

PROCESSAMENTO ENCERRADO NORMALMENTE
```

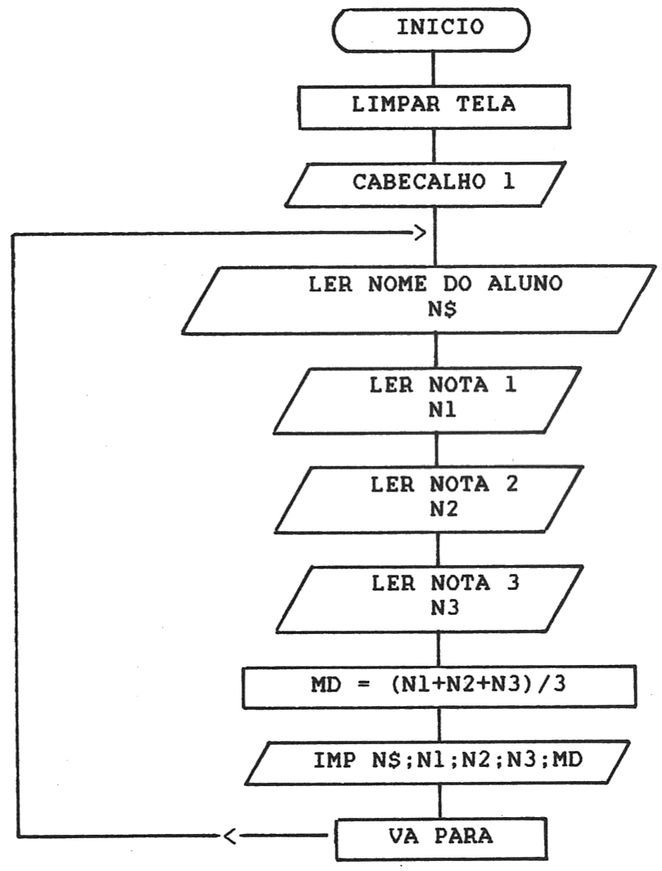
Supondo-se que o aluno seja o número 150 e tenha notas 3, 4 e 5, o resultado seria o apresentado na tela 2.

4. Fazer o fluxograma do seguinte problema: Dado o ano de nascimento e o nome de uma pessoa, calcular a idade da pessoa e apresentar no vídeo.

```
(NOME)     VOCE TEM ---------ANOS
```


5. Calcular a média de cada aluno de uma classe. Cada aluno tem três notas. Apresentar no video as notas e a média.

FLUXOGRAMA



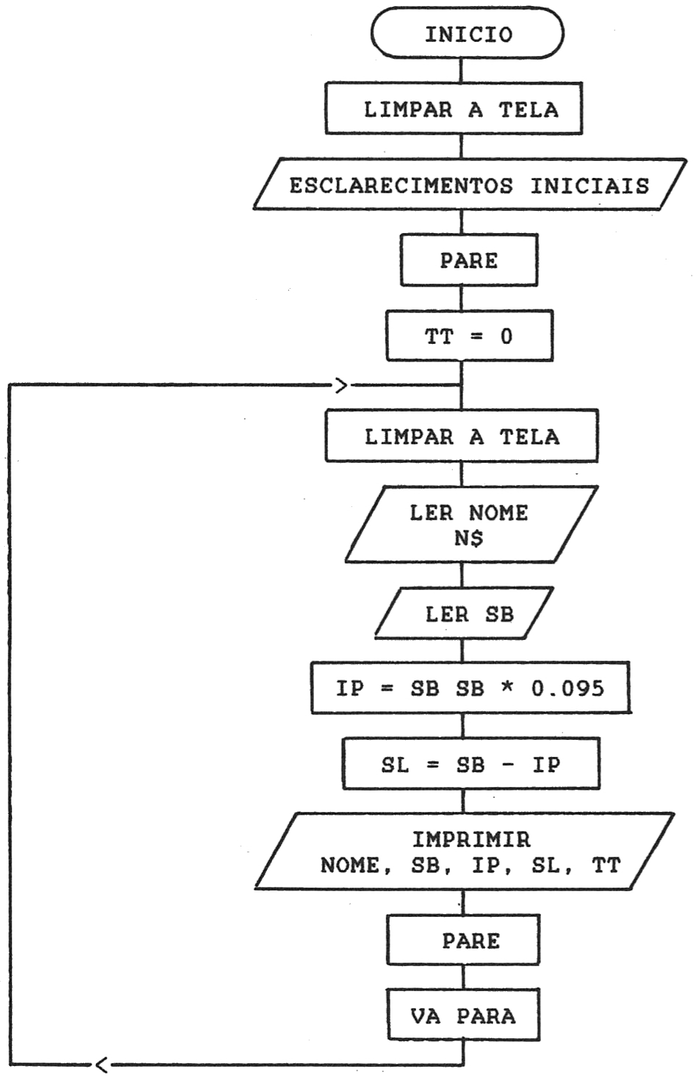
CODIFICAÇÃO

```
  5  HOME
 10  PRINT "PROGRAMA PARA CALCULAR A MEDIA DOS"
 20  PRINT "ALUNOS DE UMA CLASSE"
 30  PRINT
 40  PRINT
 50  INPUT "QUAL O NOME DO ALUNO "; N$
 60  PRINT
 70  INPUT "QUAL A NOTA 1 "; N1
 80  PRINT
 90  INPUT "QUAL A NOTA 2 "; N2
100  PRINT
110  INPUT "QUAL A NOTA 3 "; N3
120  MD = (N1 + N2 + N3)/3
130  PRINT
150  PRINT
```

```
160  PRINT "NOTA1:: ";N1; " NOTA2: ";N2; " NOTA3: ";N3
165  PRINT "MÉDIA FINAL DO ALUNO ";MD
170  GOTO 30
```

6. Calcular o recolhimento do INPS para cada empregado. A alíquota é de 9,5% do salário bruto. Calcular o total de recolhimento de INPS até o empregado que está sendo listado. Imprimir o salário bruto, o recolhimento do INPS, o salário líquido e o total de recolhimento até este empregado.

FLUXOGRAMA




```
 10  HOME
 20  REM APRESENTACAO DOS ESCLARECIMENTOS INICIAIS
 30  REM
 40  REM
 50  PRINT "ESTE PROGRAMA FOI FEITO POR ALBERTO DOS SANTOS"
 60  PRINT
 70  PRINT "FOI PROGRAMADO EM MARCO DE 1985"
 80  PRINT
 90  PRINT "ESTE PROGRAMA CALCULA O RECOLHIMENTO DO"
100  PRINT "INPS POR EMPREGADO, ASSIM COMO APRESENTA"
110  PRINT "O TOTAL RECOLHIDO POR TODOS OS EMPREGA-"
120  PRINT "DOS ATE O EMPREGADO ATUAL"
130  PRINT
140  PRINT
150  PRINT "DIGITE A PALAVRA CONT E APERTE A TECLA"
160  PRINT "RETURN PARA CONTINUAR O PROGRAMA"
170  STOP
180  REM INICIO DO PROGRAMA
190  REM
200  REM ZERANDO TT QUE E O ACUMULADOR DO
210  REM TOTAL DE RECOLHIMENTO DO INPS
220  TT = 0
230  HOME
233  PRINT "AO ENTRAR COM O NOME COMPLETE COM"
236  PRINT "ESPACOS EM BRANCO ATE FORMAR 20"
238  PRINT "POSICOES NO TOTAL"
240  INPUT "ENTRE COM O NOME DO EMPREGADO"; N$
250  PRINT
260  INPUT "ENTRE COM O SALARIO DO EMPREGADO"; SB
270  REM
280  REM LIMPANDO A TELA PARA APRESENTACAO DOS RESULTADOS
290  REM
300  HOME
310  IP = SB * 0.095
320  SL = SB - IP
325  TT = TT % IP
330  PRINT "NOME                          SAL. BRUTO"
335  PRINT
340  PRINT N$;"   ";SB
345  PRINT "INPS     SAL. LIQUIDO      TOT. RECOL"
350  PRINT IP;"    ";SL;"         "; TT
360  PRINT
370  PRINT
380  PRINT "DIGITE A PALAVRA CONT E APERTE A TECLA"
390  PRINT "RETURN PARA PROSSEGUIR"
400  STOP
410  GOTO 230
```

7. Fazer um programa para calcular o total da poupança para cada aplicador ao final de cada mês.

   Dados necessários:

   - Por aplicador
     - Código do aplicador
     - Nome do aplicador
     - Saldo no início do mês
     - Depósito no mês sem direito a correção monetária
   - Correção monetária no mês.

P A R T E II

L I N G U A G E M B A S I C

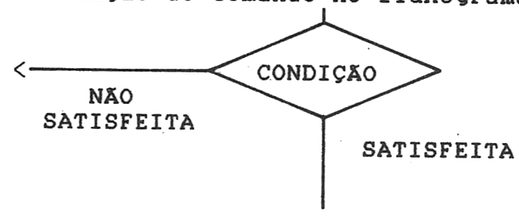
CAPÍTULO 3

COMANDO DE DESVIO CONDICIONAL: IF

A partir deste capítulo verificaremos que todos os exercícios tinham uma estrutura linear, isto é, após executar um comando sempre era executado o comando seguinte, ou seja, o de número de comando imediatamente superior.

Com o comando de desvio condicional, a próxima instrução a ser executada depende da condição imposta.

A representação do comando no fluxograma é feita por meio de um losango.



Se o resultado da condição imposta for afirmativa, a execução é desviada. Caso contrário, prossegue para o comando imediatamente seguinte ao comando condicional.

Exemplo de aplicação do comando:

```
45 IF A > B THEN GOTO 150
46 C = A + B
```

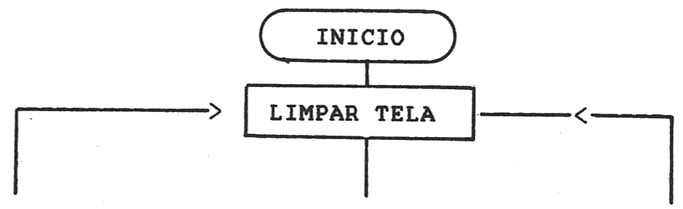
Neste caso, sempre que A for maior que B, a próxima instrução a ser executada é a de número 150. Se A for menor ou igual a B, a próxima instrução a ser executada é a de número 46, que é imediatamente superior a 45.

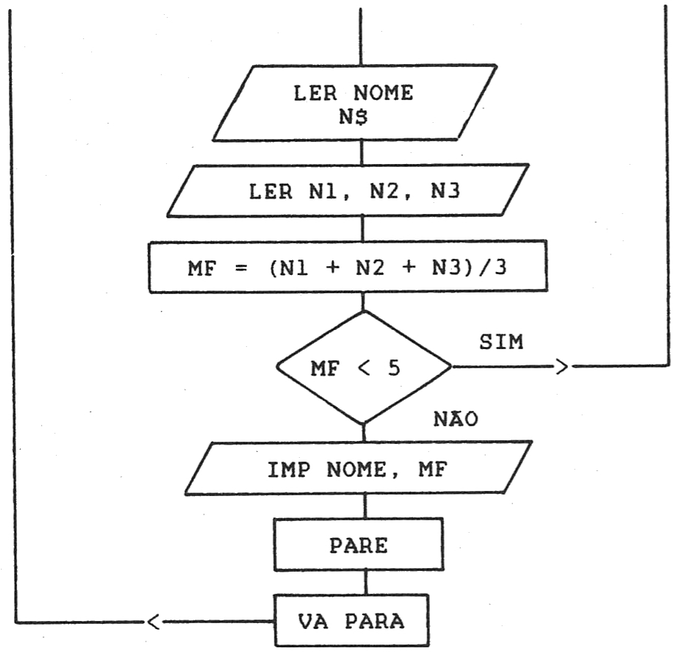
3.1 CONDIÇÕES DE TESTE

```
A > B  -  MAIOR
A < B  -  MENOR
A = B  -  IGUAL
A<> B  _  DIFERENTE
A>= B  -  MAIOR OU IGUAL
A<= B  -  MENOR OU IGUAL
```

Aplicação do comando de desvio condicional num exercício:

Listar os alunos aprovados de uma classe.
Cada aluno tem três notas.
Aprovação significa média maior ou igual a 5.





CODIFICAÇÃO

```
10  HOME
20  INPUT "QUAL O NOME DO ALUNO "; N$
30  INPUT "QUAL A NOTA1? "; N1
40  INPUT "QUAL A NOTA2? "; N2
50  INPUT "QUAL A NOTA3? "; N3
60  MF = (N1 + N2 + N3)/3
70  IF MF < 5 THEN GOTO 10
80  PRINT "ALUNO: "; N$
85  PRINT "APROVADO COM NOTA "; MF
90  PRINT
95  PRINT
100 PRINT "DIGITE CONT E PRESSIONE A TECLA"
105 PRINT "RETURN PARA PROSSEGUIR"
110 STOP
120 GOTO 10
```

3.2 ALTERNATIVAS PARA ENCERRAR AUTOMATICAMENTE UM PROGRAMA

Até este capítulo, os programas com execução cíclica, isto é, as instruções sendo executadas para cada nova entrada não possuíam um fim. Com o comando de desvio condicional é possível planejar o encerramento do programa.

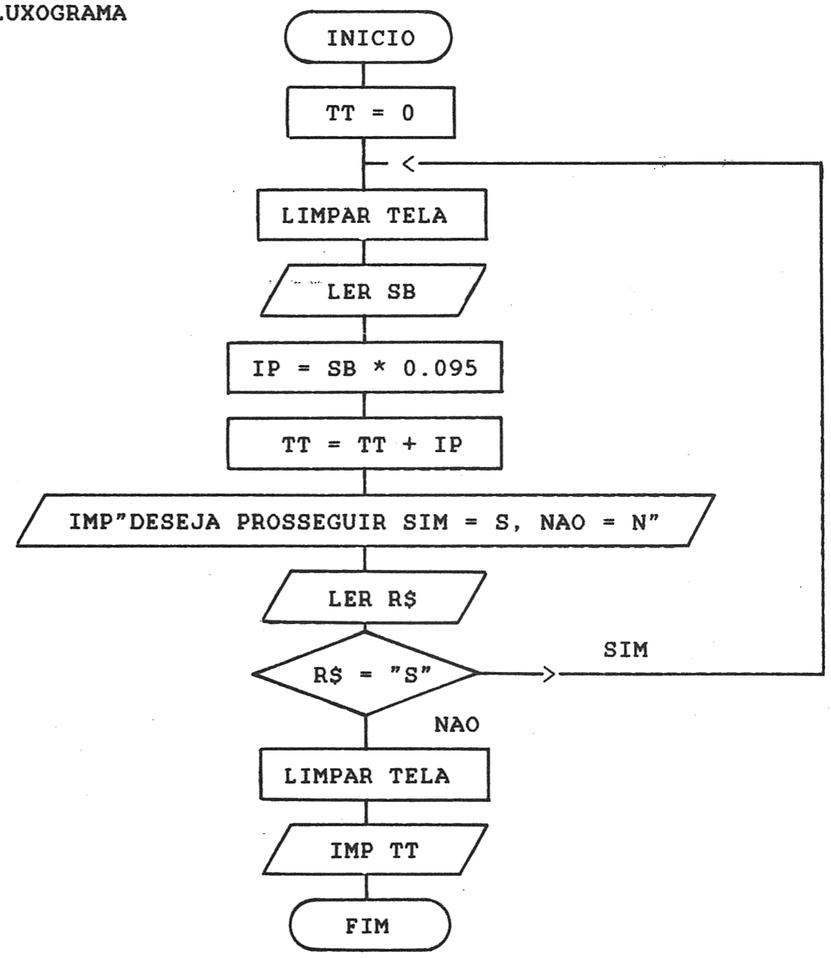
3.2.1 Alternativa 1 - Pergunta Sim/Não

O programa pergunta ao usuário se ele deseja prosseguir. Este responde sim ou não.

Exemplo:

Calcular o total de recolhimento do INPS. Alíquota 9,5%.

FLUXOGRAMA



CODIFICAÇÃO

```
10  TT = 0
20  HOME
30  INPUT "QUAL O SALARIO? "; SB
40  IP = SB * 0.095
50  TT = TT + IP
60  PRINT "DESEJA CONTINUAR"
65  PRINT "DIGITE S PARA SIM OU N PARA NAO"
70  INPUT R$
80  IF R$ = "S" THEN GOTO 20
90  HOME
100 PRINT "TOTAL DE INPS RECOLHIDO = ";TT
110 END
```

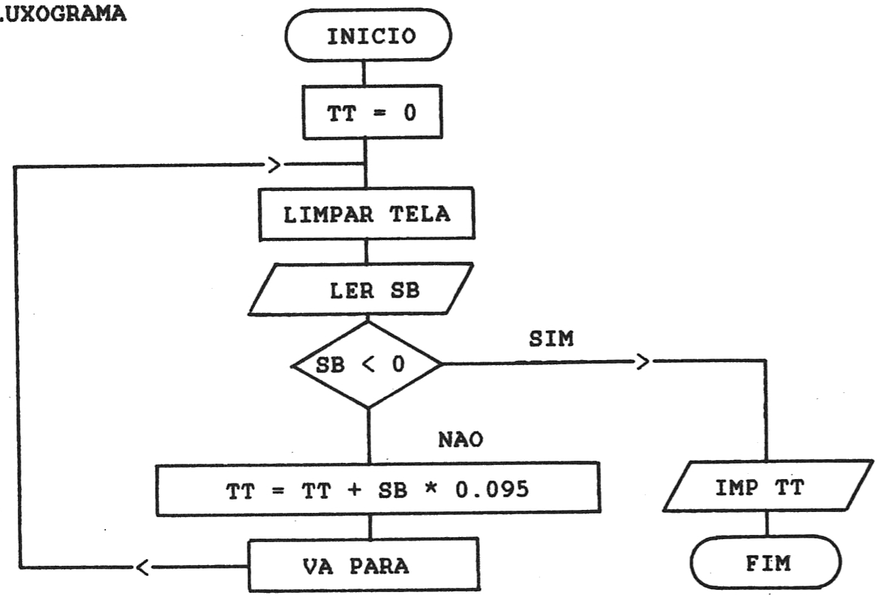
3.2.2 Alternativa 2 - Valor Absurdo

Neste caso, um valor absurdo para os dados do problema deve ser digitado para que o programa pare. Normalmente, um valor negativo é absurdo para a maioria dos problemas.

Exemplo:

Calcular o total de recolhimento do INPS. Alíquota 9,5%.

FLUXOGRAMA



CODIFICAÇÃO

```
10  TT = 0
20  HOME
30  INPUT "QUAL O SALARIO; PARA ENCERRAR  COLOQUE VALOR  NEGA-
    TIVO "; SB
40  IF SB < 0 THEN GOTO 150
50  TT = TT + SB * 0.095
60  GOTO 20
150 PRINT "TOTAL DE INPS RECOLHIDO = "; TT
160 END
```

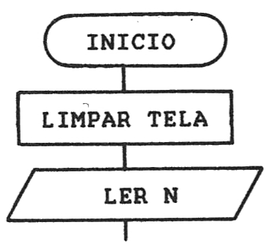
3.2.3 Alternativa 3 - Uso de um Contador

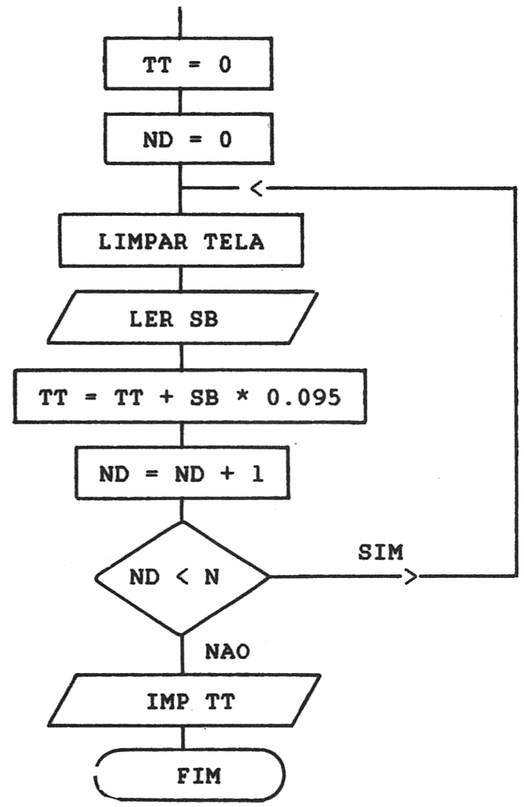
Uma última alternativa para encerrar um programa é através de um contador. O programa irá contando quantos dados foram lidos. Ao final encerra-se automaticamente.

Exemplo:

Calcular o total de recolhimento do INPS. Alíquota 9,5%.

FLUXOGRAMA





CODIFICAÇÃO

```
10  HOME
20  INPUT "QUAL A QUANTIDADE DE SALARIOS? "; N
30  TT = 0
40  ND = 0
50  HOME
60  INPUT "DIGITE O SALARIO BRUTO "; SB
70  TT = TT + SB * 0.095
80  ND = ND+1
90  IF ND < N GOTO 50
100 PRINT "TOTAL DE IMPOSTO RECOLHIDO= "; TT
110 END
```

EXERCÍCIO

Calcular o recolhimento do imposto de renda para cada funcionário.

Se SB < 3 * SM está isento. Caso contrário, a alíquota será de 5% do SB.

SB = Salário Bruto

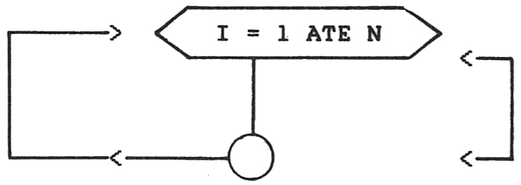
SM = Salário Mínimo (333120)

3.3 COMANDO FOR...NEXT

Com bastante frequência deseja-se executar certa parte de um programa um número limitado de vezes. Este número é controlado por um contador, como foi visto no capítulo anterior, na terceira alternativa para se parar um programa.

Neste capítulo será visto um novo comando que permite realizar de maneira mais simples este controle.

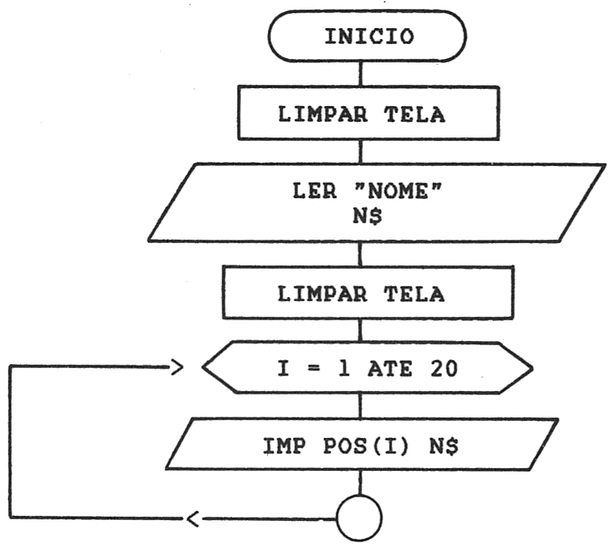
A variável contadora é controlada automaticamente pelo comando FOR...NEXT. A representação em fluxograma deste comando é:



Colocam-se neste espaço as instruções que devem ser repetidas um número N de vezes.

Exemplo:

Imprimir um nome na tela como se fosse um letreiro (correndo). Nome com máximo de 20 letras.



CODIFICAÇÃO

```
10  HOME
20  INPUT "ENTRE COM UM NOME ATE UM MAXIMO DE 20 LETRAS ";N$
30  FOR I = 1 TO 20
40  HOME
50  PRINT TAB(I)N$
60  NEXT I
70  GOTO 30
```


EXERCICIO

Repetir o exercício anterior; de tempos em tempos, o computador perguntará:

QUER ENCERRAR O PROGRAMA?
DIGITAR S PARA SIM E N PARA NÃO.

3.4 VARIAVEIS INDEXADAS - COMANDO DIM

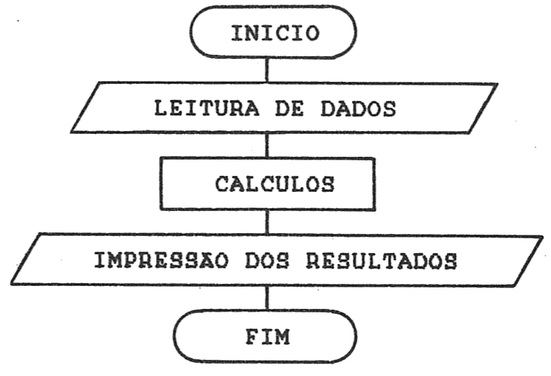
Além das variáveis vistas até agora (numéricas, alfanuméricas), temos um tipo de variável diferente, bastante utilizada quando um ciclo é repetido várias vezes, sendo que devemos guardar os valores das variáveis para cada vez que o ciclo é feito.

Este tipo de variável é constituido de várias posições, sendo que cada posição é indicada por um índice. Daí vem o nome de variável indexada.

MD(1)	MD(2)	....	....	MD(N)

Uma característica importante desta variável é que ela deve ser dimensionada, isto é, o programa deve conter um comando que dimensione a variável, determinando o número de posições que ela vai conter.

Com este tipo de variável podemos escrever, por exemplo, um programa que lê n conjuntos de dados, efetua os cálculos e depois imprime os resultados para todos os conjuntos de dados lidos. Os programas agora poderão ter a seguinte estrutura:



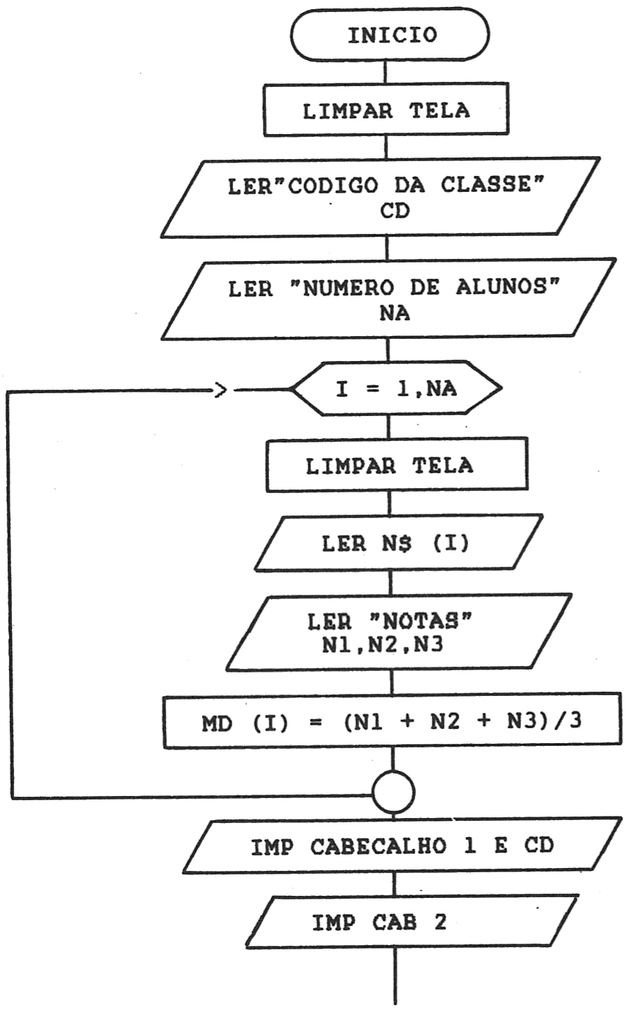
Vejamos como funciona o comando DIM e a variável indexada.

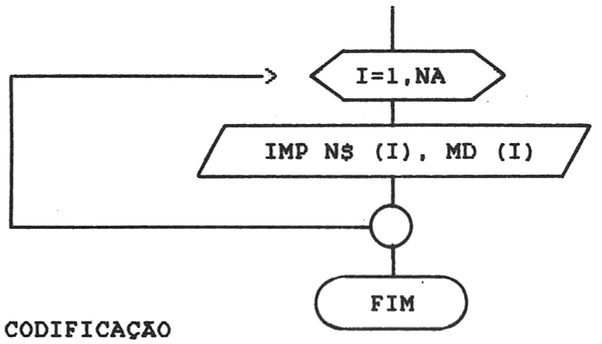
Exemplo:

Calcular a média de cada aluno de uma classe. Imprimir todos os dados ao final da leitura. Cada aluno tem um nome e três notas. A saída deve ser como segue:

```
MEDIAS     DOS     ALUNOS     DO  --------

NOME                          MEDIA

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

----------                    -------
```

FLUXOGRAMA





CODIFICAÇÃO

```
 5  DIM N$(50), MD(50)
10  HOME
20  INPUT "QUAL O CODIGO DA CLASSE?  ";CD
30  PRINT
40  INPUT "QUAL NUMERO DE ALUNOS?  ";NA
50  FOR I = 1 TO NA
60  HOME
70  INPUT "QUAL NOME DO ALUNO?  ";N$(I)
80  INPUT "QUAL A NOTA1?  ";N1
90  INPUT "QUAL A NOTA2?  ";N2
100 INPUT "QUAL A NOTA3?  ";N3
110 MD(I) = (N1 + N2 + N3)/3
120 NEXT I
130 HOME
140 PRINT "MEDIAS DOS ALUNOS DO ";CD
150 PRINT "NOME";TAB(30) "MEDIA"
160 PRINT
170 PRINT
180 FOR I = 1 TO NA
190 PRINT N$(I);TAB(32) MD(I)
200 NEXT I
210 PRINT
220 PRINT
230 PRINT "PROGRAMA ENCERRADO NORMALMENTE"
240 END
```

EXERCICIO

Repetir o exercício anterior, calculando também a média geral da classe, que é a média das médias. A saída no vídeo será:

```
MEDIA     DOS     ALUNOS     DO  ----------

NOME                          MEDIA

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

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

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

MEDIA GERAL                   ----------
```

É possível ter mais de um índice numa variável. Este mesmo exercício poderia ser reescrito como segue:

AL(I,1) - número do aluno I

AL(I,2) - média do aluno I

N$(I) - nome do aluno I

Como se observa, o índice 1 está associado ao número, e o 2, à média.

```
  5  HOME
 10  DIM AL(50,2), N$(50)
 20  REM ESTE PROGRAMA TRABALHA COM UM MAXIMO DE 50 ALUNOS
 30  INPUT "QUAL O NUMERO DE ALUNOS?  ";NA
 40  FOR I = 1 TO NA
 50  INPUT "QUAL O NUMERO DO ALUNO?  ";AL(I,1)
 60  INPUT "QUAL O NOME DO ALUNO?  ";N$(I)
 70  INPUT "QUAL A MEDIA DO ALUNO?  ";AL(I,2)
 80  HOME
 90  NEXT I
100  PRINT "NUMERO      NOME      MEDIA"
110  PRINT
120  FOR J=1 TO NA
130  PRINT AL(I,1);TAB(15) N$(I);TAB(30) AL(I,2)
140  NEXT J
150  PRINT
160  PRINT "PROCESSAMENTO ENCERRADO NORMALMENTE"
170  END
```

3.5 RECURSOS ADICIONAIS DO IF

AND significando E
OR significando OU

- READ - DATA

Certos programas impõem condições múltiplas que devem ser examinadas. Isto requer diversos testes com o Comando IF. Para facilitar tal tarefa pode-se usar os Comandos AND e OR.

Exemplos:

a) Só mandar aviso de cobrança se o cliente estiver com saldo negativo E não tiver cheque especial.

b) Só emitir pedido de reposição de estoque se o saldo estiver abaixo de certo limite E não tiver sido feito ainda pedido de reposição.

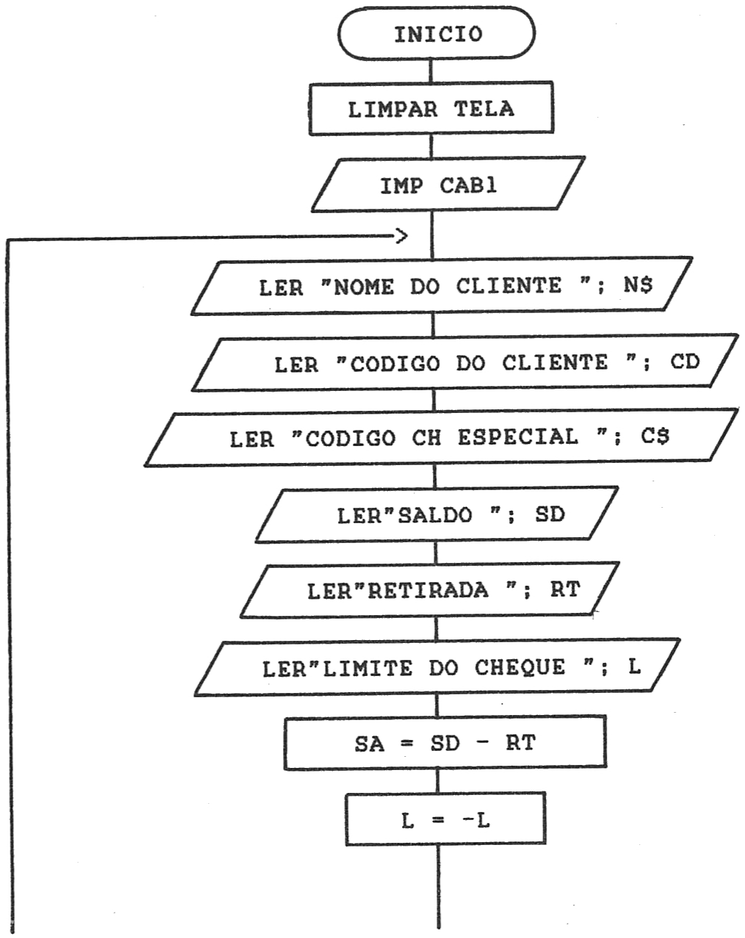
c) Deverão pagar taxa de manutenção mensal de um clube os sócios OU os seus dependentes masculinos maiores de 18 anos.

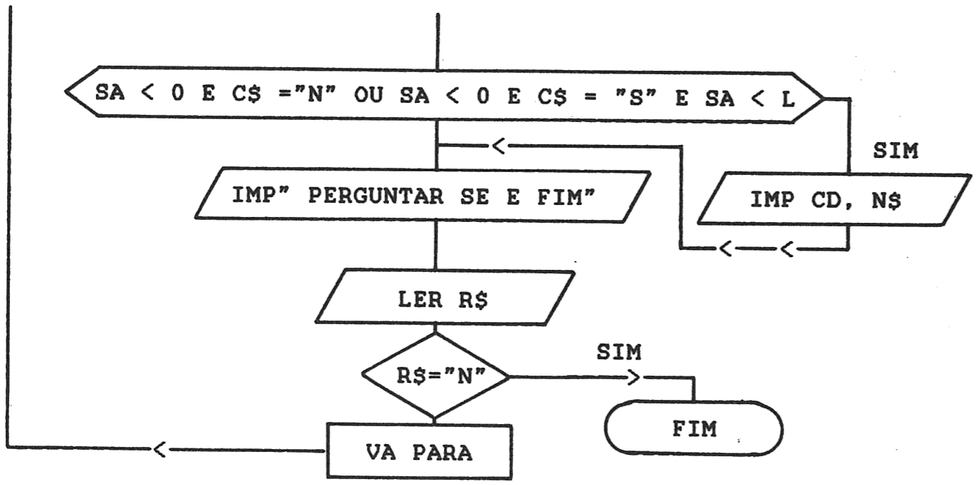
d) Devem declarar imposto de renda aqueles que tiverem renda acima de certo limite OU que possuíam no ano em questão ações no valor acima de X cruzeiros OU possuíam um bem imóvel.

Exemplo:

Emitir um aviso para os clientes que estiverem com saldo negativo E não possuem cheque especial OU estão acima do limite.

Dados - Nome do cliente
- Código do cliente
- Código de cheque especial
  1 - tem cheque especial
  2 - não tem
- Saldo anterior
- Retirada
- Limite do cheque especial





CODIFICAÇAO

```
 10  HOME
 20  PRINT "PROGRAMA PARA VERIFICAR SALDO NEGATIVO"
 30  PRINT
 50  INPUT "QUAL O NOME DO CLIENTE?  "; N$
 60  INPUT "QUAL O CODIGO DO CLIENTE?  "; CD
 70  INPUT "CLIENTE POSSUI CHEQUE ESPECIAL SIM=S, NAO=N "; C$
 80  INPUT "QUAL O SALDO?  "; SD
 90  INPUT "QUAL O SAQUE?  "; RT
100  INPUT "QUAL O LIMITE DO CHEQUE ESPECIAL? SE NAO TIVER  CO-
     LOCAR ZERO - 0 "; L
110  SA = SD - RT
120  L = -L
130  IF SA < 0 AND C$ = "N" OR SA < 0 AND C$ = "S" AND SA  <  L
     THEN GOTO 500
140  INPUT "PARA PROSSEGUIR DIGITE S PARA SIM,   OU N PARA
     NAO ";R$
150  IF R$ = "N" GOTO 400
160  GOTO 50
400  END
500  PRINT "CLIENTE ";  CD;  "  "; N$; " ESTA COM SALDO NEGATI-
     VO  NAO AUTORIZADO"
510  GOTO 140
```

3.6 COMANDO READ

Este comando lê dados armazenados no próprio programa. Normalmente este tipo de dado é permanente ou muda muito pouco no tempo.

Alguns exemplos:

- índice de correção monetária no ano;
- pesos das notas de certa matéria;
- limite do cheque especial;
- número máximo de alunos por classe (número de carteiras).

Exemplo de uso deste comando:

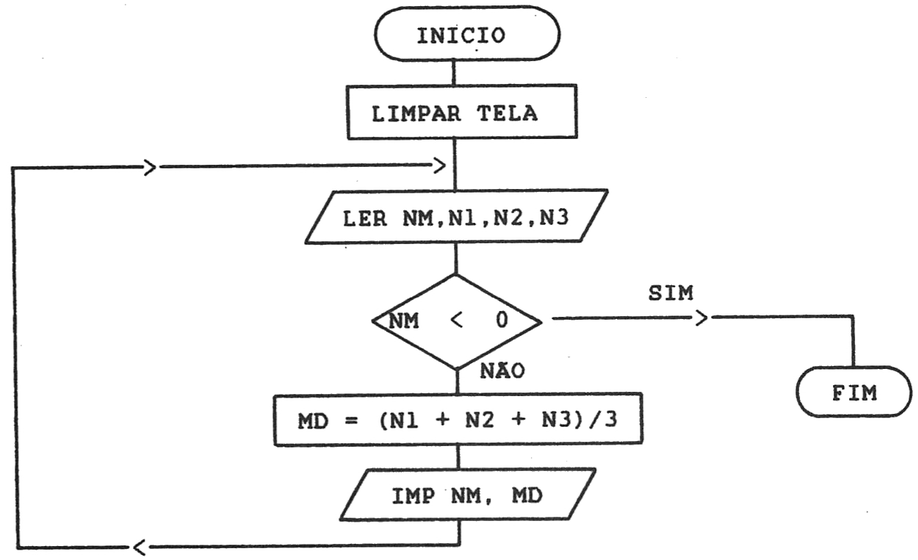
```
READ  A, B, N$
```

A diferença do READ e do INPUT é que o INPUT traz dados do meio externo e o READ lê dados armazenados no programa.

Para ler dados armazenados no programa, o READ precisa do comando DATA que armazena dados.

```
DATA 3, 5, "JOSÉ"
```

Exemplo de uso dos comandos READ E DATA:
Calcular a média de cada aluno da classe.
Dados - número e três notas.



CODIFICAÇAO

```
10  HOME
20  READ NM, N1, N2, N3
30  IF NM < 0  THEN GOTO 500
40  MD = (N1 + N2 + N3)/3
50  PRINT "NUMERO  "; NM; " TEVE MEDIA "; MD
60  GOTO 20
70  DATA 1050,3,4,5,133,7,8,9,225,3.5,5,6,-4,3,7,8
500 END
```

3.7 FUNÇÕES ARITMÉTICAS

3.7.1 Funções

Existem certos procedimentos que são muito usados, por exemplo calcular uma raiz quadrada, encontrar o logaritmo de um número etc. Um programa em BASIC para fazer estes procedimentos é razoavelmente trabalhoso, dependendo da função desejada. Desta forma já foi incorporado à linguagem BASIC estes procedimentos, os quais foram denominados FUNÇÕES.

O seu uso é extremamente simples como se verá a seguir:

	FUNÇAO	SIGNIFICADO
1	SQR(X)	EXTRAI A RAIZ QUADRADA DE X
2	INT(X)	TOMA SO A PARTE INTEIRA DE X,DESPREZANDO OS ALGARISMOS DEPOIS DA VIRGULA
3	LOG(X)	CALCULA O LOGARITMO DE X NA BASE e
4	RND(1)	APRESENTA UM VALOR ALEATORIO ENTRE 0 E 0.999999999
5	ABS(X)	APRESENTA O VALOR ABSOLUTO DE X
6	SIN(X)	CALCULA O SENO DE X
7	COS(X)	CALCULA O COSSENO DE X
8	TAN(X)	CALCULA A TANGENTE DE X
9	GET X$	ESTA FUNÇAO CONTEM O VALOR DA ULTIMA TECLA DIGITADA

3.7.2 Comentários sobre o uso das Funções

1. Se X = 16 ---> A = SQR(X) = SQR(16) = 4
   Logo, se X = 16 A = SQR(X) terá o valor 4 que é a raiz quadrada de 16.

2. Se X = 16 ---> B = INT(X) = INT(16) = 16
   Se X = 16.54 ---> B = INT(X) = INT(16.54) = 16
   Como se observa, a função INT(X) abandona a parte fracionária de X.

3. Esta função calcula o logaritmo de X na base e ou base natural. Normalmente se trabalha com a base 10 ou base decimal.
   A transformação de uma base para a outra é muito simples:
   LOG X =(LOGx)*(0.4342945)
   10

4. A função RND(1) apresenta um valor aleatório inteiro entre 0 e 0.999999999

```
10 A = RND(1)
20 PRINT A
30 GOTO 10
```


O resultado deste programa será totalmente imprevisível,pois a cada vez A terá um valor aleatório entre 0 e 0.999999999. Pode sair impresso por exemplo:

```
0.3527931
0.051
0.88183526
etc.
```

5. A função ABS(X) elimina o sinal negativo de X
   Se X = -5⟶ A = ABS(X) = ABS(-5) = 5
   Se X = 5⟶ A = ABS(X) = ABS(5) = 5

6. a 8. São funções trigonométricas. São usadas normalmente em cálculos científicos.
   Para uso leigo, são usadas em jogos eletrônicos. Será apresentado um exercício mostrando este tipo de aplicação.

9. GET X$ Esta função efetua uma parada no programa e associa a tecla que for pressionada à variável X$. O programa não prossegue enquanto não for pressionada alguma tecla. A variável associada à função GET deve ser sempre alfanumérica.

O exercício a seguir apresenta o exemplo de uso das funções - RND (1), INT(X) E GET

Para quem tem um microcomputador em casa, este exercício fará um enorme sucesso num dia de festa.

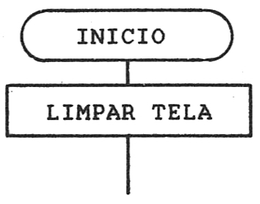
Vocês já devem ter visto ou ouvido falar do Realejo. É um instrumento musical que tem um som característico e é sempre acompanhado de um periquito que, ao som do Realejo, seleciona aleatoriamente um envelope de uma caixa. Este envelope contém algum tipo de comentário. Diz-se que é a "sorte" da pessoa.

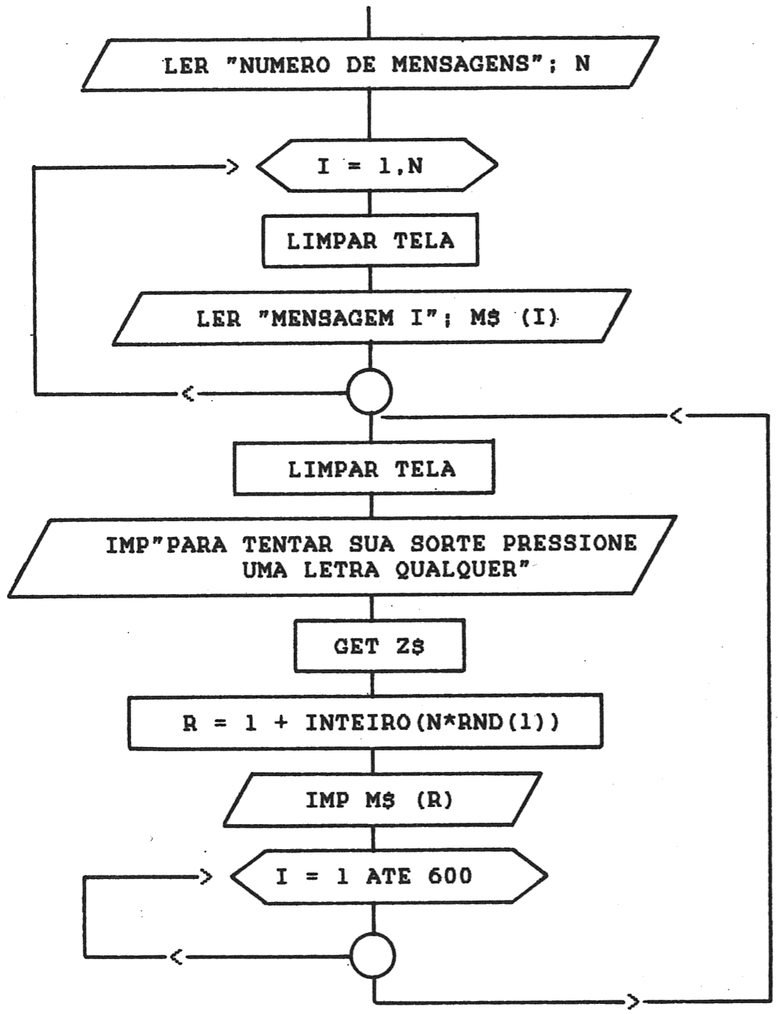
Este programa foi batizado como:

PERIQUITO ELETRONICO, porque seleciona aleatoriamente uma mensagem (previamente armazenada na memória do computador) e a apresenta no vídeo, toda vez que alguém pressionar qualquer tecla do computador.

O programa tem basicamente três partes:

PARTE I - Armazenamento das mensagens.
PARTE II - Situação de espera.
PARTE III - Apresentação da mensagem aleatoriamente selecionados.





CODIFICAÇÃO

```
 10  DIM M$(50)
 20  HOME
 30  INPUT "QUANTAS  MENSAGENS QUE IRA ARMAZENAR? ";N
 40  REM
 50  REM PARTET I - ARMAZENANDO AS MENSAGENS
 60  REM
 70  FOR I = 1 TO N
 80  HOME
 90  PRINT "ENTRE COM A MENSAGEM "; I
100  PRINT
110  INPUT M$ (I)
120  NEXT I
130  REM FIM DA PARTE I
140  REM
150  REM PARTE II, SITUACAO DE ESPERA
160  REM
165  VTAB 12
170  HTAB 10
175  PRINT "QUER TENTAR SUA SORTE?"
180  PRINT
190  PRINT TAB(8)"APERTE UMA LETRA QUALQUER"
195  GET Z$
200  REM
210  REM FIM DA PARTE II
220  REM
230  REM INICIO DA PARTE III, IMPRESSAO DA MENSAGEM
240  REM
250  REM O USO DO RND (1) GERAR UM NUMERO ALEATÓRIO
270  R = 1 + INT(N*RND(1))
280  REM
290  REM
300  REM
310  PRINT M$ (R)
320  REM
330  REM FOR NEXT DARA UM TEMPO PARA LEITURA
340  REM QUE PODE SER VARIADO PELO PROGRAMADOR
350  REM
360  FOR I = 1 TO 600
370  NEXT I
380  HOME
390  GOTO 165
```


EXERCÍCIOS

1. Ler conjuntos de valores, constando cada conjunto de um código e um nome.
   Apresentar estes dados no vídeo numa sequência aleatória. É permitido repetir valores.

Exemplo de entrada:	CÓDIGO	NOME
	1	LÁPIS
	2	BORRACHA
	3	CANETA
	.	.
	.	.
	.	.

Se a saída for aleatória, pode ocorrer:

10	Apontador
3	Caneta
5	Giz
3	Caneta

2. Mesmos dados do exercício anterior, mas sem repetição na saída.

3.8 COMANDOS ON GOTO E GOSUB

3.8.1 Comando ON GOTO

Certos problemas dependem de múltiplas situações, o que exigiria múltiplos comandos IF.

Exemplo:

A tarifa de energia elétrica depende do tipo de consumidor:

TIPO		
1	Residencial	- paga CR$ 100 por KwH.
2	Comercial I	- paga CR$ 70 por KwH se consumir até 1000 KwH/mes.
3	Comercial II	- paga CR$ 50 por KwH se consumir acima de 1000 KwH/mes.
4	Pequena Indústria	- paga CR$ 40 por KwH se consumir até 5000 KwH/mes.
5	Grande Indústria	- paga CR$ 20 por KwH se consumir acima de 5000 KwH/mes.

Sem o uso do ON GOTO, a solução seria como segue:

```
  50  IF  T=1  THEN GOTO 1000
  60  IF  T=2  THEN GOTO 2000
  70  IF  T=3  THEN GOTO 3000
  80  IF  T=4  THEN GOTO 4000
  90  IF  T=5  THEN GOTO 5000
1000  KW= 100
1010  GOTO 200
2000  KW=  70
2010  GOTO 200
3000  KW=  50
3010  GOTO 200
4000  KW=  40
4010  GOTO 200
5000  KW=  20
5010  GOTO 200
```

Com o uso do ON GOTO, este comando analisa o valor de uma variável que assume valores entre 1 e N e desvia para a instrução colocada na posição de mesmo valor que esta variável no GOTO.

```
80   ON X GOTO   D1, D2, D3, D4, D5
```

Se X=1 desvia para o endereço D1
Se X=2 desvia para o endereço D2
Se X=3 desvia para o endereço D3
Se X=4 desvia para o endereço D4
Se X=5 desvia para o endereço D5

No nosso exemplo seria:

```
50   ON   T   GOTO 1000, 2000, 3000, 4000, 5000
```

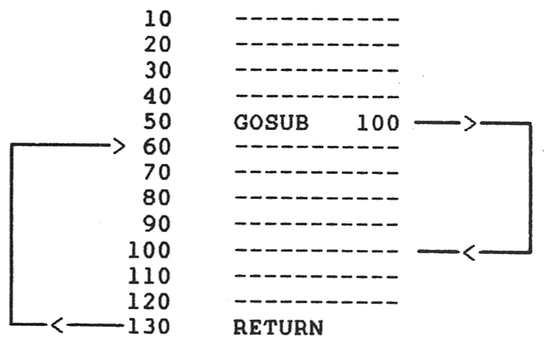
Conseguiu-se condensar todos os testes num único comando.

3.8.2 Comando GOSUB N

Certas partes de um programa são usadas mais do que uma vez, e seria muito trabalhoso ter de repetir as instruções de modo a tornar mais clara a codificação.

Os exemplos citados ocorrem com bastante frequência, existindo um comando BASIC que permite que seja simplificado o uso destas partes do programa muito utilizadas.

A idéia geral do uso do comando GOSUB é apresentada a seguir:



O comando GOSUB N desvia a execução para a instrução localizada em N. No exemplo apresentado, a sub-rotina está no endereço 100. Toda vez que se usar o GOSUB, o final da sub-rotina deve ser sempre o comando RETURN, que significa RETORNE (para o ponto de partida). No nosso exemplo, como o ponto de partida foi a instrução 50, o retorno é automático para a instrução 60.

Neste exemplo, a sub-rotina é composta pelas instruções 100, 110, 120, 130.

Exemplo de aplicação dos comandos vistos até agora:

Calcular o valor a ser pago por consumidores de energia elétrica.

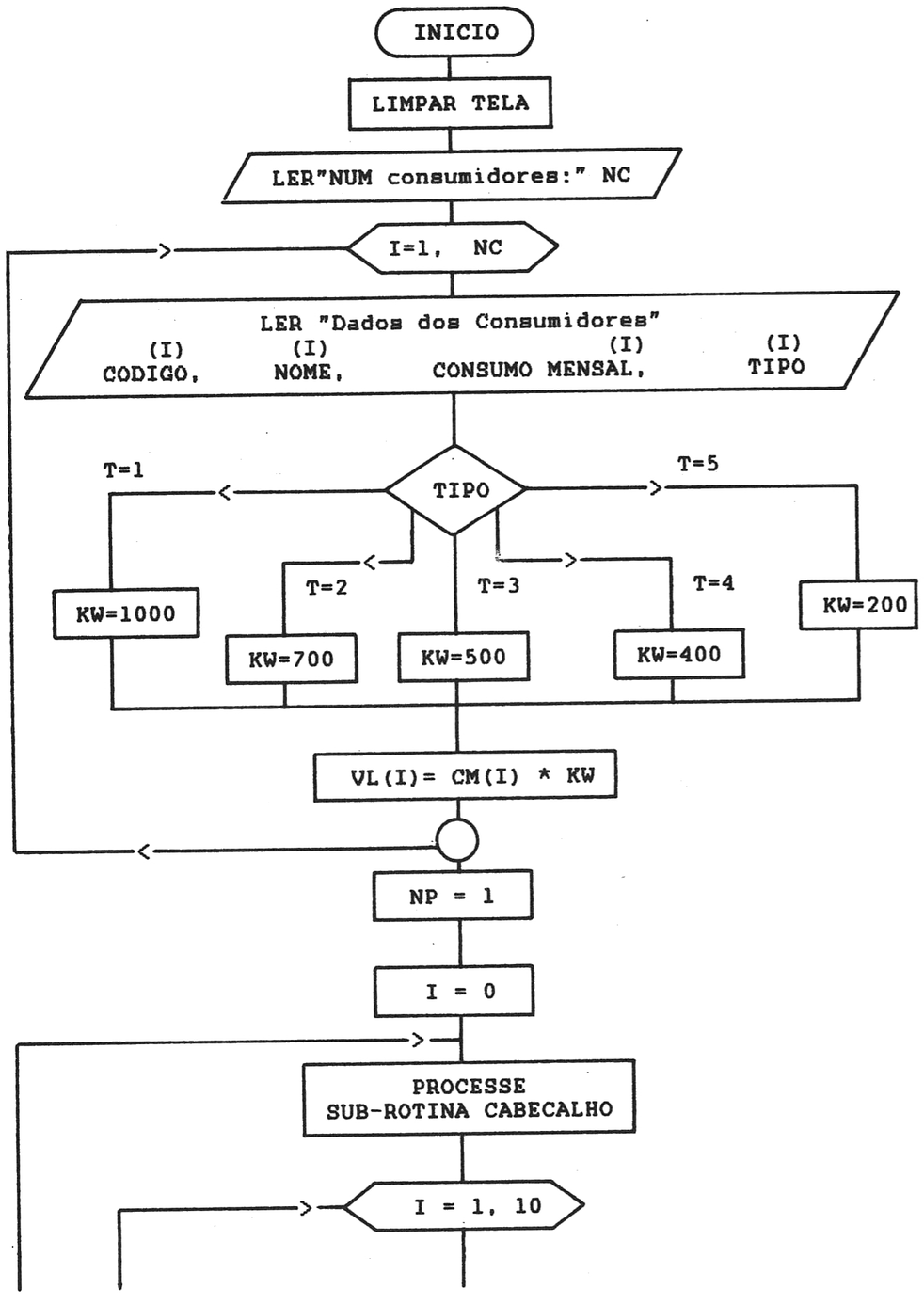
Existem cinco tipos de consumidores e o custo do KwH é diferente para cada um.

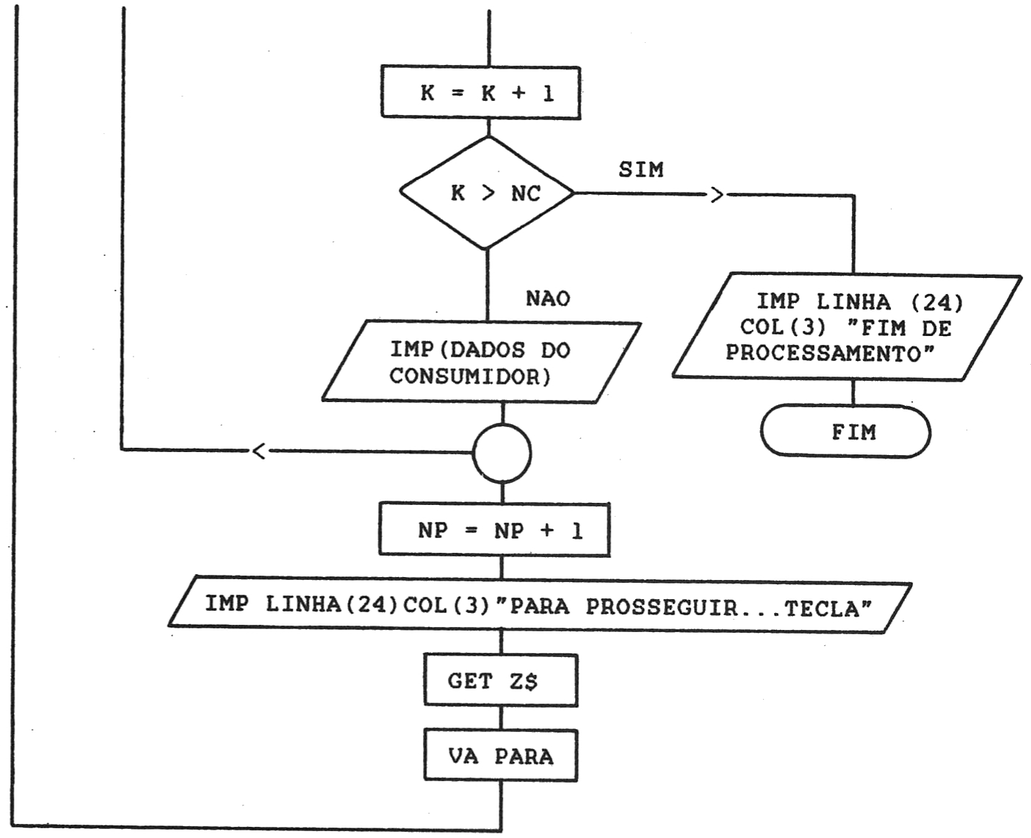
A listagem deve ser numerada, isto é, cada conjunto de 10 consumidores ocupa toda uma tela e no canto da tela aparecerá o número desta página:

```
                                        Pagina N

              CONSUMO DE ENERGIA ELETRICA

     CODIGO    NOME    CONSUMO    TIPO    VALOR
```

As três linhas de cabeçalho constituirão uma sub-rotina.

FLUXOGRAMA





CODIFICAÇÃO

```
 10  DIM  CD(100), N$(100),  CM(100), TP(100)
 20  HOME
 30  INPUT "QUANTOS CONSUMIDORES SERAO LIDOS? MAXIMO DE 100 ";NC
 40  FOR I = 1 TO NC
 45  HOME
 50  INPUT "QUAL O CODIGO DO CONSUMIDOR? "; CD(I)
 60  INPUT "QUAL O NOME  DO CONSUMIDOR? ";N$(I)
 70  INPUT "QUAL O CONSUMO MENSAL? "; CM(I)
 80  INPUT "QUAL O TIPO DE CONSUMIDOR? "; TP(I)
 90  ON TP(I) GOTO 1000, 2000, 3000, 4000, 5000
100  VL (I) = CM(I) * KW
110  NEXT I
120  NP = 1
130  K = 0
140  GOSUB 500
150  FOR I = 1 TO 10
160  K=K+1
170  IF K>NC THEN GOTO 700
180  PRINT CD(K);"  ";N$(K);" ";CM(K);" "TP(K);" ";VL(K)
190  NEXT I
200  NP = NP + 1
210  VTAB 24
```

```
  220  HTAB 3
  230  PRINT "PARA PROSSEGUIR PRESSIONE QUALQUER TECLA"
  240  GET Z$
  250  GOTO 140
  500  HOME
  510  PRINT TAB(38) NP
  520  PRINT
  530  PRINT"CONSUMO DE ENERGIA ELETRICA"
  540  PRINT
  550  PRINT" CODIGO   NOME     CONSUMO   TIPO    VALOR"
  560  PRINT
  570  PRINT
  580  RETURN
  700  VTAB 24
  710  HTAB 3
  720  PRINT"FIM DE PROCESSAMENTO"
  730  END
 1000  KW = 100
 1010  GOTO 100
 2000  KW = 70
 2010  GOTO 100
 3000  KW = 50
 3010  GOTO 100
 4000  KW = 40
 4010  GOTO 100
 5000  KW = 20
 5010  GOTO 100
```

EXERCICIOS

1. Colocar comentários REM no programa.
2. Colocar TAB para impressão dos valores no comando 180.

CAPÍTULO 4

MANIPULAÇÃO DE CARACTERES ALFANUMÉRICOS

LEFT$ (palavra, n)
RIGHT$ (palavra, n)
SOMAR palavras

4.1 COMANDO LEFT$ (palavra, n)

Este comando separa os n primeiros caracteres da palavra especificada. Tanto a palavra como n podem ser variáveis.

```
10  A$ = "ABCDEFG"
20  B$ = LEFT$ (A$, 4)
```

Neste caso B$ = "ABCD".

4.2 COMANDO RIGHT$ (palavra, n)

Separa os n últimos caracteres da palavra.

```
10  A$ = "ABCDEFG"

20  B$ = RIGHT$ (B$, 4)
```

Neste caso B$ = "DEFG".

Como última observação sobre manipulção de caracteres é importante observar que é permitido juntar palavras pela operação de soma.

```
10  A$ = "CURSO"
20  B$ = " DE "
30  C$ " "BASIC"
40  D$ = A$ + B$ + C$
50  PRINT D$
60  END
```

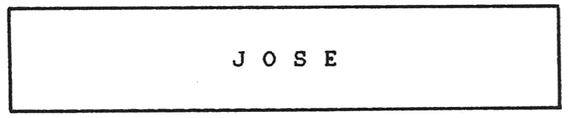
O resultado no vídeo será:

CURSO DE BASIC

Estes comandos, se associados, permitem separar uma palavra nas letras que a compõem.

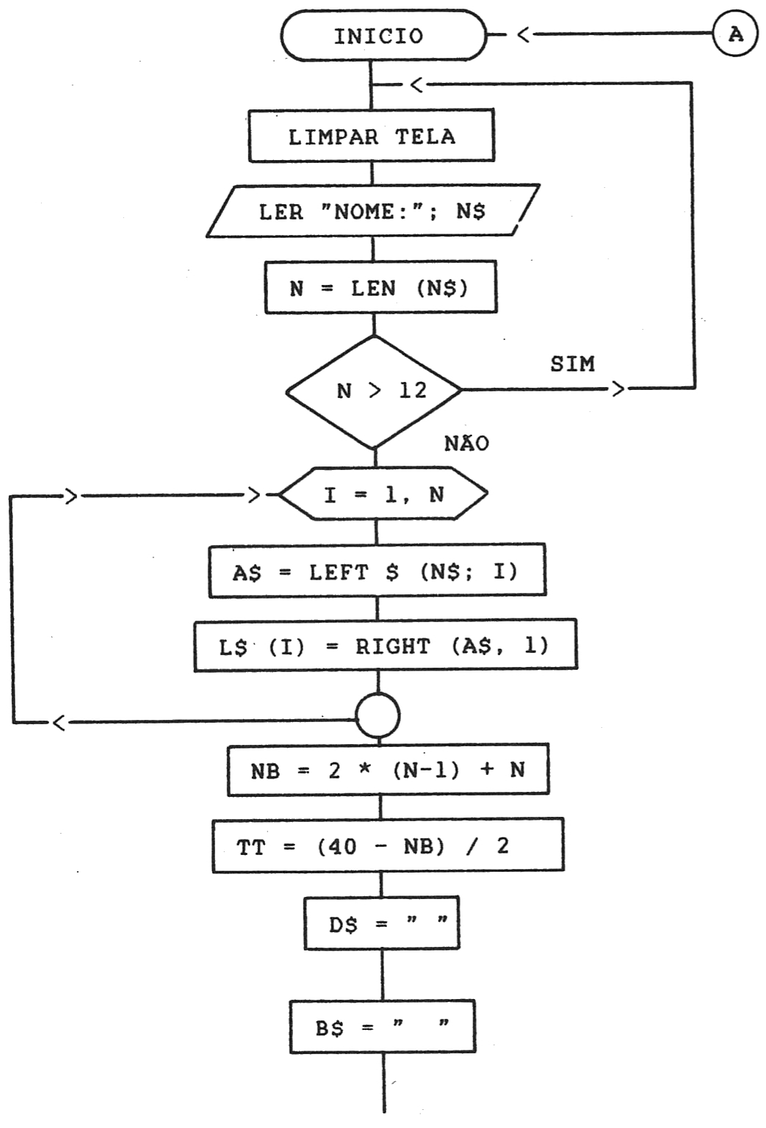
Exemplo:

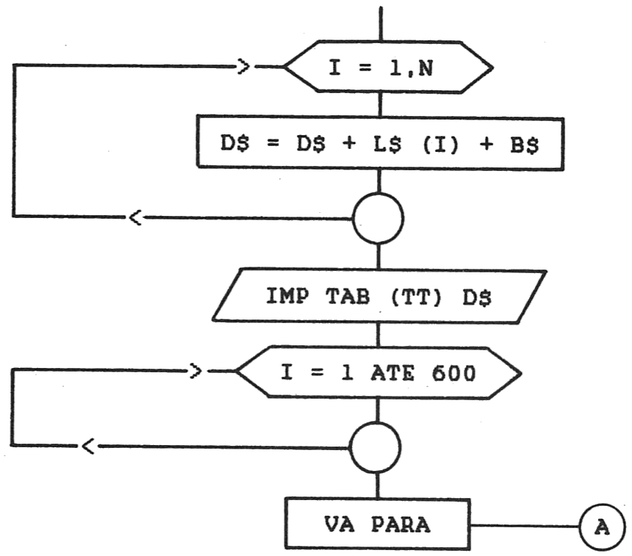
Dado um nome, imprimi-lo com suas letras em destaque co-
o modelo a seguir:



Colocar dois brancos entre letras.

FLUXOGRAMA





CODIFICAÇÃO

```
10  DIM L$ (12)
20  HOME
30  INPUT "ENTRE COM UM NOME DE 12 LETRAS NO MAXIMO "; N$
40  HOME
50  N = LEN (N$)
60  REM
70  REM VERIFICANDO SE NUMERO DE LETRAS E SUPERIOR 12
80  REM SE FOR ACARRETARIA UM ERRO POIS DIM SO PERMITE 12
90  REM
100 IF N> 12 THEN  GOTO  20
110 REM
120 REM OS COMANDOS A SEGUIR VAO SEPARAR ESTE NOME
130 REM NAS LETRAS QUE O COMPOEM
140 REM
150 FOR  I = 1 TO  N
160 A$ = LEFT$ (N$, I )
170 L$ (I) = RIGHT$ (A$,1)
180 NEXT I
190 REM  NB A SEGUIR REPRESENTA O TOTAL DE POSIÇOES
200 REM            OCUPADOS PELO NOME
220 REM
230 NB = 2* (N-1) + N
240 REM  TT A SEGUIR REPRESENTA O TOTAL DE BRANCOS
250 REM            NO INICIO DA PALAVRA
260 REM
270 TT = (40 - NB) / 2.
280 REM  TT  A SEGUIR REPRESENTA O NUMERO DE POSIÇOES
290 REM  INICIAIS DEIXADAS EM BRANCO PARA CENTRAR O
300 REM  TOTAL DE CARACTERES QUE AGORA E  TT
305 TT = TT + 1
310 REM
320 REM  B$  REPRESENTA OS DOIS ESPAÇOS EM BRANCO
330 REM  ENTRE DUAS LETRAS. PARA CADA LETRA
```

```
340  REM  TEREMOS DOIS ESPAÇOS EM BRANCO
350  REM
360  D$ = ""
370  B$ = " "
380  REM
390  REM  A SEGUIR SERA MONTADA A PALAVRA COM
400  REM  LETRA E DOIS BRANCOS - D$ E A PALAVRA
410  REM
420  FOR  I = 1  TO  N
430  D$ = D$ + L$ (I) + B$
440  NEXT I
445  PRINT TAB (TT); D$
450  FOR I = 1  TO  600
460  NEXT I
470  GOTO 20
```

EXERCÍCIO

Alterar um dos dois exercícios apresentados para que de tempos em tempos pergunte ao usuário se este deseja parar o processamento.

CAPÍTULO 5

FUNÇÕES GRÁFICAS: COLOR, GR, PLOT X,Y, TEXT

Para efeitos gráficos, a tela do TK-2000 é composta de 40 linhas, tendo para cada linha 40 pontos

5.1 COMANDOS GR,COLOR e TEXT

Para se proceder ao traçado de um gráfico, é necessário inicializar o modo gráfico. Portanto, antes de plotar (plotar quer dizer colocar um ponto numa posição definida da tela) é preciso utilizar o comando GR, que inicializa o modo GRáfico.

É importante também que se defina uma cor (mesmo que o vídeo seja de fósforo verde) para que os pontos sejam plotados. O comando que define as cores no TK-2000 é o COLOR=X, onde X é um número entre 0 e 15.

O comando TEXT é usado para sair do modo gráfico e retornar ao modo TEXTo.

5.2 COMANDO PLOT X,Y

Este comando acende o ponto do vídeo representado pelas coordenadas X e Y.

A coordenada X varia de 0 a 39.
A coordenada Y varia de 0 a 39.

O ponto (0,0) está localizado no canto superior esquerdo da tela.

O ponto (39,39) no canto inferior direito da tela.

Alguns exemplos de pontos:

(0,0)	(20,0)	(39,0)
(0,20)	(20,20)	(39,20)
(0,39)	(20,39)	(39,39)

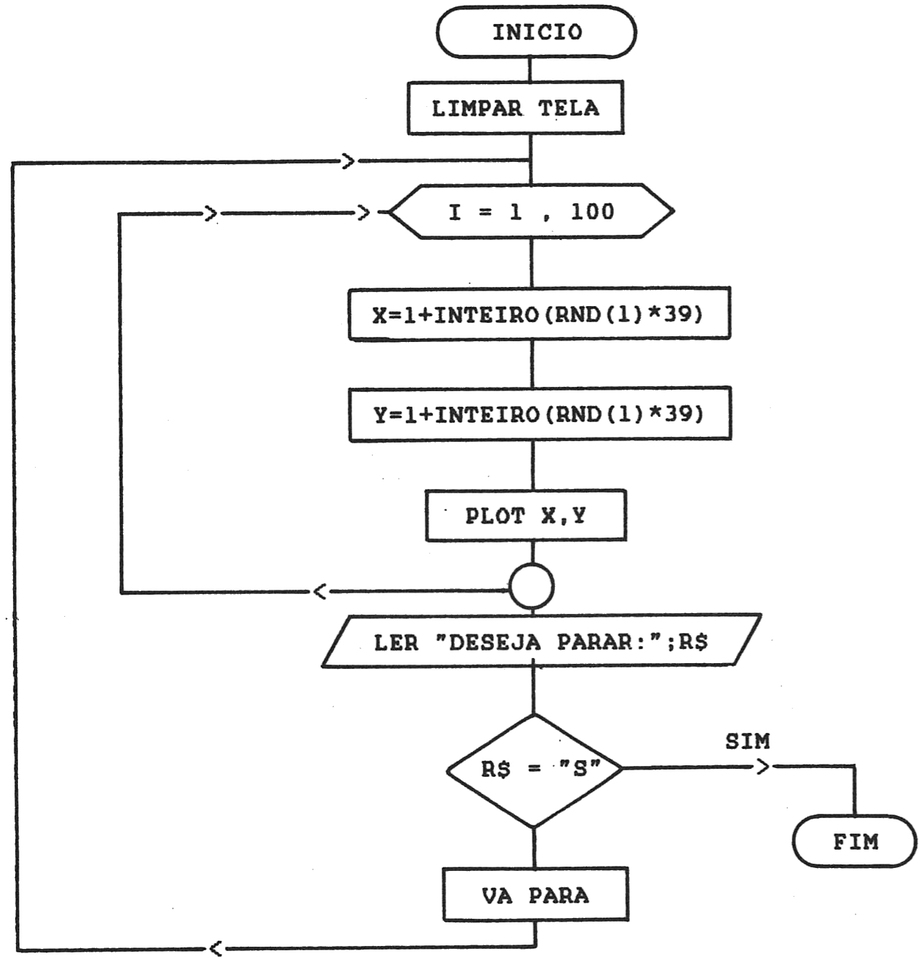
Para exemplificar, vamos fazer o seguinte exercício:

Acender pontos na tela, aleatoriamente.

OBSERVAÇAO: Quando depois de algum tempo o programa perguntar ao usuário se ele deseja parar, a análise dos pontos irá indicar se a sub-rotina de geração de números aleatórios é eficiente.

Os pontos não poderão produzir "manchas" na tela, isto é, produzir aglomerados em certa região.

FLUXOGRAMA

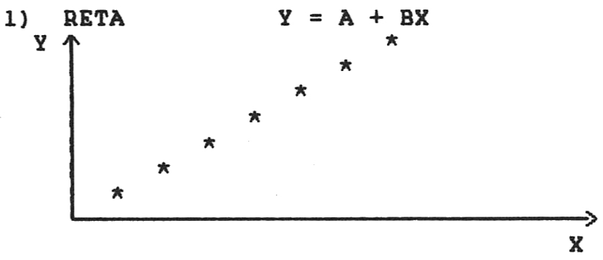


CODIFICAÇAO

```
10  HOME
20  GR
30  COLOR = 5
40  FOR I = 1 TO 100
50  X = 1 + INT(RND(1)*39)
60  Y = 1 + INT(RND(1)*39)
70  PLOT X , Y
80  NEXT I
90  INPUT "DESEJA PARAR (S=SIM, N=NAO)? ";R$
100 IF R$ = "S" THEN END
110 GOTO 40
```

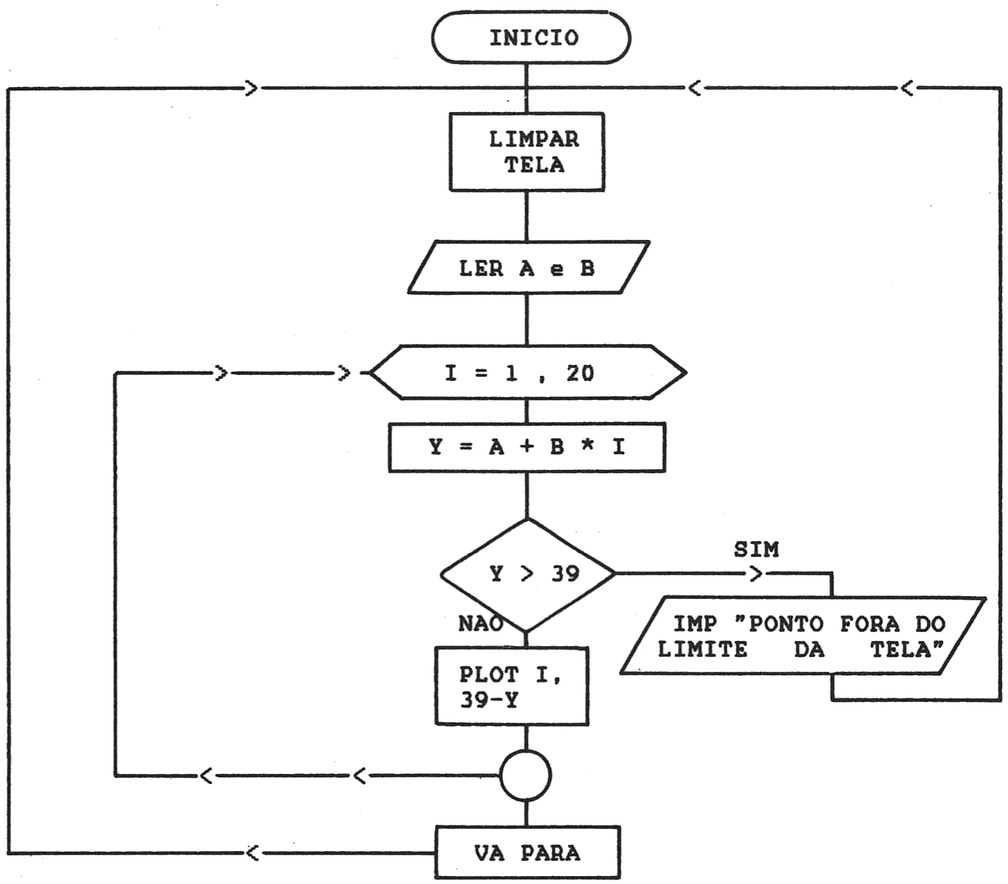

TRAÇADO DE ALGUMAS FUNÇÕES NA TELA

1) RETA Y = A + BX



Vamos traçar a reta com 20 pontos, ou seja, X=1,2,3...20. Os valores de A e B serão fornecidos pelo usuário.

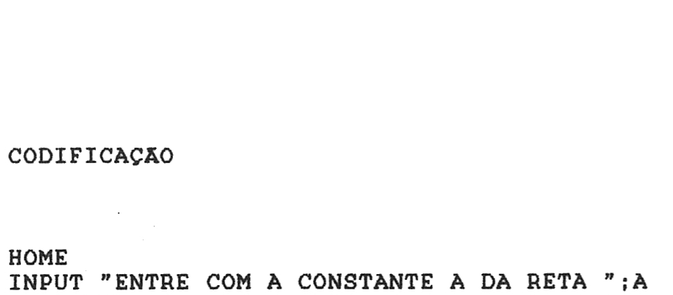
FLUXOGRAMA



CODIFICAÇAO

```
 10  HOME
 20  INPUT "ENTRE COM A CONSTANTE A DA RETA ";A
 30  INPUT "ENTRE COM A CONSTANTE B DA RETA ";B
 35  HOME
 36  GR
 37  COLOR = 5
 40  FOR I = 1 TO 20
 50  Y = A + B * I
 60  IF Y > 39 THEN GOTO 150
 70  PLOT I,39-Y
 80  NEXT I
 90  INPUT "PARA PARAR DIGITE N; PARA CONTINUAR S ";R$
100  IF R$ = "S" THEN GOTO 10
110  END
150  PRINT "PONTO FORA DO LIMITE DA TELA"
160  FOR I = 1 TO 3000
170  NEXT I
178  GOTO 10
```

2) PARÁBOLA $Y = A * X^2 + B * X + C$



CODIFICAÇÃO

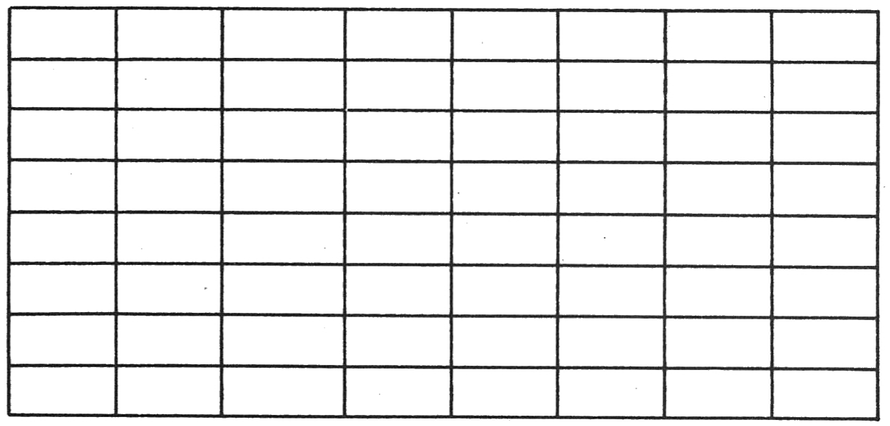
```
 10  HOME
 20  INPUT "ENTRE COM O VALOR A ";A
 30  INPUT "ENTRE COM O VALOR B ";B
 40  INPUT "ENTRE COM O VALOR C ";C
 50  HOME
 53  GR
 56  COLOR = 5
 60  FOR I = -17 TO 17
 70  Y = A * (I * I) + B * I + C
 75  Y = 0.15 * Y
 80  IF Y > 35 THEN GOTO 100
 90  PLOT 20 + I,35 - Y
100  NEXT I
```



```
102  FOR G = 1 TO 3000
104  NEXT G
110  HOME
120  PRINT "PARA CONTINUAR PRESSIONE QUALQUER TECLA"
130  GET Z
140  GOTO 10
```

EXERCÍCIO

Fazer o traçado do seguinte desenho: Um reticulado para tabuleiro de xadrez (oito linhas e oito colunas).



P A R T E III

P R O C E S S A M E N T O D E A R Q U I V O S

CAPÍTULO 6

NOÇÕES BÁSICAS SOBRE SISTEMA OPERACIONAL

O Sistema Operacional é aquele que irá administrar para o programador o acesso aos programas e arquivos em disco magnético, seja na gravação ou na recuperação daquilo que foi gravado.

Tendo-se familiarizado com determinado Sistema Operacional, o processo de adaptação a outro Sistema é muito mais rápido que o aprendizado inicial. Este livro tem por base o Sistema TKDOS do TK 2000 color.

O acesso ao disco pode ser feito em dois níveis:

I - Independente de qualquer programa.

II- Dentro de um programa BASIC.

Neste capítulo serão vistos alguns comandos do primeiro nível. Os do nível II serão examinados nos próximos capítulos.

6.1 ACESSO AO DISCO INDEPENDENTE DE PROGRAMA

Este tipo de comando permite, por exemplo, apagar programas gravados no disco, copiar um arquivo ou programa de um disco para outro. Como se observa são operações independentes de qualquer programa.

6.1.1 Comando DSK

Após inserir o disquete no DRIVE, é preciso avisar o computador para carregar o sistema operacional, o que é feito pelo comando DSK.

Após a execução do comando, aparecerá no vídeo dados sobre o disquete, tais como seu número, alguma data etc. Esses dados foram previamente armazenados no disquete.

6.1.2 Comando CATALOG

Este comando permite saber o que está gravado no disquete.

Aparecerá no vídeo uma relação dos programas e arquivos existentes.

Na frente de cada arquivo aparecerá o código T, indicando um arquivo. Na frente do programa aparecerá o código A.

Exemplo:

```
A   IDENT
A   TESTLEIT
T   ARQLEIT
```

Neste disquete estão gravados dois programas:

- IDENT
- TESTLEIT

e um arquivo:

- ARQLEIT

6.1.3 Comando INIT

Para que um disquete possa ser utilizado para gravação ou leitura, é preciso que o sistema operacional prepare este disquete para uso, o que é feito através do Comando INIT:

INIT IDENT, V

IDENT - É o nome de um programa em BASIC que contém dados sobre o disquete, e possibilitando sua identificação.

V - É o número atribuído a este disquete. Varia de 1 a 254.

O programa IDENT deve conter, por exemplo, a data em que o disquete foi gerado, a quem pertence o disquete, se é utilizado para alguma atividade específica etc.

Exemplo:

INIT IDENT, 5

Programa IDENT:

```
 5  HOME
10  REM  PROGRAMA IDENTIFICADOR DO DISQUETE
20  REM
30  PRINT " DISQUETE NUMERO 5 "
40  PRINT
50  PRINT " INICIALIZADO EM 03-02-85 "
60  PRINT
70  PRINT " ARMAZENA DADOS DO SISTEMA ESTOQUE "
80  END
```

Toda vez que o disquete for colocado no DRIVE, as mensagens do programa IDENT aparecerão no vídeo.

6.1.4 Comando SAVE

Para armazenar um programa no disquete basta usar o comando SAVE.

SAVE TESTLEIT

Após a execução deste comando, o programa TESTLEIT, que deve estar na memória, será armazenado no "drive". Se após este comando for dado o CATALOG, aparecerá o novo programa na lista. O nome de um programa deve começar por uma letra e conter um máximo de oito letras ou dígitos.

6.1.5 Comando LOAD

Para transferir um programa que está no disquete para a memória, basta usar o comando LOAD.

LOAD TESTLEIT

O programa existente na memória será destruído ao ser gravado em cima dele o programa TESLEIT.

6.1.6 Comando DELETE

Para eliminar um programa ou arquivo do disquete, deve-se usar o comando DELETE.

DELETE TESTLEIT

Se for dado um CATALOG após o DELETE, verifica-se que TESTLEIT não aparecerá na lista apresentada.

Após o comando DELETE, o disquete ficará com um espaço disponível para futura utilização.

ARQUIVO 1	ARQUIVO 2		PROGRAMA A		

*
Espaço liberado pelo programa TESTLEIT

6.1.7 Comandos COPIA E TKFID

O sistema operacional TKDOS possui dois programas utilitários, que devem ser transferidos para todos os disquetes gerados devido a sua extrema utilidade.

Logo após o comando INIT, deve-se copiar estes dois programas para o novo disquete.

Estes programas são o COPIA E TKFID.

6.1.8 Utilitário COPIA

Para fazer uma cópia de todo o conteúdo de um disquete, deve ser utilizado o programa COPIA. Será gerada uma cópia fiel do disquete original.

RUN COPIA

Após este comando, o programa instrui o operador para colocar o disquete fonte e em seguida o disquete destino.

6.1.9 Utilitário TKFID

Este programa permite diversos procedimentos extremamente úteis, a maioria destes já existentes no sistema operacional.

Para rodar este programa, é preciso dar o comando:

BRUN BTKFID

É utilizado BRUN e não o usual RUN por tratar-se de um programa em linguagem de máquina e não em BASIC.

O programa tem sete opções:

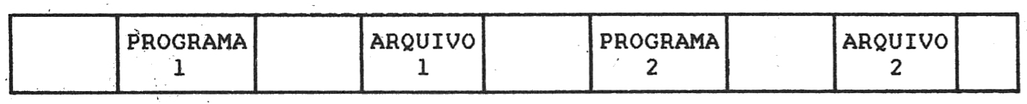
1. COPIA - Copiar ARQUIVOS ou PROGRAMAS.
2. CATALOG - Fornece conteúdo do disquete.
3. ESPAÇO - Fornece o número de BYTES ocupados e os disponíveis no disquete.
4. DESTRAVA - Destrava um arquivo ou programa.
5. TRAVA - Trava um arquivo ou programa.
6. VERIFICA - Verifica a integridade de arquivos ou programas.
7. FIM - Encerra o uso do TKFID.

OPÇÃO 1

A opção 1 permite copiar apenas um programa ou arquivo para outro disquete. Notar que é diferente do programa COPIA o qual copia todo o conteúdo do disquete.

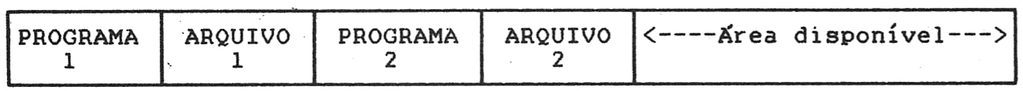
Periodicamente é interessante copiar pelo TKFID todo o conteúdo de um disquete para outro, para melhorar o aproveitamento do disquete.

DISCO ORIGINAL



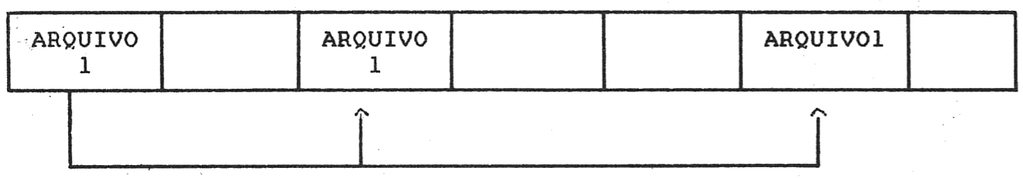
Após a cópia pelo TKFID de todos os arquivos e programas do disquete teremos:

DISCO COPIA



Como se observa, todos os espaços entre programa e/ou arquivos foram suprimidos, restando uma área disponível única, o que facilita o tratamento de arquivos.

Se um arquivo estiver segmentado em vários subarquivos, o tempo de leitura ou gravação é muito maior, pois irá depender do movimento mecânico do cabeçote de leitura/gravação para percorrer todo o arquivo.



OPÇÃO 2

Esta opção permite verificar o espaço disponível no disquete, o que é útil quando se deseja armazenar novos dados ou programas.

Em capítulos futuros será visto como calcular o espaço ocupado por um arquivo.

EXERCÍCIOS

1. Verificar se o programa ou arquivo de nome FOLHAPAG está gravado no disco.

2. O disco em questão está muito cheio ou possui muita área disponível.

3. Desgravar o arquivo anterior DADSPESS.

4. Verificar o conteúdo do disquete.

Preparar um conjunto de instruções para gerar um novo disquete que contém:

- Programa COPIA.

- Programa TKFID.

CAPÍTULO 7

ARQUIVOS

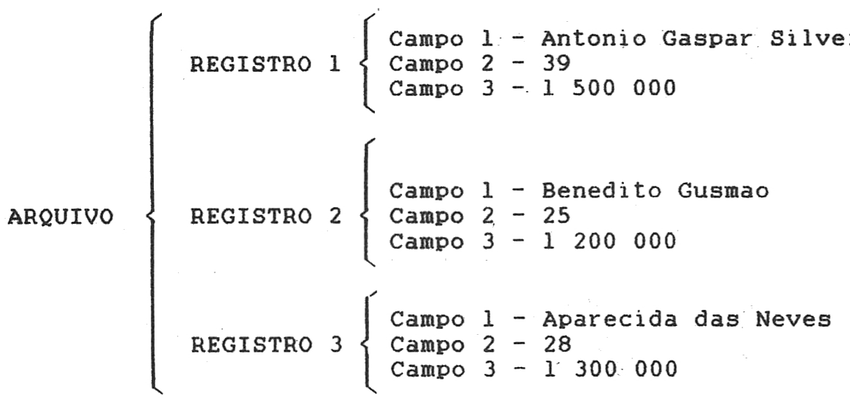
Neste capítulo serão abordados assuntos aparentemente diversos, que terão aplicação no tópico 4, quando será possível especificar o espaço necessário no disco para armazenar um arquivo desejado.

7.1 ARQUIVO - REGISTRO - CAMPO

Para melhor compreender esses três conceitos, examinemos a seguinte situação:

Deseja-se armazenar as informações referentes aos dados pessoais dos funcionários de uma firma. Cada funcionário terá no arquivo seu nome, idade e salário. Com base nestas informações define-se:

ARQUIVO - É o conjunto de todas as informações de todos os funcionários.
REGISTRO - É o conjunto de todas as informações de um funcionário.
CAMPO - É uma informação de um funcionário.



O arquivo que contém os dados pessoais dos funcionários desta empresa será chamado de DADSPESS. Cada registro terá três campos:

CAMPO 1 - NOME
CAMPO 2 - IDADE
CAMPO 3 - SALARIO

7.2 TIPOS DE VARIAVEIS E BYTES OCUPADOS

Para calcular o número de BYTES de um registro, é necessário que se saiba qual o espaço ocupado por todos os campos do registro. Para isto, deve-se saber como é definida cada variável de cada campo que compõe o registro.

Existem quatro tipos de variáveis na linguagem BASIC, as quais serão estudadas a seguir.

7.2.1 Variável STRING

Ocupa um BYTE por caractere. Se o campo **NOME** do arquivo DADSPESS tiver um máximo de 35 letras, isto é, tiver capacidade para um nome com até 35 letras, o exemplo a seguir:

N$ = "ANTONIO GASPAR SILVEIRA"

ocupa 23 BYTES, incluindo-se nesta contagem os brancos entre os nomes. Como foi especificado um máximo de 35, haverá uma sobra de 12 BYTES no caso específico de ANTONIO GASPAR SILVEIRA.

7.2.2 Variável INTEIRA

Ocupa dois BYTES por variável. Os números inteiros que podem ser armazenados neste formato devem estar compreendidos entre -32767 e 32767. Qualquer número fora desta faixa deverá ter uma representação diferente de inteiro.

Identifica-se uma variável inteira pelo símbolo %
Alguns exemplos de números inteiros:

N% = 1 A% = -15300 Z% = 1047

Z% = 152327 Apesar de matematicamente ser um número inteiro, está fora do limite permitido. Não pode ser armazenado com variável inteira.

7.2.3 Variável REAL

Esta especificação ocupa quatro BYTES por variável. Pode armazenar um máximo de sete dígitos significativos. Quando não for especificado o tipo de variável, a linguagem BASIC assume como sendo de precisão simples. Alguns exemplos de números reais.

A = 1234567 B = 54362.98 C = 1234.567

7.3 SUBDIVISÃO DO DISCO EM SETORES

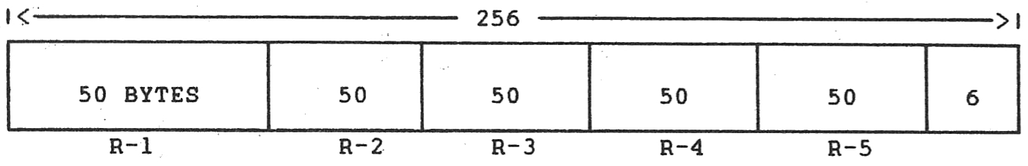
O disco de 5 1/4" de face simples, no Sistema Operacional TKDOS, é composto de 35 trilhas. Cada trilha é subdividida em 16 setores, tendo cada setor 256 BYTES.

A capacidade total de um disco de face simples é calculada a seguir:

256 X 16 X 35 = 143360 BYTES
Bytes p/setor — Setores p/trilha — trilhas

Cada registro físico de um arquivo ocupa um setor, tendo, portanto, 256 BYTES. Os registros definidos pelo usuário poderão ter um máximo de 256 BYTES:

Se um registro definido pelo usuário, chamado de registro lógico, tiver 50 BYTES, num setor ou registro físico serão armazenados cinco registros lógicos. Haverá uma sobra de seis BYTES.



7.4 CALCULO DO ESPAÇO OCUPADO POR UM ARQUIVO

Voltando ao exemplo do arquivo DADSPESS, cada registro ocupa:

REGISTRO	Campo 1 - Nome	35 letras	35 BYTES
	Campo 2 - Idade	Inteiro	2 BYTES
	Campo 3 - Salário	Precisão Dupla	8 BYTES
		TOTAL	45 BYTES

$$\text{N.REGISTROS LÓGICOS POR SETOR} = \frac{256}{45} = 5.688$$

Como o número de REGISTROS LÓGICOS POR SETOR deve ser inteiro, não é possível colocar seis REGISTROS LÓGICOS POR SETOR neste caso, devendo-se restringir a cinco registros por setor.

Se a empresa tem 217 empregados, o número de Registros Físicos necessários será:

$$\text{N. REGISTROS FISICOS} = \frac{217}{5} = 43.4$$

Serão necessários 44 Registros Físicos para arquivar dados de 217 empregados.

EXERCICIOS

1. Projetar um arquivo para conter dados da conta corrente dos clientes de um banco. Para simplificar admite-se que cada cliente terá no seu registro um código, subdividido em número da agência, e um número sequencial; nome do cliente; saldo; indicação de cheque especial.

2. Tomando-se por base o arquivo do item anterior:

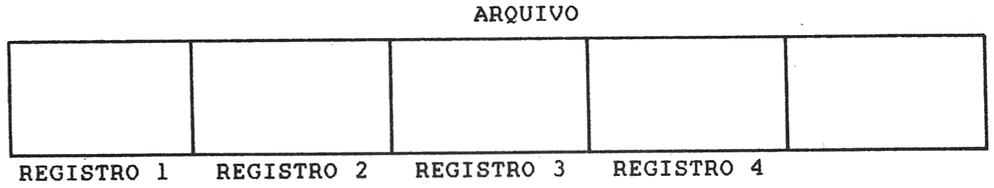
a. Determinar o número de BYTES ocupados pelo registro.

b. Determinar o espaço (em BYTES) necessário para armazenar dados de 100 clientes.

CAPÍTULO 8

MANIPULAÇÃO DE ARQUIVOS SEQUENCIAIS

As informações armazenadas num disquete sobre determinado conjunto de dados são chamadas de arquivos, conforme foi visto no capítulo anterior.



Na manipulação de arquivos, isto é, na gravação de registros ou leitura de informações nele contidas, duas situações podem ocorrer:

a) As informações sempre serão gravadas ou lidas sequencialmente, isto é, após ler o registro 1, deseja-se ler o registro 2, depois o 3 e assim por diante. Este tipo de arquivo é chamado de sequencial, pois os registros serão acessados um após o outro, em sequência. Um arquivo deste tipo é chamado de ARQUIVO DE ACESSO SEQUENCIAL.

b) Outra situação que pode ocorrer é aquela em que nem todos os registros serão lidos num processamento. Quando se deseja as informações do funcionário Aparecido de Oliveira, seria uma perda de tempo acessar todos os registros do arquivo à procura do Aparecido de Oliveira; seria conveniente podermos acessar diretamente o registro deste funcionário. Isto é possível e um arquivo deste tipo é chamado de ARQUIVO DE ACESSO DIRETO.

Neste capítulo será estudado o acesso aos arquivos sequenciais.

Para o tratamento de arquivos sequenciais, os comandos e conceitos a seguir são essenciais:

1. O armazenamento no Sistema TKDOS.
2. Comando OPEN.
3. Comando CLOSE.
4. Comando WRITE.
5. Comando READ.
6. Exemplo.

8.1 O ARMAZENAMENTO NO SISTEMA TKDOS

8.1.1 Introdução

Os arquivos no TKDOS são compostos de registros com uma única variável, não se aplicando o conceito normal de registro, que é um conjunto de campos. Cada campo será um registro físico, ficando por conta do programador controlar a leitura e gravação dos campos.

Para facilitar o entendimento, vamos fazer a seguinte distinção:

Registro Físico - Cada variável armazenada

Registro Lógico - Conjunto de variáveis

Um exemplo torna mais fácil o entendimento do conceito de registro físico e lógico.

Deseja-se criar um arquivo com dados de alunos de uma classe:

REGISTRO LÓGICO	Campo Nome - 30 letras	registro físico 1
	Campo Número - 4 dígitos do Aluno	registro físico 2
	Campo Nota 1 - 4 dígitos	registro físico 3
	Campo Nota 2 - 4 dígitos	registro físico 4
	Campo Nota 3 - 4 dígitos	registro físico 5

O registro que contém dados de um aluno será chamado de REGISTRO LÓGICO. Fisicamente ele não existe. Serão armazenados de maneira independente os cinco registros físicos de cada aluno, que irão compor um Registro Lógico.

No momento da leitura ou gravação, o programa precisa gravar para cada aluno um conjunto de cinco variáveis, sempre na mesma sequência e no mesmo formato, por exemplo, uma variável string (Nome), uma inteira (Número) e três variáveis inteiras (Notas) no caso do registro lógico do aluno.

8.1.2 Comando de Acesso ao Drive

Os comandos de acesso ao disquete no Sistema TKDOS quase sempre terão a forma:

```
PRINT D$ " comando desejado "
```

D$ - é uma variável string que terá sempre o valor:

```
D$ = CHR$(4)
```

O valor CHR$(4) é chave de acesso ao drive, isto é, PRINT CHR$(4) é o código que avisa o computador que será feita uma comunicação com o disquete.

8.2 Comando OPEN

Este comando abre um arquivo para uso, tanto para leitura como para gravação.

Se o arquivo não existir, o comando OPEN cria o arquivo.

```
" OPEN  ARQUIVO "
```

ARQUIVO - é o nome do arquivo a ser aberto ou criado.

Exemplo:

```
10  REM  EXEMPLO DE COMANDOS PARA ACESSO AO DISCO
20  REM
30  D$ = CHR$(4)
40  PRINT D$ "OPEN DADALUNO"
50  END
```

Após executar este programa, estará criado o arquivo DADALUNO, o que pode ser verificado pelo comando CATALOG.

A opção **Espaço** do programa TKFID mostra o espaço ocupado por este arquivo.

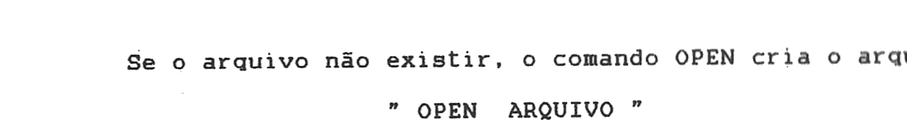
8.3 Comando CLOSE

```
PRINT D$ " CLOSE ARQUIVO "
```

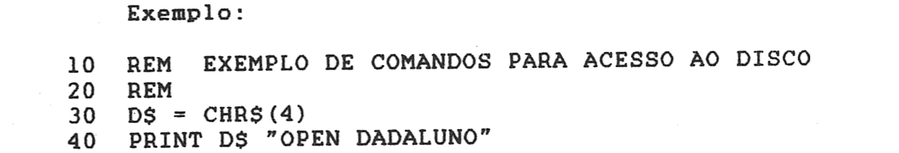
Após o uso de um arquivo ele deve sempre ser fechado, caso contrário pode-se ter situações de erro por um descuido de programação, podendo inclusive destruir outros arquivos gravados.

O Comando CLOSE coloca uma indicação de fim de arquivo após o último registro (EOF - END OF FILE)

Arquivo antes do Comando CLOSE



Arquivo após o Comando CLOSE



O programa anterior com o comando CLOSE será:

```
10  REM  EXEMPLO DE COMANDOS PARA ACESSO AO DISCO
20  REM
30  D$ = CHR$(4)
40  PRINT D$ "OPEN DADALUNO"
50  PRINT D$ "CLOSE DADALUNO"
60  END
```

8.4 Comando WRITE

Este comando libera o arquivo para que a gravação de novos registros possa ser feita.

```
PRINT D$ " WRITE ARQUIVO "
```

Após o comando WRITE, todo PRINT armazenará uma variável no arquivo.

Exemplo:

Gravar dados de um aluno, sendo que para cada aluno tem-se :

a) NOME - N$
b) NÚMERO - ID

```
 10  REM  EXEMPLO DE GRAVAÇÃO SEQUENCIAL
 20  REM
 30  INPUT "QUAL O NOME DO ALUNO"; N$
 40  PRINT
 50  INPUT "QUAL O NÚMERO DO ALUNO"; ID
 60  REM
 70  D$ = CHR$(4)
 80  PRINT  D$ "OPEN DADALUNO"
 90  PRINT  D$ "WRITE DADALUNO"
100  PRINT  N$
110  PRINT  ID
120  REM
130  PRINT  D$ "CLOSE DADALUNO"
140  END
```

É possível abrir num mesmo programa até três arquivos simultaneamente. Para trabalhar com mais de três arquivos, é preciso indicar este fato por um comando a ser visto futuramente.

8.5 Comando READ

Para ler o que foi gravado num arquivo é preciso indicar uma situação de leitura através do comando:

```
PRINT D$ " READ  ARQUIVO "
```

Após o comando READ através do comando INPUT é possível recuperar informações do arquivo.

Exemplo 1:

```
 10  REM EXEMPLO DE LEITURA SEQUENCIAL
 20  REM
 30  D$ = CHR$(4)
 40  PRINT  D$ "OPEN DADALUNO"
 50  PRINT  D$ "READ DADALUNO"
 60  INPUT  N$
 70  INPUT  ID
 80  PRINT  D$ "CLOSE DADALUNO"
 90  PRINT  D$
100  INPUT "DESEJA LER MAIS UM REGISTRO"; R$
110  IF  R$ = "SIM" THEN GOTO 50
120  END
```


Notar que após a leitura foi desativado o comando D$ na linha 90. Isto é necessário sempre que se deseja uma entrada via teclado após a leitura do disquete. Caso contrário, o INPUT será entendido como nova leitura do "drive".

Exemplo 2:

Deseja-se criar um arquivo para armazenar o nome e o RG dos 100 funcionários de uma empresa.

Dimensionamento do arquivo:
NOME - 35 letras, o que equivale a 35 BYTES - N$
RG - 8 dígitos, deve se usada uma variável de precisão dupla ocupando 8 BYTES - RG#

	N$	- 35 BYTES
	RG$	- 8 BYTES
	TOTAL	43 BYTES

$$\text{N. Registros por Setor} = \frac{256}{43} = 5.95$$

consegue-se colocar cinco registros completos por Setor.

Cálculo do número de registros:

$$\text{NREGISTROS} = \frac{100 \text{ FUNCIONÁRIOS}}{5 \text{ REGISTROS POR SETOR}} = 20 \text{ registros.}$$

Serão necessários 20 registros de 256 BYTES cada um.

A - PROGRAMA NA LINGUAGEM BASIC PARA GRAVAR OS DADOS DOS FUNCIONÁRIOS

```
 10  REM "GRVDDFUN"
 20  REM PROGRAMA PARA GRAVAR DADOS DOS
 30  REM FUNCIONARIOS DA EMPRESA NO ARQUIVO
 40  REM "DADSPESS" - MAXIMO DE 100 FUNCIONÁRIOS
 50  REM
 60  D$ = CHR$(4)
 70  PRINT  D$ "OPEN DADSPESS"
 80  FOR  I = 1 TO 100
 90  HOME
100  INPUT "ENTRE COM O NOME DO FUNCIONARIO"; N$
110  INPUT "ENTRE COM O RG DO FUNCIONARIO"; RG#
120  PRINT  D$ "WRITE DADSPESS"
130  PRINT  N$
140  PRINT  RG#
150  PRINT  D$
160  INPUT "NOVO FUNCIONARIO - SIM OU NAO"; R$
170  IF  R$ = "NAO" GOTO 190
180  NEXT  I
190  PRINT  D$ "CLOSE DADPESS"
200  PRINT  D$
210  HOME
```

```
220  PRINT "PROCESSAMENTO ENCERRADO"
230  END
```

B - ARMAZENANDO O PROGRAMA NO DISCO:

SAVE GRVDDFUN

Exemplo 3:

Fazer um programa para listar o arquivo "DADSPESS". Este programa deve ter um cabeçalho com os dizeres a seguir:

N. REGISTRO NOME RG

A - PROGRAMA NA LINGUAGEM BASIC PARA LISTAR O ARQUIVO

```
 10  REM PROGRAMA LSTDDFUN
 20  REM PROGRAMA PARA LISTAR DADOS DOS FUNCIONARIOS
 30  REM DA EMPRESA. USA ARQUIVO DADSPESS
 40  REM
 50  D$ = CHR$(4)
 60  PRINT D$ "OPEN DADSPESS"
 70  HOME
 80  PRINT D$
 90  PRINT " N. REGISTRO                    NOME              RG"
100  PRINT
110  FOR I = 1 TO 100
120  PRINT D$ "READ DADSPESS"
130  INPUT N$, RG#
140  PRINT D$
150  PRINT "     ";I;"        ";N$;"      "RG#
160  NEXT I
170  PRINT D$ "CLOSE DADSPESS"
180  PRINT D$
190  HOME
200  PRINT "PROCESSAMENTO ENCERRADO"
210  END
```

B - ARMAZENANDO O PROGRAMA

SAVE LSTDDFUN

Este programa precisa ter comandos para controle da saída no vídeo, pois a tela comporta 32 linhas, e os nomes aparecerão "correndo no vídeo".

EXERCÍCIOS

1. Fazer alterações no programa de listagem para a saída no vídeo tenha uma pausa, com tempo para leitura dos dados. O usuário indicará quando a listagem deve prosseguir.

2. Alterar o programa para apresentar no vídeo somente o funcionário com número de registro I, isto é, sabendo-se que os funcionários têm número de 1 a 100, o programa pergunta qual o funcionário que deverá ter seus dados no vídeo. O programa deve permitir apresentar vários funcionários no vídeo, um após o outro.

CAPITULO 9

CONCEITOS RELACIONADOS COM ARQUIVOS EM DISCO

9.1 ARQUIVO PERMANENTE

As informações armazenadas no arquivo "DADSPESS", criado no capítulo anterior, devem permanecer no arquivo enquanto o funcionário estiver na empresa. Um arquivo deste tipo é denominado ARQUIVO PERMANENTE. O Arquivo Permanente é o arquivo-base do Sistema.

Os programas que compõem um Sistema podem ter como finalidade alterar o Arquivo Permanente, gravando novos registros, eliminando registros existentes, alterando informações de um registro ou então acessando o arquivo para emitir relatórios.

9.2 ARQUIVO MOVIMENTO

As alterações dos dados de um Arquivo Permanente, normalmente são armazenadas num outro arquivo, transitório, que será desgravado após o uso, isto é, após as alterações terem sido feitas no Arquivo Permanente.

Na rotina de uma empresa, funcionários são admitidos, alguns pedem demissão, outros têm alteração no salário etc.

Essas informações serão armazenadas num arquivo denominado ARQUIVO MOVIMENTO, que será destruído após as informações terem sido transferidas para o Arquivo Permanente.

ARQUIVO PERMANENTE

ANTONIO OLIVEIRA	BENEDITO SANTOS	PEDRO LINHARES	SILVIO BRITO	
		*	**	

O Arquivo Permanente de um Sistema possui informações sobre os funcionários ANTONIO OLIVEIRA, BENEDITO SANTOS, PEDRO LINHARES, SILVIO BRITO E OUTROS.

Deseja-se incluir mais dois funcionários neste arquivo - MARIA DA CONCEIÇÃO E SERGIO PIMENTA. Para manter a ordem alfabética eles devem entrar nas posições assinaladas, respectivamente, por * e **.

As informações dos novos funcionários serão gravadas num arquivo chamado Movimento, o qual será desgravado logo após ter cumprido sua finalidade, que é o armazenamento temporário dos novos funcionários.

ARQUIVO MOVIMENTO

MARIA DA CONCEICAO	SERGIO PIMENTA

Após a execução de um programa que faz a inserção dos novos funcionários no Arquivo Permanente, este terá a seguinte configuração:

ANTONIO OLIVEIRA	BENEDITO DOS SANTOS	MARIA CONCEIÇÃO	PEDRO LINHARES	SERGIO PIMENTA	SILVIO BRITO

9.3 ARQUIVO AUXILIAR

Como o próprio nome sugere, este tipo de arquivo contém informações que irão auxiliar o processamento dos outros arquivos.

Exemplo:

Cada funcionário tem certo cargo dentro da organização. Este cargo tem um nome, o qual deve aparecer nas listagens. Para isto, o nome do cargo deve estar armazenado.

Se este nome for armazenado no registro de cada empregado, haverá um desperdício muito grande da área do disco:

JOSE SILVEIRA, OPERADOR DE TORNO, 30, 800.000
CARLOS ROBERTO, OPERADOR DE TORNO, 28, 900.00

Tendo-se diversos operadores de torno, tem-se uma repetição desnecessária do nome do cargo. Pode-se armazenar os nomes num Arquivo Auxiliar e através de um código fazer a respectiva associação.

"OPERADOR DE TORNO"	ocupa	17 BYTES
Código 10	-	2 BYTES

Se o nome do cargo tiver um máximo de 20 BYTES, reserva-se espaço num Arquivo Auxiliar para nomes com 20 BYTES e 2 BYTES para o código, totalizando 22 BYTES por Registro.

O arquivo DADSPESS deverá ser alterado para conter o código do cargo em lugar do nome do cargo.

ARQUIVO PERMANENTE

JOSE SILVEIRA, 15, 30, 800 000

Registro 1

Em lugar de gastar 20 BYTES para indicar que JOSE SILVEIRA

é OPERADOR DE TORNO, gastaram-se apenas 2 BYTES para indicar que o código é 15.

Será criado o Arquivo Auxiliar:

01, PINTOR	02, SOLDADOR	03, FUNILEIRO	04,
Registro 1	Registro 2	Registro 3	R 4

Na impressão do relatório, o programa deve fazer a associação entre o número 15 e o nome do cargo correspondente.

9.4 EXPANSÃO DE UM ARQUIVO - COMANDO APPEND

Novos funcionários podem ser contratados pela empresa ou novos cargos podem ser criados. Essas alterações devem ser incorporadas aos arquivos já existentes. A expansão de um arquivo de acesso sequencial é indicado pelo comando "APPEND".

```
PRINT D$ "APPEND ARQUIVO"
```

Com este comando no início do programa, o Sistema Operacional providencia para que a gravação seja feita após o último registro gravado anteriormente. O procedimento de gravação é normal.

ATENÇÃO - Um OPEN desgrava todo o conteúdo do arquivo, iniciando o acesso pelo Registro 1; por isso, para expansão do arquivo lembrar sempre de usar o comando APPEND.

ARQUIVO ANTES DO APPEND

REGISTRO 1	REGISTRO 2	REGISTRO 3	REGISTRO 4	EOF

ARQUIVO APÓS COMANDO APPEDN

REGISTRO 1	REGISTRO 2	REGISTRO 3	REGISTRO 4	

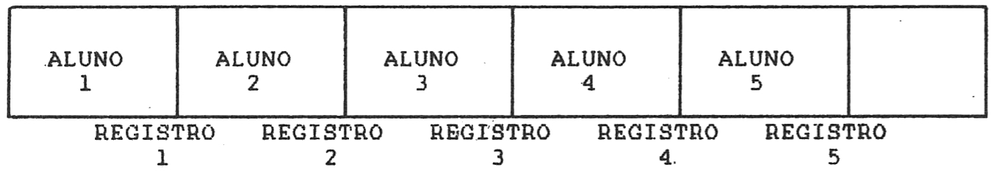
↑
Posição do primeiro registro a ser gravado

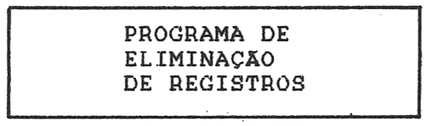
9.5 ELIMINAÇÃO DE REGISTROS DE UM ARQUIVO

Para eliminar um registro de um arquivo, deve-se copiá-lo sem este registro. É o mesmo conceito de desgravar a música de uma fita. Deve-se copiar a fita, suprimindo-se esta música.

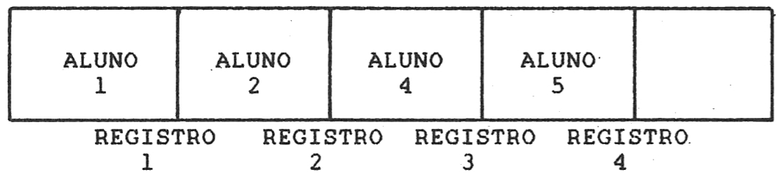
Não existe comando específico, sendo preciso fazer um programa especial.

ARQUIVO PERMANENTE ORIGINAL





ARQUIVO PERMANENTE SEM O REGISTRO DESEJADO



Como se observa foi eliminado o registro referente ao aluno 3.

9.6 PROTEÇÃO DE UM ARQUIVO CONTRA FALHAS OPERACIONAIS

Conforme comentado anteriormente, se um arquivo é aberto pelo comando OPEN, ele fica sem a indicação EOF de fim de arquivo e não poderá ser utilizado novamente, se por alguma falha operacional (falta de energia por exemplo), o programa for interrompido antes do comando CLOSE.

Duas alternativas serão fornecidas para evitar este tipo de problema, mas para que elas sejam viáveis é preciso que TODO ARQUIVO TENHA SEMPRE UMA DUPLICATA EM OUTRO DISCO.

a) No caso de falha, basta recomeçar o processamento com a cópia.
b) Outra solução é a criação de um arquivo Auxiliar com o número do último registro atualizado.

Em caso de falha, basta recomeçar o processamento a partir do último registro atualizado. Esta alternativa é útil para processamentos muito demorados, pois evita-se repetir todo o processo já feito de maneira correta.

Para que esta alternativa funcione é preciso que cada acesso aos arquivos seja dado um OPEN e CLOSE. Este procedimento evidentemente tornará o processamento mais lento, apesar de mais seguro.

Exemplo 1:

Adição de novos registros num ARQUIVO.

Supõe-se que, na gravação do arquivo "DADSPESS", somente alguns funcionários foram cadastrados num dia, os demais ficaram para outro dia. O programa de atualização do arquivo será chamado de "NOVSFUNC".


```
 10  REM "NOVSFUNC"
 20  REM PROGRAMA QUE CADASTRA NOVOS FUNCIONARIOS
 30  REM ARMAZENADOS NO ARQUIVO "DADSPESS"
 40  REM
 50  D$ = CH$(4)
 60  INPUT "QUANTOS FUNCIONARIOS JA FORAM CADASTRADOS"; N
 70  PRINT D$ "APPEND DADSPESS"
 80  FOR I=N TO 100
 90  INPUT "QUAL O NOME DO FUNCIONARIO ";N$
100  INPUT "QUAL O RG"; RG#
110  PRINT D$ "WRITE DADSPESS"
120  PRINT N$
130  PRINT RG#
140  PRINT D$
150  INPUT "TEM MAIS ALGUEM PARA CADASTRAR"; R$
160  IF R$ = "NAO THEN GOTO 180
170  NEXT I
180  PRINT D$ "CLOSE DADSPESS"
190  END
```

Exemplo 2:

Eliminar um funcionário do Arquivo "DADSPESS".

Será criado um arquivo temporário "TEMP" para receber o arquivo sem o funcionário . Em seguida este arquivo será copiado em cima do - "DADSPESS", original, resultando finalmente um "DADSPESS" sem o funcionário excluído.

A - Programa para exclusão de registros de um arquivo

```
 10  REM PROGRAMA NVSFUN2
 20  REM EXCLUI FUNCIONARIOS DO ARQUIVO DADSPESS
 30  REM
 40  D$ = CHR$(4)
 50  PRINT D$ "OPEN DADSPESS"
 60  PRINT D$ "OPEN TEMPORAR"
 70  PRINT D$
 80  INPUT "QUANTOS FUNCIONARIOS ESTAO NO ARQUIVO"; N
 90  FOR I = 1 TO N
100  PRINT D$ "READ DADSPESS"
110  INPUT N$, RG#
120  PRINT D$
130  PRINT "O FUNCIONARIO";N$;"CONTINUA NO ARQUIVO"
140  INPUT "DIGITE SIM OU NAO"; R$
150  IF R$ = "NAO" THEN GOTO 200
160  PRINT D$ "WRITE TEMPORAR"
170  PRINT N$
180  PRINT RG#
190  NEXT I
200  PRINT D$ "CLOSE TEMPORAR"
210  PRINT D$ "CLOSE DADSPESS"
220  END
```

B - Armazenando este programa

```
SAVE NOVSFUN2
```

C - Desgravando arquivo DADSPESS antigo

DELETE DADSPESS

D - Alterando nome do arquivo TEMPORAR para novo DADSPESS

RENAME TEMPORAR, DADSPESS

Neste ponto o arquivo DADSPESS não conterá mais os funcionários excluídos.

Exemplo 3:

Proteção contra falhas operacionais num arquivo extenso.

Se o arquivo contém muitos registros, uma falha durante a sua gravação fará com que todo o conteúdo seja perdido. Para evitar tal imprevisto, o programa terá um OPEN e CLOSE para cada acesso.

```
 10  REM "GRVFUN2"
 20  REM ESTE PROGRAMA E UMA NOVA VERSAO
 30  REM DO PROGRAMA "GRDDFUN".
 40  REM
 50  REM APESAR DE MAIS LENTO E MAIS SEGURO
 60  REM POIS NAO PERDE TODO O ARQUIVO SE FOR
 70  REM INTERROMPIDO POR FALHA OPERACIONAL
 80  REM
 90  D$ = CHR$(4)
100  FOR I = 1 TO 100
110  HOME
120  INPUT "ENTRE COM O NOME DO FUNCIONARIO"; N$
130  INPUT "ENTRE COM O RG"; RG#
140  PRINT "D$ "OPEN DADSPESS"
150  PRINT D$ "WRITE DADSPESS"
160  PRINT N$
170  PRINT RG#
180  PRINT D$ "CLOSE DADSPESS"
190  PRINT D$
200  INPUT "NOVO FUNCIONARIO. SIM OU NAO"; R$
210  IF R$ = "NAO" GO TO 190
220  NEXT I
230  HOME
240  END
```

EXERCÍCIOS

1. Atualizar um Arquivo Permanente através do uso de um Arquivo Movimento.

2. Incorporar o código do cargo ao arquivo "DADSPESS". Criar um Arquivo Auxiliar.

3. Listar o nome do cargo, tendo sido armazenado o código do cargo (usar Arquivo Auxiliar).

4. Fazer uma alteração no programa GRVFUN2 para que seja gravado um Arquivo Auxiliar com o número do último funcionário gravado.

CAPITULO 10

MANIPULAÇAO DE ARQUIVOS DE ACESSO DIRETO

Um arquivo de acesso direto ou arquivo de acesso aleatório é aquele no qual é possível gravar ou ler informações de um registro qualquer. Nos arquivos estudados no capítulo anterior não havia como controlar o registro a ser lido ou gravado, pois cada novo acesso é sempre feito no registro seguinte, daí o nome de arquivo sequencial.

Existem inúmeras aplicações onde é desejável acessar um registro fora de sequência, pois não é possível saber com antecedência qual o próximo registro a ser acessado.

Exemplo:

Mantém-se um arquivo com os dados pessoais dos funcionários de uma empresa. Ao longo do tempo os dados precisam ser atualizados, pois um funcionário casa, outro passa a ter mais um dependente etc. Em cada processamento de atualização dos dados, somente um ou dois registrados deverão ser atualizados. Seria uma perda de tempo, apesar de o computador ser muito rápido, percorrer todo o arquivo indagando em cada acesso se aquele registro corresponde ao funcionário desejado.

O procedimento correto nesta situação é acessar diretamente o registro desejado. Um arquivo que permite este tipo de acesso é chamado de arquivo de acesso direto, ou de acesso aleatório (Random).

Com o conhecimento do uso de arquivos é possível montar um conjunto de programas que acessem arquivos com dados relacionados.

Entende-se por Sistema de Processamento de Dados um conjunto de programas acessando arquivos que mantêm relações entre seus dados. O arquivo-base do Sistema é chamado de Arquivo Permanente.

Conforme já mencionado, o sistema tem outros arquivos que o compõem, cuja finalidade é facilitar o acesso ao Arquivo Permanente.

Os programas que compõem um Sistema podem ser subdivididos em dois grandes grupos:

- Programas para criação/atualização do Arquivo.
- Programas para emissão de relatórios.

10.1 COMANDO OPEN PARA ARQUIVO DE ACESSO DIRETO

Conforme visto anteriormente, a manipulação de arquivos sequenciais é feita através de variáveis e não de registros. Foi necessário criar o conceito de REGISTRO LÓGICO, composto de um conjunto de registros físicos que são os campos do REGISTRO LÓGICO.

No caso do acesso direto, existe o conceito de REGISTRO como um conjunto de campos, sendo NECESSÁRIO especificar o número de BYTES do registro.

A especificação do número de BYTES do registro é feita através do parâmetro L(lenght) - extensão ou comprimento - do registro.

"OPEN ARQUIVO,LN"

N - indica o número de BYTES do registro
LN - significa: registro com extensão N ou registro com N BYTES

10.2 COMANDO WRITE PARA ARQUIVO DE ACESSO DIRETO

É preciso indicar neste caso qual o registro a ser gravado.

"WRITE ARQUIVO, R"; N

Neste caso, além do nome do ARQUIVO, é preciso indicar o número do registro: - N

10.3 COMANDO READ PARA ARQUIVO DE ACESSO DIRETO

Indica-se, da mesma forma que na gravação, qual o registro a ser acessado.

"READ ARQUIVO, R"; N

Exemplo de uso dos comandos deste capítulo:
Criar um arquivo para armazenar os dados dos alunos de uma escola. Cada aluno terá: CÓDIGO - NOME - 3 NOTAS.

Para tornar o teste do Sistema viável, deve-se criar um arquivo para um máximo de 1000 alunos e testar com apenas 15 dados reais.

COMPONENTES DO SISTEMA

- Arquivo - "ARQALN"
- Programa de Gravação - "GRVALN"
- Programa de emissão de dados dos alunos solicitados - "LSTALN"

Para simplificar o sistema, os alunos serão numerados de 1 a 1000.

10.3.1 Cálculo do tamanho do arquivo

Número	- NU%	-	2 BYTES
Nome	- NO$	-	35 BYTES
Nota1	- N1	-	4 BYTES
Nota2	- N2	-	4 BYTES
Nota3	- N3	-	4 BYTES
TOTAL BYTES P/REGISTRO - 49 BYTES			

$$\text{N Registros por setor} = \frac{256}{49} = 5.69$$

O N.Registros por Setor = 5

Para armazenar 1000 alunos necessita-se de $\frac{1000}{5} = 200$ setores.

EXEMPLO DE PROGRAMA DE GRAVAÇÃO DE ARQUIVO DE ACESSO DIRETO

```
 10  REM  NOME DO PROGRAMA  GRVALN
 20  REM  PROGRAMA USA O ARQUIVO ARQALN QUE CONTEM
 30  REM  DADOS DOS ALUNOS DE UMA ESCOLA
 40  REM
 50  REM
 60  D$ = CHR$(4)
 70  INPUT "QUANTOS ALUNOS SERAO GRAVADOS"; NR%
 80  PRINT  D$ "OPEN ARQALN,L49
 90  FOR  I% = 1 TO NR%
100  HOME
110  INPUT "QUAL O NUMERO DO ALUNO"; NU%
120  INPUT "QUAL O NOME DO ALUNO"; NO$
130  INPUT "QUAL A 1. NOTA DESTE ALUNO"; N1
140  INPUT "QUAL A 2. NOTA DESTE ALUNO"; N2
150  INPUT "QUAL A 3. NOTA DESTE ALUNO"; N3
160  REM
170  REM  GRAVANDO OS DADOS DO ALUNO
180  REM
190  PRINT  D$ "WRITE ARQALN,R"; I%
200  PRINT  NU%
210  PRINT  NO$
220  PRINT  N1
230  PRINT  N2
240  PRINT  N3
250  REM
260  REM  DESATIVANDO O ACESSO AO DISCO - PRINT  D$
270  REM
280  PRINT  D$
290  NEXT  I%
300  PRINT  D$ "CLOSE ARQALN"
310  PRINT  D$
320  HOME
330  PRINT "PROCESSAMENTO ENCERRADO
340  END
```

EXEMPLO DE LEITURA DE UM ARQUIVO DE ACESSO DIRETO

```
 10  REM  PROGRAMA  LSTALN
 20  REM  ESTE PROGRAMA LE O ARQUIVO ARQALN
 30  REM
 40  D$ = CHR$(4)
 50  PRINT  D$ "OPEN ARQALN,L49"
```

```
 60  HOME
 70  INPUT "DESEJA LISTAR DADOS DE UM ALUNO - SIM OU NAO"; R$
 80  IF  R$ = "NAO" THEN GOTO 210
 90  INPUT "QUAL O NUMERO DO ALUNO"; NU%
100  PRINT  D$ "READ ARQALN,R"; NU%
110  INPUT  NU%,NO$,N1,N2,N3
120  PRINT  D$
130  PRINT "AS NOTAS DO ALUNO SAO"
140  PRINT  N1,N2,N3
150  REM
160  REM  PAUSA PARA LEITURA COMANDOS 180 E 190
170  REM
180  FOR  I% = 1 TO 2000
190  NEXT  I%
200  GOTO 60
210  PRINT  D$ "CLOSE ARQALN"
```

EXERCÍCIOS

1. Fazer uma alteração no Sistema Aluno apresentado, que minimize o perigo de uma destruição do arquivo.

2. Fazer um programa para atualizar dados de um aluno.

O programa deve perguntar qual o campo a ser alterado. Em seguida fazer a alteração.

CAPÍTULO 11

MONTAGEM DE UM SISTEMA DE PROCESSAMENTO DE DADOS

Neste capítulo não será apresentado nenhum conceito novo apenas um exemplo de montagem de um Sistema de Processamento de Dados.

Deseja-se montar um arquivo onde cada registro contém os dados para endereçamento postal dos assinantes de uma revista. Neste tipo de arquivo as atualizações são frequentes e o número de registros é muito grande, o que torna antieconômico a utilização de um arquivo sequencial para este sistema.

O Sistema proposto é conhecido comercialmente como Sistema MAILING LIST. As informações constantes de cada registro são:

NOME	- Variável NM$	-	35	BYTES
EMPRESA	- Variável EP$	-	35	BYTES
ENDEREÇO	- Variável ED$	-	30	BYTES
CEP	- Variável CP$	-	5	BYTES
CIDADE	- Variável CD$	-	20	BYTES
ESTADO	- Variável ET$	-	2	BYTES
TOTAL DO REGISTRO		-	127	BYTES

Com base nos dados apresentados tentar montar:
- O Programa de gravação das etiquetas
- O Programa de emissão das etiquetas

A seguir é feita a montagem detalhada dos programas propostos. Sugere-se que somente se prossiga a leitura após a tentativa de montagem dos programas propostos.

Cálculo do N.Registros por Setor = $\frac{256}{127}$ = 2.01

Para cadastrar 500 clientes = $\frac{500}{2 \text{ por Setor}}$ = 250 Setores.

São necessários 250 Registros de 256 BYTES, ou 250 Setores.

O Sistema é composto de:
1. ARQUIVO DE DADOS "ARQETQ"
2. PROGRAMA PARA GRAVAR ARQUIVO -"GRVETQ"
3. PROGRAMA PARA EMITIR ETIQUETAS SELETIVAS-"LSTETQ"

Este Sistema tem saída por vídeo. Para torná-lo real, basta trocar o comando PRINT por uma saída pela impressora.

PRINT - apresenta saída no vídeo
LPRINT - apresenta saída pela impressora

Para tornar o exercício real, vamos trabalhar com apenas 10 clientes.

$$\text{10 clientes ocupam } \frac{\text{10 clientes}}{\text{2 clientes/Setor}} = \text{5 Setores}$$

11.1 PROGRAMA "GRVETQ"

Este programa somente grava as etiquetas. Em seguida o programa de impressão irá imprimir somente àquelas que forem selecionadas.

```
 10  REM  PROGRAMA  GRVETQ
 20  REM
 30  REM  ESTE PROGRAMA GRAVA O ARQUIVO ARQETQ
 40  REM  COM DADOS PARA ENDEREÇAMENTO
 50  REM
 60  CLEAR = 1000
 70  D$ = CHR$(4)
 80  PRINT  D$ "OPEN ARQALN,L127"
 90  PRINT  D$
100  FOR  I% = 1 TO 10
110  HOME
120  INPUT "ENTRE COM O NOME DO CLIENTE(MAX-35)"; N$
130  INPUT "ENTRE COM O NOME DA EMPRESA(MAX-35)"; E$
140  INPUT "NOME DA RUA E NÚMERO(MAX-30)"; R$
150  INPUT "QUAL O CEP"; C$
160  INPUT "NOME DA CIDADE(MAX-20)"; A$
170  INPUT "SIGLA DO ESTADO";B$
180  REM
190  REM  GRAVANDO
200  PRINT  D$ "WRITE ARQALN,R"; I%
210  PRINT  N$
220  PRINT  E$
230  PRINT  R$
240  PRINT  C$
250  PRINT  A$
260  PRINT  B$
270  PRINT  D$
280  NEXT  I%
290  PRINT  D$ "CLOSE ARQALN"
300  HOME
310  PRINT  D$
320  PRINT "PROCESSAMENTO ENCERRADO"
330  END
```

11.2 PROGRAMA "LSTETQ"

Este programa deverá ler o número do registro a ser acessado para emissão da etiqueta.

```
 10  REM  PROGRAMA  LSTETQ
 20  REM
 30  REM  ESTE PROGRAMA USA O ARQUIVO DE ACESSO DIRETO
 40  REM  ARQALN
 50  REM
 60  CLEAR 1000
 70  DIM  CL%(10)
 80  D$ = CHR$(4)
 90  PRINT  D$ "OPEN ARQALN,L127"
100  HOME
110  PRINT  D$
120  PRINT "QUANTAS ETIQUETAS SERÃO IMPRESSAS(MAX 10)"; NE%
130  REM
140  REM  LENDO OS NÚMEROS DESEJADOS
150  REM  NA ORDEM DESEJADA DE IMPRESSÃO
160  REM
170  FOR  I% = 1 TO NE%
180  HOME
190  INPUT "QUAL  NÚMERO DO CLIENTE,N<10";CL%(I%)
200  NEXT  I%
210  REM
220  REM  LENDO E IMPRIMINDO AS ETIQUETAS
230  REM
240  FOR  I% = 1 TO NE%
250  HOME
260  PRINT  D$ "READ ARQALN,R";CL%(I%)
270  INPUT  N$
280  INPUT  E$
290  INPUT  R$
300  INPUT  C$
310  INPUT  A$
320  INPUT  B$
330  PRINT  D$
340  PRINT  N$
350  PRINT  E$
360  PRINT  R$
370  PRINT  C$;"  ";A$;"  ";B$
380  FOR  K% = 1 TO 2000
390  NEXT  K%
400  NEXT  I%
410  PRINT  D$ "CLOSE ARQALN"
420  END
```

EXERCÍCIOS

1. Fazer um programa para atualizar dados de um cliente.

2. Fazer um programa para gerar um Arquivo Movimento que contém dados das etiquetas selecionadas para listagem. Com este procedimento é possível listar diversas vezes o arquivo, sem ter de acessar novamente o arquivo permanente.

3. Fazer um programa para listar as etiquetas do arquivo movimento criado.

CAPÍTULO 12

ORDENAÇÃO DE ARQUIVOS

Neste capítulo serão vistos conceitos que permitem classificar um arquivo, segundo um critério preestabelecido.

12.1 NOÇÃO DE SORT

Sort significa classificar, ordenar.

Com frequência deseja-se colocar os registros de um arquivo ordenados segundo certo critério. Este critério pode ter por base um ou mais campos do registro. Se o critério tiver por base um único campo, diz-se que a ordenação tem uma chave de classificação. Se tiver por base dois campos, diz-se que é uma ordenação com duas chaves. Neste caso, a ordem de prioridade das chaves deve ser indicada.

NOME	IDADE	ESTADO
ANTONIO CARLOS	50	SP
JOSÉ FERREIRA	18	SP
CARLOS MOREIRA	40	RJ
TADEU DOS SANTOS	32	PR
MILTON PEREIRA	18	BA

Se a idade for única chave de classificação, ao final o arquivo ficará assim:

NOME	IDADE	ESTADO
JOSÉ FERREIRA	18	SP
MILTON PEREIRA	18	BA
TADEU DOS SANTOS	32	PR
CARLOS MOREIRA	40	RJ
ANTONIO CARLOS	50	SP

Se classificação tiver por base duas chaves, sendo a primeira ESTADO e a segunda IDADE, o arquivo deverá ficar após a classificação como segue:

NOME	IDADE	ESTADO
MILTON PEREIRA	18	BA
TADEU DOS SANTOS	32	PR
CARLOS MOREIRA	40	RJ
JOSÉ FERREIRA	18	SP
ANTONIO CARLOS	50	SP

Se a classificação tiver como primeira chave a IDADE e segunda o ESTADO, o arquivo terá a seguinte configuração:

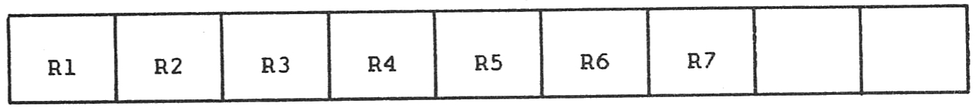
NOME	IDADE	ESTADO
MILTON PEREIRA	18	BA
JOSÉ PEREIRA	18	SP
TADEU DOS SANTOS	32	PR
CARLOS MOREIRA	40	RJ
ANTONIO CARLOS	50	SP

12.2 CLASSIFICAÇÃO DE ARQUIVOS

12.2.1 Ordem crescente

A classificação de um arquivo, com uma única chave é feita com base na seguinte idéia:

ARQUIVO

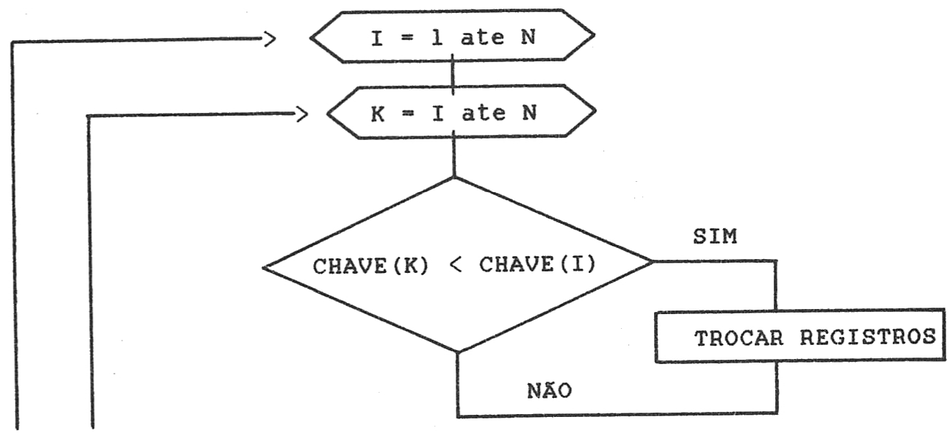


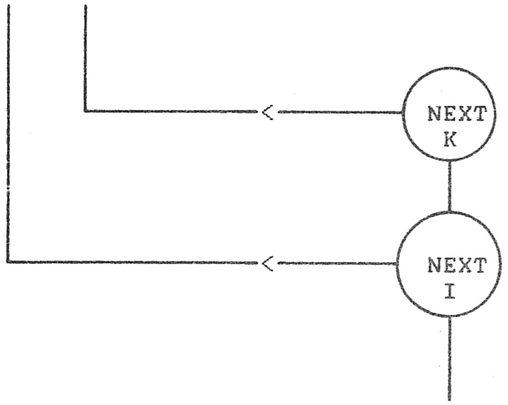
Compara-se inicialmente todos os registros com o registro número 1. Se o valor da chave de classificação de um registro qualquer for menor que o valor da chave do registro 1, troca-se o conteúdo do registro 1 com o registro em questão. Após comparar-se todos os registros com o de número 1, nesta posição estará o registro que tem o menor valor para a chave.

Repetindo-se o processo, agora com o registro 2, isto é, comparando-se todos os demais registros 3, 4, 5, ... com o registro 2 e trocando-se sempre que for encontrado um com a chave de menor valor que o da chave do registro 2, teremos ao final a menor chave colocada na posição 2 dentre os registros R2, R3, ... RN, restantes após a classificação do registro 1.

Repetindo-se este processo para o Registro 3, isto é, comparando-se R3 com R4, R5, ... RN e trocando-se sempre que for encontrado uma chave menor, tem-se na terceira posição o registro com a menor chave entre R3, R4,...RN. Este processo será repetido para todos os registros do arquivo.

Apesar de parecer um processo complexo, torna-se extremamente simples com uso de dois comandos FOR _ NEXT:





O FOR - NEXT com a variável I controla o registro base, ou seja, aquele que será comparado com os demais.

Na primeira vez I=1, significa que o valor da chave(1) será comparada com as demais. Sempre que outra chave for menor, os registros serão trocados.

Após comparar chave(1) com todas as outras chaves(K), tem-se o registro(1) com a menor chave. Neste momento, I assume o valor 2 e o processo se repete. Quando I atingir o valor N, todo o arquivo estará em ordem crescente pela chave desejada.

O caso de mais de uma chave de classificação será visto na aula 10.

Exemplo de ordenação de arquivos:

Cada registro exemplo arquivo terá dois campos:

```
CAMPO1 - N$   =  NOME    - Variável STRING  - 35 BYTES
CAMPO2 - NM$% = NÚMERO - Variável INTEIRA -  2 BYTES
                                  TOTAL     37 BYTES
```

Num campo STRING, as comparações abaixo são sempre verdadeiras.

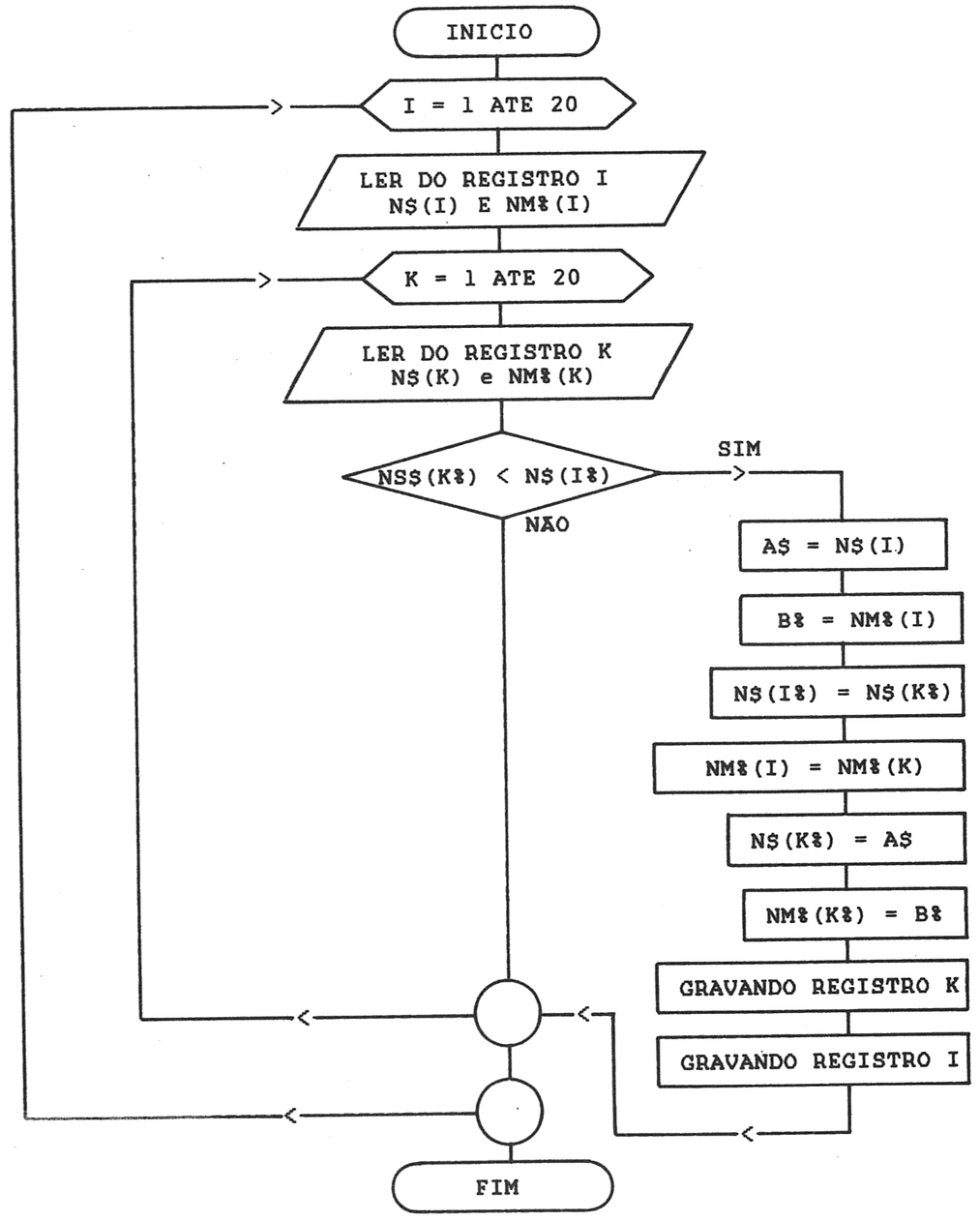
"A" < "B"

"AB" < "AM"

"AMADEU" < "ANTONIO"

"ANTONIO CARLOS" < "ANTONIO JOSE"

Como se observa, a ordem alfabética é obtida comparando-se uma STRING diretamente com outra. Neste exemplo, o arquivo será limitado a 20 registros de 37 BYTES cada um. Vamos admitir que já foi criado o Arquivo Permanente, chamado de ARQALUNO.



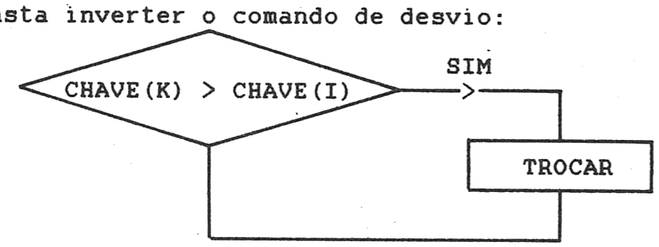

```
10  REM PROGRAMA PARA ORDENAR O ARQUIVO ARQALUNO
20  REM
30  D$ = CHR$(4)
40  PRINT D$ "OPEN ARQALUNO, L37"
50  FOR I% = 1 TO 20
60  PRINT D$ "READ ARQALUNO, R"; I%
70  INPUT N$(I%), NM(I%)
80  FOR K% = I% TO 20
```

```
 90  PRINT D$ "READ ARQALUNO, R"; K%
100  INPUT N$(K%), NM%(I%)
110  IF N$(K%) < N$(I%) THEN GOTO 500
120  NEXT K%
130  NEXT I%
500  REM
510  REM TROCANDO OS REGISTROS
520  REM
530  A$ = N$(I%)
540  B% = NM(I%)
550  N$(I%) = N$(%)
560  NM%(I%) = NM%(K%)
570  N$(K%) = A$
580  NM%(K%) = B%
590  PRINT D$ "WRITE ARQALUNO, R"; K%
600  PRINT N$(K%)
610  PRINT NM%(K%)
620  REM
630  PRINT D$ "WRITE ARQALUNO, R "; I%
640  PRINT N$(I%)
650  PRINT NM%(I%)
660  REM
670  NEXT K%
680  NEXT I%
690  PRINT D$ "CLOSE ARQALUNO"
700  PRINT D$
710  HOME
720  PRINT "PROCESSAMENTO ENCERRADO"
730  END
```

O comando 700 é para desativar o acesso ao "drive". Caso contrário seria feita uma tentativa de gravar no ARQALUNO, mas como ele está fechado, sairia uma mensagem de erro.

12.2.2 Classificação em Ordem Decrescente

Neste caso basta inverter o comando de desvio:



EXERCICIOS

1. Fazer o programa para gravar o Arquivo Permanente.

2. Fazer o programa para gravar o Arquivo Movimento.

3. Fazer o programa para listar o Arquivo Movimento.

CAPÍTULO 13

CONCEITO RENAME E NOÇÃO MERGE

13.1 COMANDO RENAME

Este comando faz parte do Sistema Operacional TKDOS e permite trocar o nome do arquivo ou programa, o que é um recurso muito útil como será visto neste capítulo.

```
RENAME "NOMEVELHO"    "NOMENOVO"
```

Como se observa, basta colocar entre aspas o nome do arquivo (ou programa) a ser trocado e após uma vírgula o novo nome do arquivo (ou programa) também entre aspas.

13.2 MERGE

Merge traduz-se por intercalar.

A ordenação de vetores é feita na memória para ser rápida. Com isto tem-se uma limitação física que é a capacidade de memória do computador.

Para classificar-se um arquivo extenso, a solução é subdividi-lo em arquivos menores, compatíveis com a capacidade de memória do computador utilizado. Em seguida estes arquivos já classificados serão intercalados 2 a 2. A este processo de intercalar dois arquivos dá-se o nome de MERGE.

Exemplo:

O Arquivo Permanente do Sistema Alunos está ordenado pelo campo NOME, isto é, está em ordem alfabética:

Um programa de atualização deve incluir novos nomes no arquivo, os quais deverão ser colocados de maneira que a ordem alfabética seja mantida.

ARQUIVO PERMANENTE

ANTONIO ALMEIDA	REGISTRO 1
DIRCEU RABELO	REGISTRO 2
EDUARDO GONCALVES	REGISTRO 3
FRANCISCO GUIMARÃES	REGISTRO 4
FRANCISCO PEREIRA	REGISTRO 5
GUSTAVO DE ABREU	REGISTRO 6
HILDEBRANDO PASSO	REGISTRO 7
MARIO DE ALMEIDA	REGISTRO 8
NESTOR DE CASTRO	REGISTRO 9
PAULO CUNHA	REGISTRO 10
RENATO SILVA	REGISTRO 11

Os nomes que deverão ser colocados no Arquivo Permanente são:

ANTONIO CARLOS	REGISTRO 1
JOSÉ FERREIRA	REGISTRO 2
CARLOS MOREIRA	REGISTRO 3
TADEU DOS SANTOS	REGISTRO 4
MILTON PEREIRA	REGISTRO 5

Estes nomes deverão ser colocados no Arquivo Movimento do Sistema. Após classificado, o arquivo apresenta o seguinte conteúdo.

REGISTRO 1	ANTONIO CARLOS
REGISTRO 2	CARLOS MOREIRA
REGISTRO 3	JOSÉ FERREIRA
REGISTRO 4	MILTON PEREIRA
REGISTRO 5	TADEU DOS SANTOS

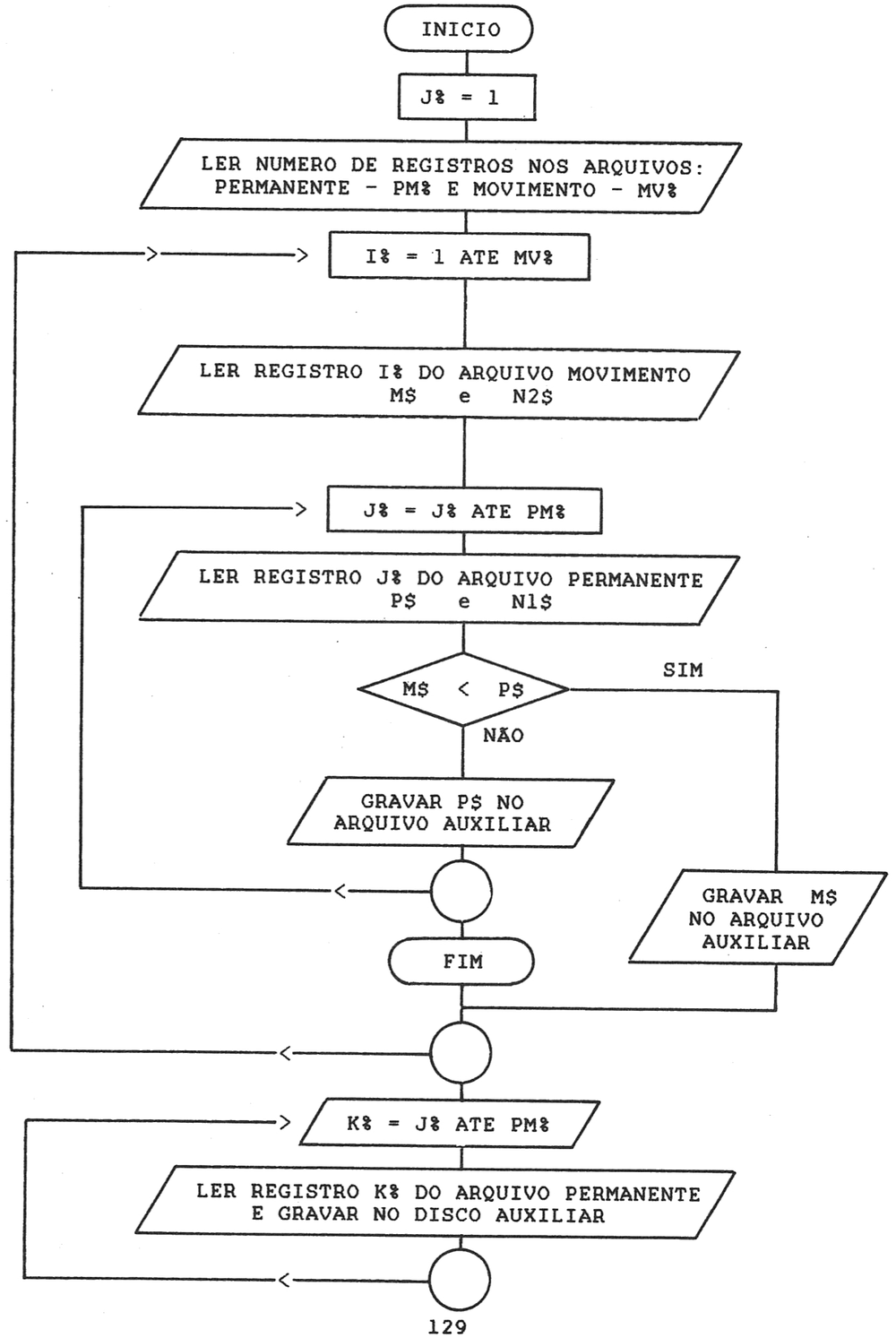
O programa que faz o MERGE deve ler um registro do Arquivo Movimento e descobrir onde ele deve ser inserido no Arquivo Permanente. É gravado um novo Arquivo Permanente composto do Arquivo Permanente anterior mais os registros provenientes do Arquivo Movimento. Este arquivo junção do Permanente e Movimento é mais um exemplo de ARQUIVO AUXILIAR.

Os nomes intercalados foram colocados para melhor visualização na listagem, um pouco deslocados à direita. O novo Arquivo Permanente terá os seguintes registros:

NOVO ARQUIVO PERMANENTE

REGISTRO 1	ANTONIO ALMEIDA
REGISTRO 2	ANTONIO CARLOS
REGISTRO 3	CARLOS MOREIRA
REGISTRO 4	DIRCEU RABELO
REGISTRO 5	EDUARDO GONÇALVES
REGISTRO 6	FRANCISCO GUIMARÃES
REGISTRO 7	FRANCISCO PEREIRA
REGISTRO 8	GUSTAVO DE ABREU
REGISTRO 9	HILDEBRANDO PASSOS
REGISTRO 10	JOSÉ FERREIRA
REGISTRO 11	MÁRIO DE ALMEIDA
REGISTRO 12	MILTON PEREIRA
REGISTRO 13	NESTOR DE CASTRO
REGISTRO 14	PAULO CUNHA
REGISTRO 15	RENATO SILVA
REGISTRO 16	TADEU DOS SANTOS

O FLUXOGRAMA A SEGUIR REALIZA O MERGE DE DOIS ARQUIVOS



A codificação a seguir supõe que já foram criados e gravados os Arquivos Permanente e Movimento.

Cada registro destes arquivos tem:

NOME	- com 35 caracteres	(35 BYTES)
NÚMERO	- variável inteira	(2 BYTES)

Supõe-se também que os dois arquivos já foram ordenados pelo programa de ordenação apresentado.

1. PROGRAMA PARA INTERCALAR DOIS ARQUIVOS

```
 10  REM  PROGRAMA  MERGE
 20  REM
 30  REM  INTERCALA DOIS ARQUIVOS PERMANENTE - ARQPERM
 40  REM  E O MOVIMENTO  - ARQMOV
 50  REM  USA UMA ARQUIVO AUXILIAR - ARQMERGE
 60  REM
 70  CLEAR 1000
 80  HOME
 90  D$ = CHR$(4)
100  PRINT  D$ "OPEN ARQPERM,L37"
110  PRINT  D$ "OPEN ARQMOV,L37"
120  PRINT  D$ "OPEN ARQMERGE"
130  REM
140  J%=1
150  PRINT  D$
160  INPUT "QUANTOS REGISTROS TEM O ATUAL ARQ PERMANENTE"; PM%
170  INPUT "QUANTOS REGISTROS TEM O ARQUIVO MOVIMENTO"; MV%
180  FOR I% = 1 TO MV%
190  PRINT  D$ "READ ARQMOV,R";I%
200  INPUT  M$
210  INPUT  N2%
220  FOR J% = J% TO PM%
230  PRINT  D$ "READ ARQPERM,R";J%
240  INPUT  P$
250  INPUT  N1%
260  IF M$<P$ THEN GOTO 400
270  REM
280  GRAVANDO DADOS DO ARQUIVO PERMANENTE NO AUXILIAR
290  PRINT  D$ "WRITE ARQMERGE"
300  PRINT  P$
310  PRINT  N1%
320  NEXT  J%
330  GOTO 580
400  REM
410  GRAVANDO DADOS DO ARQUIVO MOVIMENTO NO AUXILIAR
420  PRINT D$ "WRITE ARQMERGE"
430  PRINT  M$
440  PRINT  N2%
450  NEXT  I%
460  REM
470  REM  GRAVANDO OS REGISTROS FINAIS DO ARQUIVO
480  REM  PERMANENTE NO ARQUIVO AUXILIAR
490  REM
500  FOR K% = J% TO PM%
510  PRINT  D$ "READ ARQPERM,R";K%
520  INPUT  P$
530  INPUT  N1%
540  PRINT  D$ "WRITE ARQMERGE"
550  PRINT  P$
560  PRINT  N1%
570  NEXT  K%
580  PRINT  D$ "CLOSE ARQPERM"
590  PRINT  D$ "CLOSE ARQMOV"
600  PRINT  D$ "CLOSE ARQMERGE"
610  END
```


2. DESGRAVANDO O ARQUIVO PERMANENTE ORIGINAL E O ARQUIVO MOVIMENTO

```
DELETE  ARQPERM
DELETE  ARQMOV
```

3. CRIANDO O NOVO ARQUIVO PERMANENTE

```
RENAME  ARQMERGE,ARQPERM
```

Neste ponto tudo se passa como se ao arquivo ARQPERM tivesse sido incorporado os registros do Arquivo Movimento, os novos registros colocados em posições que mantêm a ordenação alfabética de todo o conjunto.

EXERCÍCIOS

1. Fazer um programa para classificar o Arquivo Movimento e o Arquivo Permanente.

2. Alterar o sistema para classificar por duas chaves Ordem alfabética e dentro desta por número de matrícula.

CAPITULO 14

COMANDO MAXFILES E ARQUIVOS DE ACESSO CHAVEADO

Na montagem de um Sistema de Processamento de dados, os arquivos sofrem acréscimos de registros, alterações de conteúdo.

14.1 COMANDO MAXFILES

Este comando permite aumentar o número de arquivos num programa, podendo-se trabalhar com um máximo de 16 arquivos simultaneamente.

Se este comando não for utilizado, o máximo estará limitado automaticamente a três, conforme mencionado anteriormente.

"MAXFILES N"

N - está limitado a 16

Exemplo de uso dos comandos APPEND E MAXFILES

```
10  REM  EXEMPLO DE USO DE COMANDOS
20  REM
30  D$ = CHR$(4)
40  PRINT  D$ "APPEND  ARQUIVO"
50  PRINT  D$ "MAXFILES 5"
60  PRINT  D$ "MAXFILES 5"
70  PRINT  D$ "MAXFILES 5"
```

Foi definido que será feito um acréscimo de registros no ARQUIVO e o programa trabalhará simultaneamente com cinco arquivos.

14.2 ARQUIVOS DE ACESSO CHAVEADO

Anteriormente foi apresentado um Sistema de cadastramento de alunos, onde cada um tem número, nome e três notas. Como restrição, foi imposta a condição de o número variar entre 1 e 1000, ou seja, para um conjunto de 1000 alunos, a numeração é totalmente sequencial, sem faltar nenhum aluno.

Ocorre que esta é uma situação hipotética. Geralmente, os códigos são atribuídos de acordo com algum critério, dificilmente gerando uma numeração sequencial.

Exemplo:
O número do aluno é composto de segmentos.

Segmento 1 - Ano de ingresso (dois dígitos)
Segmento 2 - Semestre de ingressão (um dígito)
Segmento 3 - Numeração sequencial (quatro dígitos)

Como resultado deste critério, o número de um aluno resulta em 8211020, se for um aluno que entrou na Faculdade no primeiro semestre de 82.

Colocados num arquivo tem-se por exemplo:

7913040 ,	7922045 ,	8011519 ,	8011525 ,	8021740 ,....
Registro 1	Registro 2	Registro 3	Registro 4	Registro 5
aluno 1	aluno 2	aluno 3	aluno 4	aluno 5

Neste caso não é possível ser feita a associação anterior:

Registro 1 aluno número 1
" 2 " " 2
" 3 " " 3

O problema a ser resolvido é o seguinte:

Como encontrar um aluno neste arquivo? Sabe-se que ele tem número de matrícula 8012137. Uma solução imediata é ler cada registro e perguntar se o número de matrícula do aluno é o procurado. Para este procedimento é necessário que seja lido o registro de cada aluno e em seguida ser feita uma comparação.

O acesso ao disco é um processo extremamente lento se comparado às operações na memória interna do computador, pois depende de movimentação mecânica do cabeçote de leitura.

Para que o processo de busca se realize na memória é preciso que os números dos alunos estejam na memória:

Esta é essência do método de acesso chaveado, ou acesso indireto, que são diferentes nomes para o mesmo processo.

Este método consiste na criação de um Arquivo Auxiliar que contém somente os **NÚMEROS DOS ALUNOS**, constituindo a chave para acessar o Arquivo Permanente.

Arquivo Permanente

7913040 Jose de Almeida	7922045 Gaspar dos Santos
Registro 1	Registro 2

8011519 Maria Goncalves	8011525 Benedito Pereira
Registro 3	Registro 4

Arquivo Auxiliar

7913040	7922045	8011519	8011525	
Registro 1	Registro 2	Registro 3	Registro 4	

O Arquivo Auxiliar será acessado **uma única vez e armazenado na memória** interna. A pesquisa será feita na memória, o que é extremamente rápido.

Exemplo:

No Sistema alunos feito anteriormente, a eliminação da restrição do número do aluno está compreendida entre 1 e 1000. Fazer um programa e atualização do Arquivo Permanente.

As alterações no Sistema, decorrentes da eliminação da restrição são:

- Criar um Arquivo Auxiliar.
- O programa de gravação deve gravar dados no Arquivo Permanente e no Auxiliar.
- O programa de leitura deve ler o número do aluno, em seguida descobrir em qual registro este aluno se encontra, para em seguida atualizar o registro.

Para tornar o teste real, vamos limitar o número de alunos a 20.

A codificação do programa está na página seguinte.

- Programa de Atualização do Arquivo Permanente.

```
  5  DIM NM%(100)
 10  REM NOME DO PROGRAMA "ATLZAL"
 20  REM ESTE PROGRAMA ATUALIZA O ARQUIVO PERMANENTE
 30  REM "ARQALN" USANDO TAMBEM O "ARQAUX"
 40  REM
 50  CLEAR 1000
 60  D$ = CHR$(4)
 70  PRINT D$ "OPEN ARQALN, L49"
 80  PRINT D$ "OPEN ARQAUX, L4"
 90  HOME
100  REM LENDO DADOS DO ARQUIVO AUXILIAR
110  REM
120  PRINT D$
130  INPUT "QUANTOS ALUNOS ESTAO NO ARQUIVO"; NA%
140  FOR I% = 1 TO NA%
150  PRINT D$ "READ ARQAUX, R"; I%
160  INPUT NM%(I%)
170  NEXT I%
180  HOME
190  PRINT D$
200  INPUT "QUAL O NUMERO DO ALUNO"; NU%
210  INPUT "QUAL O NOME DO ALUNO"; NO%
220  INPUT "QUAL A NOTA1 DO ALUNO"; N1
230  INPUT "QUAL A NOTA2 DO ALUNO"; N2
240  INPUT "QUAL A NOTA3 DO ALUNO"; N3
250  REM
260  REM DESCOBRINDO O REGISTRO
```



```
270  REM
280  FOR I% = 1 TO NA%
290  IF NU% = NM%(I%) THEN GO TO 330
300  NEXT I%
310  REM
320  REM GRAVANDO NOVOS DADOS DO ALUNO
330  PRINT D$ "WRITE ARQALN, R"; I%
340  PRINT NU%
350  PRINT NO%
360  PRINT N1
370  PRINT N2
380  PRINT N3
390  HOME
400  PRINT D$
410  INPUT "TEM OUTRO ALUNO PARA ALTERAR DADOS SIM OU NAO"; R$
420  IF R$ ="SIM" THEN GO TO 200
430  END
```

EXERCÍCIOS

1. Fazer o programa de gravação do Arquivo Permanente.
2. Fazer o programa de gravação do Arquivo Auxiliar.

CAPITULO 15

STRINGS

A existência de recursos na linguagem BASIC para transformar variáveis numéricas em STRING e a possibilidade de somar duas STRINGs que geram uma terceira STRING permitem classificar arquivos, com mais de uma chave de classificação, de maneira relativamente simples.

15.1 COMANDO STR$

Este comando transforma um número ou variável numérica na STRING correspondente.

```
10  A% = 1200
20  B$ = STR$ A%
30  PRINT B$
```

O resultado impresso será 1200 decorrente de B$ ="1200", diferente de A% = 1200 que é um número e não uma STRING. Em termos de ocupação de memória, A% e B$também diferem:

A% ocupa 2 BYTES - por ser variável inteira
B$ ocupa 4 BYTES - por ter de armazenar quatro caracteres

15.2 CONCATENAÇAO DE STRING

É possível adicionar ou juntar duas STRINGs gerando uma terceira STRING

```
10  A$ = "12"
20  B$ = "15"
30  C$ = A$+B$
40  PRINT  C$
```

O resultado armazenado em C$ será "1215".

Com estes dois conceitos mais o de arquivo chaveado visto anteriormente é possível classificar arquivos com mais de uma chave de classificação.

A idéia básica consiste na justaposição das chaves transformadas em STRING e que constituirá a nova chave de classificação.

É importante ressaltar que a ordem da justaposição é que irá indicar qual a prioridade nas chaves. Supondo-se um Registro que contém:

NOME - N$
IDADE - ID
ESTADO - E$

Para fazer uma classificação com duas chaves, sendo a primeira chave a Idade e a segunda Estado, a concatenação deve ser:

B$ = STR$ID + E$

Se a primeira chave for Estado e a segunda, Idade, deve-se ter:

B$ = E$ + STR$ID

Um exemplo esclarecerá o que foi comentado e mostrará como deve ser feita a classificação.

Deseja-se classificar um arquivo com dados dos eleitores de uma cidade. Os eleitores pertencem a distritos e dentro de cada distrito têm-se zonas eleitorais.

REGISTRO 1
- NOME _ CARLOS NASCIMENTO
- DISTRITO - SANTA CECILIA
- ZONA - 19

REGISTRO 2
- NOME - BENEDITO DOS SANTOS
- DISTRITO - TATUAPÉ
- ZONA - 31

15.3 ORDENAÇÃO COM DUAS OU MAIS CHAVES DE CLASSIFICAÇÃO

A ordenação deverá ser primeiro por distrito e dentro de cada distrito por zona.

Vai ser criado um Arquivo Auxiliar que contém somente a nova chave de classificação e o número do registro desta chave dentro do arquivo.

ARQUIVO AUXILIAR ANTES DA CLASSIFICAÇÃO

"SANTA CECILIA19"	1	"TATUAPE31	2				
REGISTRO 1		REGISTRO 2		REGISTRO 3		REGISTRO 4	

ARQUIVO AUXILIAR DEPOIS DA CLASSIFICAÇÃO

	18		41		"SANTA CECILIA19"	1		"TATUAPE31"	2
REGISTRO 1		REGISTRO 2			REGISTRO 15			REGISTRO 25	

Após a classificação do arquivo, tem-se a ordem de classificação do Arquivo Permanente. Basta copiar os registros do Arquivo Permanente de acordo com a sequência indicada no Arquivo Auxiliar: 18,41,....,1,....2,

Após copiado nesta ordem, deve-se suprimir o Arquivo Permanente inicial e trocar o nome do novo Arquivo gerado para que tenha o mesmo nome do Arquivo Permanente inicial.

Passos para a classificação:

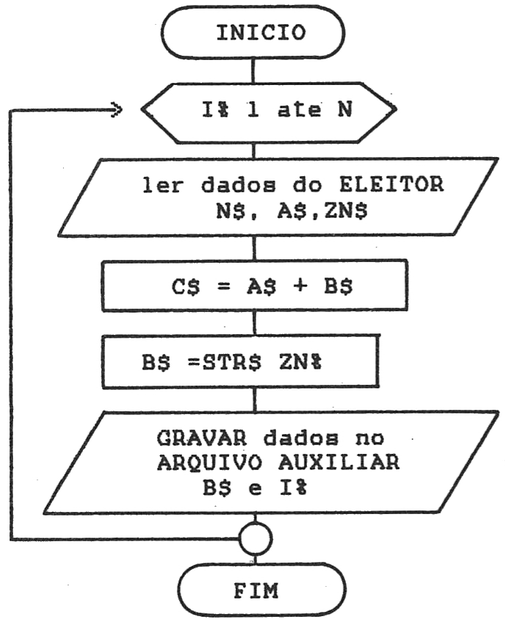
1. Criar um Arquivo Auxiliar com a nova chave de classificação e um número sequencial.
2. Classificar o Arquivo Auxiliar.
3. Copiar o Arquivo Permanente inicial de acordo com a sequência indicada no Arquivo Auxiliar.
4. Suprimir o Arquivo Permanente inicial.
5. Trocar o nome do arquivo-cópia.

EXEMPLOS:

1. Criação de um Arquivo Auxiliar com a nova chave de classificação e um número sequencial.

O fluxograma para criação deste Arquivo Auxiliar é bastante simples, como se observa a seguir:

O Arquivo Permanente é chamado de ELEITRES.
O Arquivo Auxiliar será chamado de ARQAUX.



NOME	- N$	-	30 BYTES
DISTRITO _	A$	-	20 BYTES
ZONA	- ZN$	-	2 BYTES
TOTAL			52 BYTES

B$	- 22 BYTES
I%	- 2 BYTES
TOTAL	24 BYTES

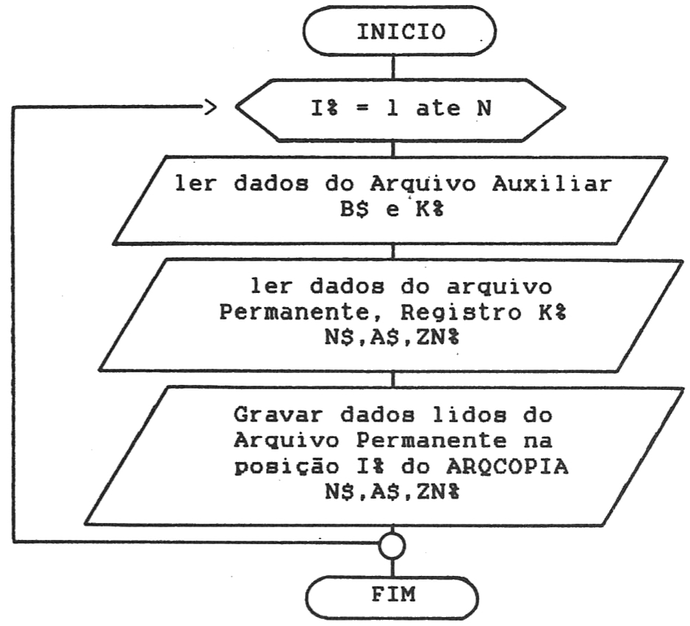
A codificação correspondente é apresentada a seguir:

```
 10  REM  PROGRAMA GERAUX PARA GERAR ARQUIVO AUXILIAR
 20  REM
 30  D$ = CHR$(4)
 40  PRINT  D$ "OPEN ELEITRES,L52"
 50  PRINT  D$ "OPEN ARQAUX,L24"
 60  HOME
 70  INPUT "QUANTOS ELEITORES NO ARQUIVO PERMANENTE";N%
 80  FOR I% = 1 TO N%
 90  PRINT  D$ "READ ELEITRES,R";I%
100  INPUT  N$,A$,ZN%
110  B$ = STR$ ZN%
120  C$ = A$+B$
130  PRINT  D$ "WRITE ARQAUX,R";I%
140  PRINT  B$
150  PRINT  I%
160  NEXT  I%
170  END
```

2. Classificação do Arquivo Auxiliar.

Este procedimento é exatamente igual à classificação com uma chave visto no Capítulo 7, motivo pelo qual ficará como um exercício adicional.

3. Copiar o Arquivo Permanente inicial de acordo com a sequência indicada no Arquivo Auxiliar.



A codificação é apresentada a seguir:

```
 10  REM PROGRAMA GERCOPIA PARA GERAR COPIA
 20  REM  DO ARQUIVO PERMANENTE
 30  REM
 40  D$ = CHR$(4)
 50  PRINT  D$ "OPEN ELEITORES,L52
 60  PRINT  D$ " OPEN ARQCOPIA,L52"
 70  HOME
 80  PRINT  D$ "OPEN ARQAUX,L24
 90  INPUT "QUANTOS ELEITORES NO ARQUIVO PERMANENTE"; N%
100  FOR I = 1 TO N%
110  PRINT  D$ "READ ARQAUX,R";I%
120  INPUT  B$,K%
130  PRINT  D$ "READELEITORES,R";K%
140  INPUT  N$,A$,ZN%
150  PRINT  D$ "WRITE ARQCOPIA,R";I%
160  PRINT  N$
170  PRINT  A$
180  PRINT  ZN%
190  NEXT  I%
200  END
```

4. Suprimir Arquivo Permanente inicial:

DELETE ELEITORES

5. Trocar nome do Arquivo - ARQCOPIA:

RENAME "ARQCOPIA","ELEITORES"

Neste ponto tudo se passa como se o Arquivo Eleitores original tivesse sido classificado. Na realidade, o novo Arquivo Elei-

tores é uma cópia ordenada do antigo Arquivo Eleitores, inclusive em outra posição física dentro do disquete.

EXERCICIO

O exercício a seguir pode ser considerado um teste final, para avaliar o domínio no manuseio de arquivos.

Montar um sistema para administrar os dados dos itens de um estoque. O sistema deve ter os seguintes programas:

a) Programa para criar o Arquivo Permanente: ESTOQUE.

b) Programa para classificar o Arquivo por equipamento e código de item.

c) Programa para adicionar novos itens ao Arquivo Permanente (para quem quiser, é possível englobar num mesmo programa os itens a e c).

d) Programa para listar dados do Arquivo Estoque.

e) Programa para saber a quantidade em estoque de certo item.

O registro para cada item deve conter os seguintes dados:

- Código do item.
- Nome do item.
- Uso Final - equipamento onde o item é aplicado.
- Quantidade em estoque.
- Custo unitário.

Como se observa, a empresa revende peças de reposição que são aplicadas em certo equipamento. Para aqueles que desejam uma sugestão, supor que um liquidificador tem as seguintes peças que o compõem:

- copo
- pás
- carvão p/ o motor
- interruptor etc.

O tempo previsto para a montagem deste sistema é de uma semana.

O resultado final deve conter:
- Listagem de todos os programas.
- Exemplo do processamento de cada um deles com dados criados pelo programador.
- Todos os programas devem ser auto-explicativos durante a execução.

impresso em offset ABRIL — 1986
artegráfica ind. e com. de artes gráficas ltda.
rua bresser, 1.299 - brás são paulo

PROGRAMAÇÃO COM TK 2000

Este livro é resultado do trabalho de uma equipe envolvida no ensino de programação para microcomputadores. Essa equipe constitui o Programa de Informática — PROINFO — e integra o corpo de profissionais do Fundo de Pesquisa do Instituto de Administração — FUNAD —, órgão complementar de ensino e pesquisa da Faculdade de Economia e Administração da Universidade de São Paulo.

Como um dos objetivos da Universidade é o apoio à comunidade, iniciou-se em 1984 uma série de três cursos voltados exclusivamente para o uso de microcomputadores. Esses cursos que deram origem a este livro foram modulados e exaustivamente testados com a finalidade de oferecer um texto prático e ajustado às necessidades de alunos e usuários de microcomputadores.

Para facilidade didática, o livro está dividido em três partes. A Parte I apresenta problemas classificados com lineares, não se introduzindo nenhum comando de desvio condicional. A Parte II inclui a solução de problemas mais complexos, incorporando o comando de desvio condicional e demais recursos da linguagem BASIC. Com o domínio da Parte II consegue-se resolver vários tipos de problemas, com diversos graus de dificuldades. A Parte III aborda os conceitos básicos de manipulação de arquivos, além de introduzir algumas noções sobre a montagem de Sistemas de Processamento de Dados.

NOTA SOBRE O AUTOR

ALOISIO PINTO ALVES é mestre em Administração pela Faculdade de Economia e Administração da USP (FEA/USP), onde está concluindo seu doutoramento. É professor-assistente da FEA/USP e professor titular da Faculdade de Administração da Fundação Armando Álvares Penteado (FAAP). Atualmente é diretor administrativo da FUNAD e do Instituto de Administração da FEA/USP, onde também coordena projetos na área de Informática. É supervisor administrativo da Unidade de Processamento de Dados da FEA/USP desde 1978.

APLICAÇÃO

Livro-texto para as disciplinas INICIAÇÃO À PROGRAMAÇÃO e LINGUAGEM BASIC dos cursos introdutórios de Processamento de Dados. Complementa o manual do fabricante, orientando na utilização de todos os comandos apresentados naquele manual. Proporciona os conceitos básicos para os usuários que pretendam a utilização mais intensiva do equipamento.

publicação atlas