

VICTOR MIRSHAWKA



BASIC
SEM SEGREDOS

BASIC

SEM SEGREDOS

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

M655b Mirshawka, Vítor, 1941-
BASIC sem segredos / Victor Mirshawka. — São Paulo : Nobel, 1983.

Bibliografia.
ISBN 85-213-0176-6

1. BASIC (Linguagem de programação para computadores) 2. Microcomputadores - Programas I. Título.

83-1572

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

Índices para catálogo sistemático:

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

Datilografia: Toshiro Iqueda
Ilustrações e Capa: Mauro Elme

VICTOR MIRSHAWKA

Professor titular de Cálculo Numérico, Estatística e Pesquisa Operacional da Faculdade de Engenharia da Fundação Armando Álvares Penteado.

Professor titular de Cálculo Numérico e Ciência da Computação, Estatística e Pesquisa Operacional da Universidade Mackenzie.

Professor associado do departamento fundamental da Escola de Engenharia Mauá.

Mestre em Estatística pela Universidade de São Paulo.

BASIC SEM SEGREDOS

4ª edição

1985



LIVRARIA NOBEL S.A.
EDITORA — DISTRIBUIDORA
LOJA 1: RUA DA CONSOLAÇÃO, 49 - CEP01301
LOJA 2: RUA MARIA ANTÔNIA, 108 - CEP01222
LOJA 3: RUA PEDROSO ALVARENGA, 704 - CEP04531
LOJA 4: RUA BARÃO DO TRIUNFO, 371 - CEP04602
EDITOR: RUA DA BALSA, 559 - CEP02910
FONES: (PABX) 857-9444 e 257-2144 - SÃO PAULO - S.P.

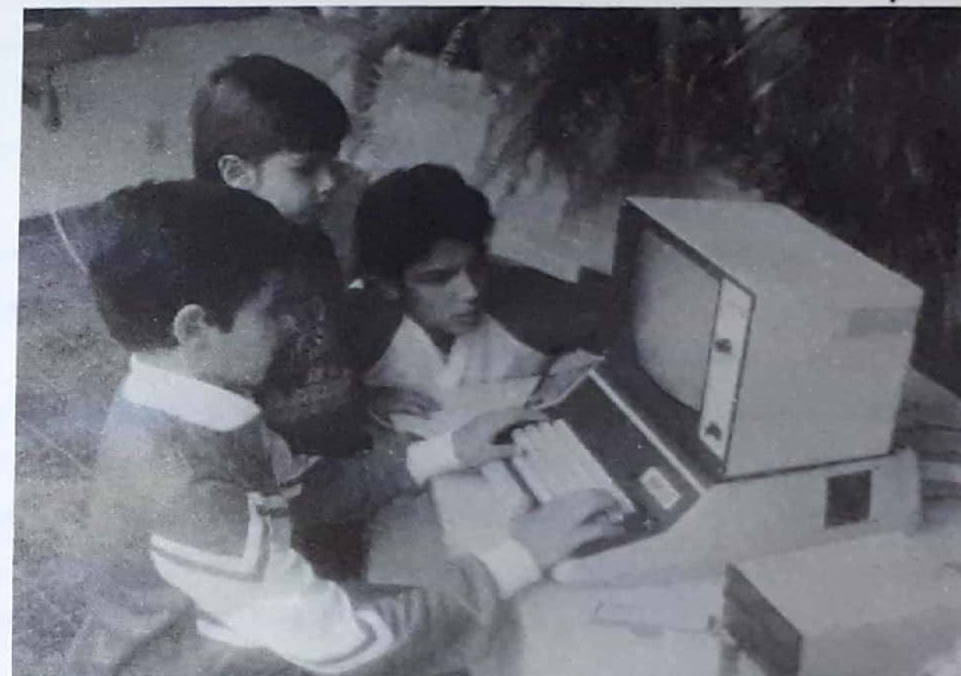
Copyright by Livraria Nobel S.A.

É PROIBIDA A REPRODUÇÃO

Nenhuma parte desta obra poderá ser reproduzida, sem a permissão por escrito dos editores, através de qualquer meio: XEROX, FOTOCÓPIA, FOTOGRÁFICO, FOTOMECÂNICO. Tampouco poderá ser copiada ou transcrita, nem mesmo transmitida através de meios eletrônicos ou gravações. Os infratores serão punidos através da Lei 5.988, de 14 de Dezembro de 1973, artigos 122-130.

Impresso no Brasil/Printed in Brazil

Dedicatória



Quando um homem se vê reproduzido num filho, o que entrevê é ser ele mesmo que renasce, e que sobreviverá a si próprio.

Será que não é este o conceito correto de infinito?

Dedico este livro aos meus três filhos, Victor Jr., Sergio e Alexandre, que vivem em uma das mais espetaculares eras da Humanidade e irão gozar de todos os benefícios da mesma - a era do computador.

Agradecimento

Em primeiro lugar quero agradecer à amiga e educadora Maria Dulce Dias Bastos com a qual conversei muito sobre os trabalhos de Mc Luhan, O'Neill, Lauro de Oliveira Lima, Orwell e outros homens ligados à Educação e aos destinos da Humanidade surgindo daí a idéia da mensagem que enviei ao leitor(a) no Capítulo 6.

Maria Dulce, espero não ter sido muito festivo e tão pouco inconsequente em relação aos malefícios que o microcomputador também poderá provocar.

Foi muito válido o incentivo dado a mim pelos amigos eng^o Sergio E.S. Fronterotta e eng^o José Maria Lamoglia para que o uso do microcomputador fosse implantado o mais rapidamente possível em todas as áreas de treinamento do pessoal ligado ao setor elétrico e especificamente nos cursos desenvolvidos pela Eletrobrás.

Isto ensejou a elaboração de programas bem específicos e alguns aparecem já neste livro-"BASIC SEM SEGREDOS".

O obrigado de sempre à amiga e companheira de trabalho prof^a Josefa A. Alvarez, que começou comigo uma nova experiência didática, qual seja, ensinar a linguagem BASIC para crianças de 10 até 14 anos e daí se originaram muitas das novas idéias de apresentação do material que compõe este livro, tornando-o acessível àqueles que têm pouca ou quase nenhuma cultura matemática.

É impressionante a rapidez com que as crianças aprendem tanto os problemas matemáticos como os não matemáticos no micro e as perguntas que fizeram a mim tiveram grande influência na elaboração deste texto.

Ao inseparável amigo e datilógrafo fenomenal Toshiro Iqueda que está já no seu terceiro livro de Computação de minha autoria (e virão outros brevemente) e que praticamente está tirando um curso de programador BASIC com veloci-

dade espantosa.

Esta vez Toshio você excedeu as medidas. Muito obrigado a Angélica Ayres, a Carlos Roberto F. Silva, a Denise Azevedo de Faria e ao criativo e excelente artista Mauro Elme, pois devo a vocês a excelente apresentação deste livro.

Vocês trabalharam com um empenho que deu-me a certeza de que o livro não é só da minha autoria, mas, sim, tornou-se parte de cada um de nós.

Expresso aqui a minha gratidão à Diretoria da Polymax Sistemas e Periféricos S.A., que colocou a nossa disposição todas as facilidades para rodarmos os programas no seu equipamento.

Aos amigos da Livraria Nobel S.A., Ary K. Benclo-wicz e Luigi Zamboni que tornaram concretas, mais uma vez, todas as condições para que este novo livro saísse com o melhor acabamento possível e no menor prazo de tempo.

Muito obrigado pelos recursos a mim fornecidos para que pudesse pesquisar e dessa forma apresentar um trabalho que espero possa impulsionar o uso dos microcomputadores no lazer, nos negócios e principalmente na Educação.

O Luigi até aulas de BASIC assistiu, para me estimular...

Aos meus filhos Alexandre, Sergio e Victor Jr., pelo desenvolvimento de alguns programas, sendo que os dois últimos, com 12 e 14 anos respectivamente, já colaboraram, e eficientemente, em muitas das aulas que lecionei em 1983.

Realmente este "triunvirato" pela rapidez com que domina e se desenvolve intelectualmente no micro, dá a certeza de que o mesmo será um grande bem para a Humanidade.

Os filhos, em geral, tornam-se para os pais, segundo a Educação que recebem, uma recompensa ou um castigo.

Creiam caros pais, leitores deste livro, que o micro entrando como componente da Educação dos seus filhos, facilitará para que o resultado seja a recompensa.

Finalmente restou a minha querida esposa Nilza Maria.

Ela, que além de ser a paciente revisora de todos os meus trabalhos submeteu-se desta vez a vários cursos de programação em BASIC inclusive a aquele que lecionei na Livraria Nobel S.A. nas férias de julho de 1983 e passou agora realmente a ser uma eficiente colaboradora nas alterações que efetuamos nas repetidas leituras do texto.

Recentemente, e com muita satisfação, recebi o recorte de um comentário feito no "Diário de Ribeirão Preto" (17/07/83) pelo crítico Oswaldo Lopes de Brito, que dizia que leu o nosso livro "Linguagem BASIC", principalmente pelo lírico agradecimento que fiz a Nilza Maria.

Caro Oswaldo Lopes de Brito, vivo realmente com uma criatura maravilhosa.

Há quem diga que deveríamos escolher para esposa a mulher que escolheríamos para amigo, se ela fosse homem.

E no que reside a função de um amigo na vida de cada um de nós?

Entre outras coisas, para nos levar a realizar o que podemos fazer.

Já houve também quem dissesse que "talvez eu fosse feliz se conseguisse esquecer o bem que pude e não fiz e o mal que fiz sem querer..."

Você, Nilza Maria, esposa e amiga, tem o crédito de ter minimizado as possibilidades de que eu fizesse o mal e maximizado as oportunidades para que eu terminasse o que tinha condições de fazer, e principalmente ter me dado com o seu amor a luz que não deixa escurecer a vida.

Aliás, viver sem ter sentido um verdadeiro amor, é ter deixado passar a vida sem viver...

Prefácio

No nosso primeiro livro "Linguagem BASIC" dizíamos, inicialmente que a finalidade principal do mesmo era dar suporte aos alunos da Faculdade de Engenharia da Fundação Armando Álvares Penteado (FEFAAP) para obter rapidamente a eficácia e a eficiência no uso dos microcomputadores, visto que estamos convencidos que desta forma se processa muito melhor o aprendizado de todas as disciplinas.

Não bastaria simplesmente termos escrito o livro, pois era necessária a aplicação, e para isto precisávamos de um Laboratório de Microcomputadores instalado e funcionando.

Nisto a Diretoria da FAAP, tendo a frente o ilustre casal Lucia e Roberto Pinto de Souza, não apenas incentivou a idéia da sua instalação como aprovou e continua dando os recursos financeiros necessários para a ampliação gradativa do mesmo adquirindo a Instituição, assim que surgem, as versões mais sofisticadas de micros.

Colaboramos, assim, apenas com a filosofia da Diretoria da FAAP de que aos jovens alunos de hoje deve-se maximizar a possibilidade de revelar novas informações de forma completa e no mais curto espaço de tempo.

Com isto, dentro do finito de um dia, deve sobrar para cada um mais tempo para meditar, para raciocinar, pois muitas das tarefas repetitivas e cálculos, antes muito morosos, acabam quase que instantaneamente tendo-se a mão um micro...

Neste segundo livro o intuito continua sendo o mesmo.

Hoje está reforçado, pois dentro da própria FAAP já surgiu um excelente livro sobre a linguagem BASIC, e cujo autor prof. Arsonval F. Pereira, é professor da Faculdade de Enge-

nharia.

Aliás no intervalo da publicação do primeiro e deste segundo do livro excelentes revistas foram publicadas como Video News, Interface, Microhobby, Micro Sistemas, Micromundo sem deixar de lembrar a excelente Nova Eletrônica.

Neste momento ao lançarmos o nosso 2º livro sobre o BASIC procuramos tornar mais ampla a informação do primeiro com exemplos totalmente diferentes.

Queremos com isto atingir também os alunos da Universidade Mackenzie em cuja Faculdade de Engenharia nos formamos, os alunos da Escola de Engenharia Mauá, na qual lecionamos desde 1966 e principalmente a todos, de qualquer idade e nível cultural que queiram manipular o *micro*.

O livro propriamente dito tem um 1º Capítulo destinado a informações descritivas sobre o computador, sua evolução e suas aplicações.

No 2º Capítulo desenvolvemos a linguagem BASIC de maneira que seja acessível a todos, bastando para isto ser alfabetizado.

É por isso, que as explicações são as mais amenas possíveis e aparecem programas para problemas sem grande complexidade tentando-se com isto tornar o micro fonte de conhecimento e principalmente de lazer.

Nos capítulos 3 e 4 fizemos o BASIC intermediário e um pouco do avançado.

Tentamos dessa forma atingir um detalhe maior explorando os vários recursos do micro sem nunca ter como meta levar o leitor ou leitora ao profissionalismo da programação.

Dedicamos o capítulo 5 a algumas aplicações gráficas, usando especificamente o MAXXI que é um micro da POLYMAX PERIFÉRI

COS E SISTEMAS S.A.

Aliás no início de 1984 sairá um livro específico para este micro.

Nos capítulos 2, 3, 4 e 5 entremeados ao texto aparecem muitos problemas que devem ser programados e que são oferecidos ao leitor ou leitora como exercícios a serem feitos por ele (ela) mesmo ou então servindo de base para que os professores que usarem o livro como texto apliquem os mesmos em testes de avaliação.

Finalmente no capítulo 6 analisamos o micro e as suas perspectivas na Educação.

Dada a variedade de terminais, micros e sistemas ora existentes é impossível incluir uma orientação geral.

As minúsculas diferenças que por ventura apareçam entre os comandos e instruções BASIC usadas no livro e aquelas que o leitor tenha no seu equipamento particular não devem impedir a utilização dos programas apresentados.

Aliás procuramos diversificar "invadindo" a linha do Apple através do sensacional MAXXI da POLYMAX e a linha do TRS-80 com o uso dos equipamentos da Prológica.

Referimo-nos também aos equipamentos da Microdigital e do Micro-Engenho da Spectrum.

Acreditamos que o eclético leitor ou leitora saberá rapidamente adaptar os programas a sua máquina, caso por exemplo ela seja um Atari.

O desenvolvimento da máquina a vapor foi traçado em séculos e o do avião em décadas.

Em cada caso os conceitos, no sentido de uma visão futura do seu uso, precederam em muito a realidade!

Nós estamos vivendo a era do desenvolvimento dos microcomputadores e é um processo que marcha de tal forma que visões e realidades se confundem e o ciclo da vida de algumas máquinas tem sido tão pequeno que sua morte pode ser prevista ao nascer.

A velocidade de criação e produção é tão grande que o conceito de desejável e exequível se confundem.

Isto às vezes é muito perigoso e o exemplo mais evidente disto é o Concorde.

Era viável então deveria ser desejável.

Muito esforço foi dispendido em criar e promover o Concorde e ficou em segundo plano o que os futuros clientes estavam desejosos de pagar.

E agora onde está o Concorde?

É um alerta muito importante para a indústria dos micros este ponto de vista altamente seletivo.

Não há nada de estático neste mundo em transição, sendo que as recentes pesquisas têm demonstrado que o "abominável ser discordante de tudo" está sendo em alguns casos substituído pelo "detestável procrastinador", ou seja, aquele que só adia.

O detestável procrastinador vive dizendo "tudo no devido tempo", vai demorando e não toma nenhuma decisão, atrasa tudo o que pode.

O atraso é a forma mais mortal de NEGATIVA.

Aliás C.N. Parkinson no seu livro "A lei do atraso" define atraso da seguinte forma

$$t = \frac{(P + n)^{\tau}}{3i}$$

onde P é o período de insistência ou vida útil da pessoa que

propõe a reforma, τ é o tempo decorrido entre a entrada da proposição e a sua solução final, n é o número de questões estranhas abordadas durante a discussão, i é a idade do "detestável procrastinador" e t é o tempo de demora ou atraso da implantação da idéia, reforma ou novidade.

No caso do computador, e mais especificamente do micro diremos que o período, P ou seja, a velocidade com que entram hoje as novas máquinas apresentando novas soluções e tornando as "últimas novas" obsoletas é curtíssimo, o número de questões estranhas (n) ao fato de precisarmos de um micro é mínima e não há necessidade de se perder muito tempo (τ) para que se aceite a solução informatizada, excluída a reação natural daqueles que acham que vão perder o emprego ou as vezes a "boquinha".

O jovem geralmente não entra na categoria do detestável procrastinador, porém mesmo que o i de alguns seja baixo (e este i não é o i do Q.I. - quociente de inteligência e sim i da idade) o t está ficando *infimo*.

Torna-se assim óbvia a razão de termos passado para o lado do reformador, prevendo a vitória e a aplicação indefinida do micro.

A lei de atraso de Parkinson será assim evitada no que se refere ao uso generalizado dos micros.

Da mesma forma que esta dentro da engenhosidade humana fazer o homem voar sobre a superfície terrestre e para longe dela (Gagarin, Glenn, Valentina, Sally, etc. que o digam) espero que este livro ajude (será que estou voando alto?) a frutificar novas idéias e que colocadas no micro faça a cada um dos leitores alçar um maravilhoso vôo além da sua imaginação...

Sumário

CAPÍTULO 1 - APLICAÇÕES E UM POUCO DE HISTÓRIA SOBRE OS COMPUTADORES

\$1.1 - Seja bem vindo ao primeiro contato com os computadores	1
\$1.2 - O computador e a sua história	9
\$1.3 - Um olhar mais detalhado sobre os microcomputadores e o seu equipamento	16
\$1.4 - O mínimo que cada um deve saber sobre os microprocessadores	23
\$1.5 - Preparação de um programa	31
\$1.6 - Apresentando o sistema BASIC e conhecendo algo mais sobre o Software	41
\$1.7 - Sistemas de processamento de informações	46
\$1.8 - Qualidades e fraquezas do computador	48

CAPÍTULO 2 - A LINGUAGEM BASIC ELEMENTAR

\$2.1 - A etiqueta do computador	53
\$2.2 - Expandindo um pouco o nosso elementar programa	55
\$2.3 - Fale comigo, oh maravilhoso micro! Você pode executar operações matemáticas?	63
\$2.4 - Notação científica e regras de precedência entre os operadores aritméticos	69
\$2.5 - Operadores relacionais - Instruções IF-THEN e GOTO	75
\$2.6 - O micro "fala" e escuta - A instrução INPUT..	82
\$2.7 - Os outros usos do PRINT	88
\$2.8 - O uso da fita cassete e do disco	94
\$2.9 - Enfim o par de instruções FOR-NEXT!!!	95
\$2.10 - Como é bom conhecer-te TAB!	106
\$2.11 - O filho, o neto, o bisneto, etc. do par FOR-NEXT	110
\$2.12 - As funções INT e SGN	119
\$2.13 - Mais instruções de desvio GOSUB-RETURN, ON-GOSUB, ON-GOTO	122
\$2.14 - O que são números aleatórios?	129

Aplicações e um pouco de história sobre os computadores

§1.1 - SEJA BEM VINDO AO MUNDO DOS COMPUTADORES

Os computadores, estão tendo um efeito fantástico no mundo de hoje, ajudando as pessoas a resolver qualquer tipo de problema desde o mais simples até o mais complexo.

Os computadores podem ser classificados nas seguintes categorias:

- I) super computadores;
- II) computadores de grande escala;
- III) computadores de média escala;
- IV) pequenos computadores para negócios;
- V) minicomputadores;
- VI) microcomputadores;
- VII) microprocessadores;
- VIII) computadores com finalidades especiais.

Todos os computadores consistem de uma unidade central de processamento (U.C.P.), dispositivos de armazenamento ou seja memória e unidades de entrada e de saída.

A UCP é composta de uma unidade aritmética/lógica e de uma unidade de controle.

Por sua vez a memória do computador é de dois tipos:

- 1) armazenamento principal
- 2) armazenamento auxiliar.

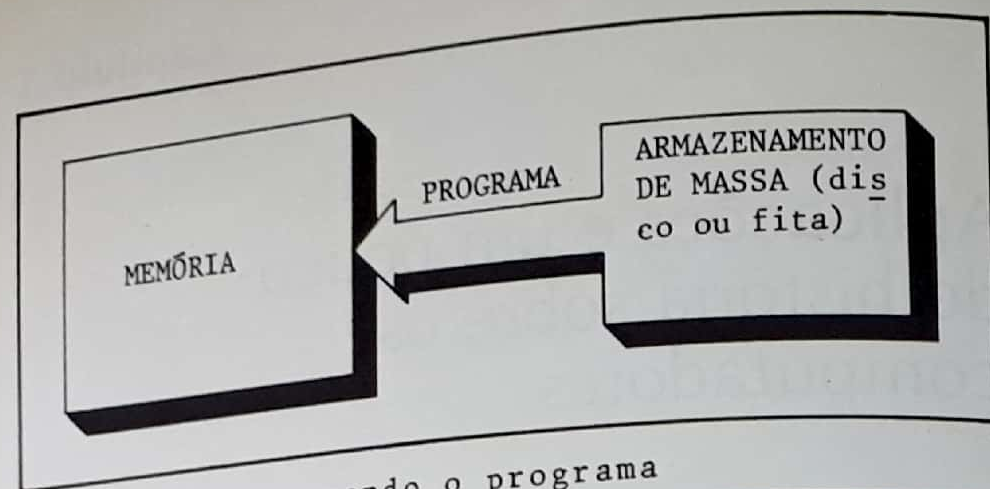
CAPÍTULO 3 - O BASIC INTERMEDIÁRIO OU SEJA DO MEIO DE CAMPO	
§3.1 - O par READ-DATA (mas quanto "parzinho" tem esse BASIC!!!)	141
§3.2 - Variável alfanumérica ou STRING	149
§3.3 - A necessidade da variável indexada ou subcrita	153
§3.4 - Funções matemáticas	171
§3.5 - As funções trigonométricas	187

CAPÍTULO 4 - O BASIC AVANÇADO	
§4.1 - O PRINT USING	203
§4.2 - O conjunto ASCII	218
§4.3 - Outras funções avançadas	220
§4.4 - Até que enfim o micro como artista. Ele como mestre dos gráficos e desenhos	242
§4.5 - As instruções PRINT@ e PRINT AT	251
§4.6 - O interessante e definitivo 2º tempo com a SET e com a RESET	255
§4.7 - Gráficos avançados	268
§4.8 - PEEK e POKE	285
§4.9 - Operadores lógicos AND, OR e NOT	289

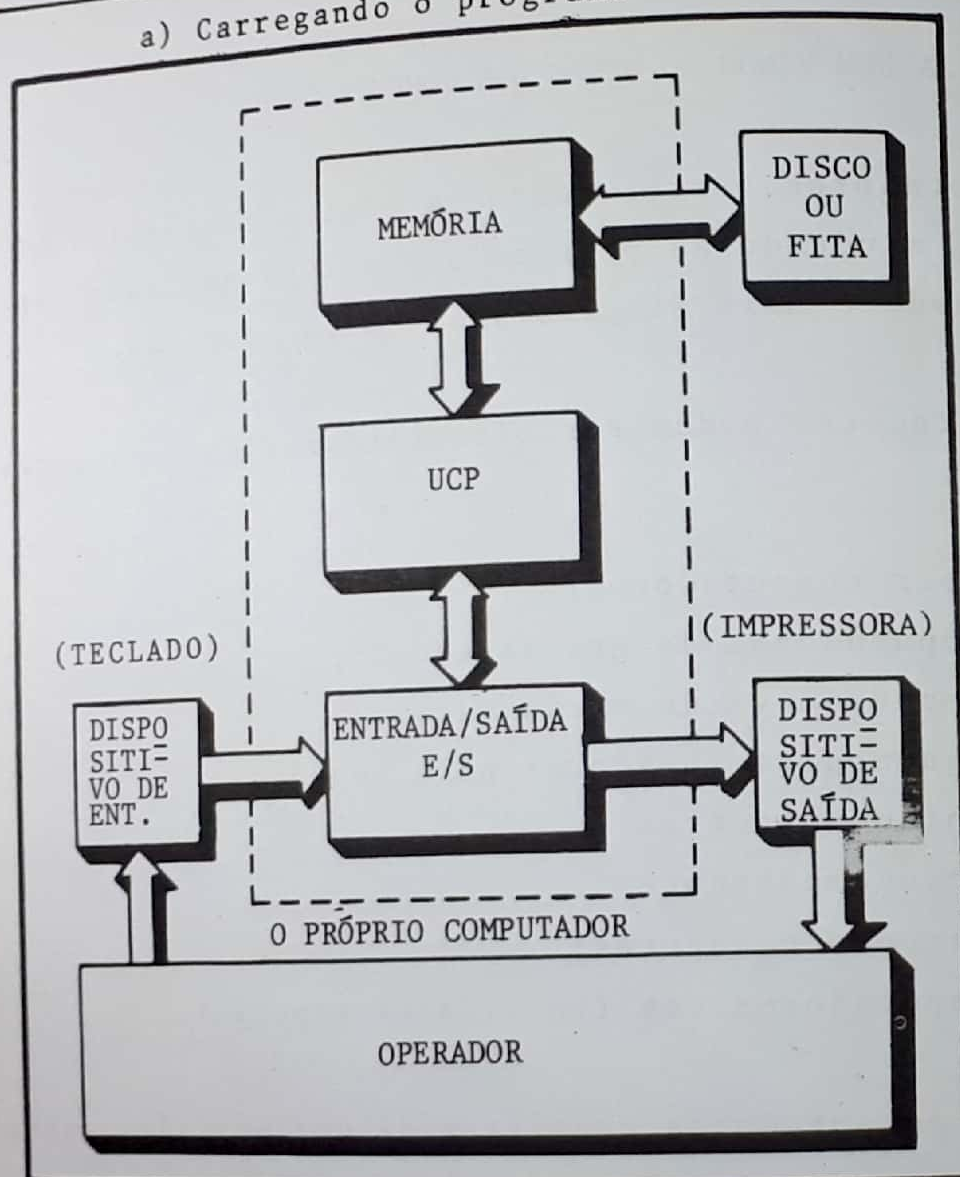
CAPÍTULO 5 - A HORA E A VEZ DE ALGUNS DETALHES SOBRE O SOFISTICADO MAXXI	
§5.1 - Gráficos de baixa resolução	301
§5.2 - Gráficos de alta resolução	320

CAPÍTULO 6 - O MICROCOMPUTADOR E A EDUCAÇÃO	
§6.1 - A época dos micros e o nosso sistema educacional vivendo ainda na época pré-Gutenberg	327
§6.2 - Os projetos I.A.C. e I.D.C.	332
§6.3 - O histórico mundial dos microcomputadores nas escolas	338
§6.4 - Formas de introdução do computador na educação - Experiências em diversos países	345

BIBLIOGRAFIA	359
ÍNDICE REMISSIVO	365



a) Carregando o programa



b) Modelo de um sistema de computador

Figura 1.1

A unidade elementar de memória é o bit (em inglês Binary Digit) e que significa que só pode assumir dois valores 1 ou 0, sim ou não, ou ainda, em eletrônica dois tipos de voltagens: "alta" (1) e "baixa" (0). Como vemos, um bit só pode fornecer dois códigos: (1) "bit on" e (0) "bit off".

Prosseguindo neste raciocínio, vemos que para um "bit" existem $2^1 = 2$ códigos, porém se tivermos 7 bits, existem $2^7 = 128$ códigos diferentes. Foi possível, então, desenvolver códigos, em que as letras, números e demais caracteres utilizados pelo homem em sua linguagem, correspondessem de forma biunívoca a códigos binários, contendo diversos bits. Conclui-se, pela conveniência dos 7-bits na representação dos principais caracteres. Nos microcomputadores, utiliza-se, normalmente, um código chamado "American Standard Code for Information Interchange" ou ASCII, onde as letras, algarismos e caracteres estão relacionados com as combinações possíveis de 0s e 1s contidos nos 7-bits. Assim, a letra A = 1000001; B = 1000010; C = 1000011, etc.

Internamente, os bits de informação são, em geral, organizados em grupos de oito. Na verdade, dos oito bits armazenados, apenas sete seriam necessários. Entretanto, por conveniência de projeto, preferiu-se utilizar oito bits ao invés de sete. Cada grupo de 8-bits é chamado um byte de memória. Um byte é, portanto, o conjunto de bits, (em geral, oito para os micros) que representa um caráter: letra, algarismo ou símbolo (veja a Figura 1.2).

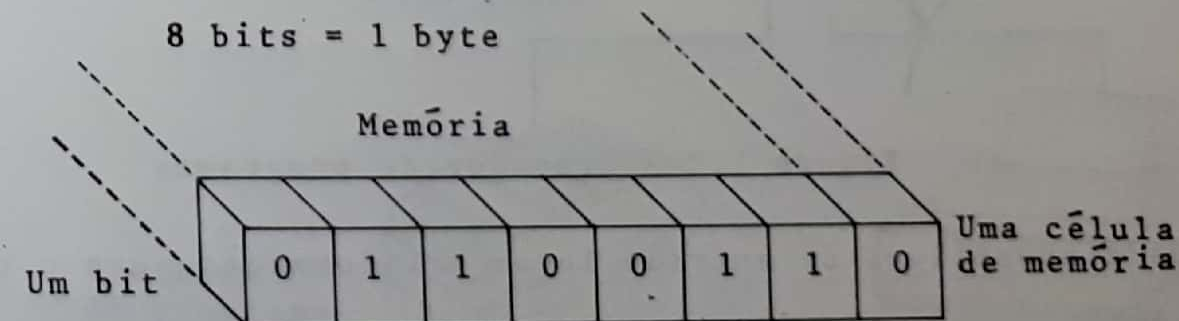


Figura 1.2 - Uma célula de memória

Os computadores do futuro terão uma memória de bolha ("bubble memory") que irá substituir tanto aos dispositivos de memória principal como os de auxiliar.

A memória de semiconductor é o tipo mais popular de memória principal que é usada nos computadores mais modernos.

Memória de núcleos magnéticos já foi entretanto muito usada nos computadores mais antigos.

Existem três tipos de memória auxiliar

- 1) disco magnético;
- 2) fita magnética;
- 3) tambor magnético.

Os "floppy disks" e fitas magnéticas cassette são hoje profusamente utilizadas nos sistemas de microcomputadores (Figura 1.3).

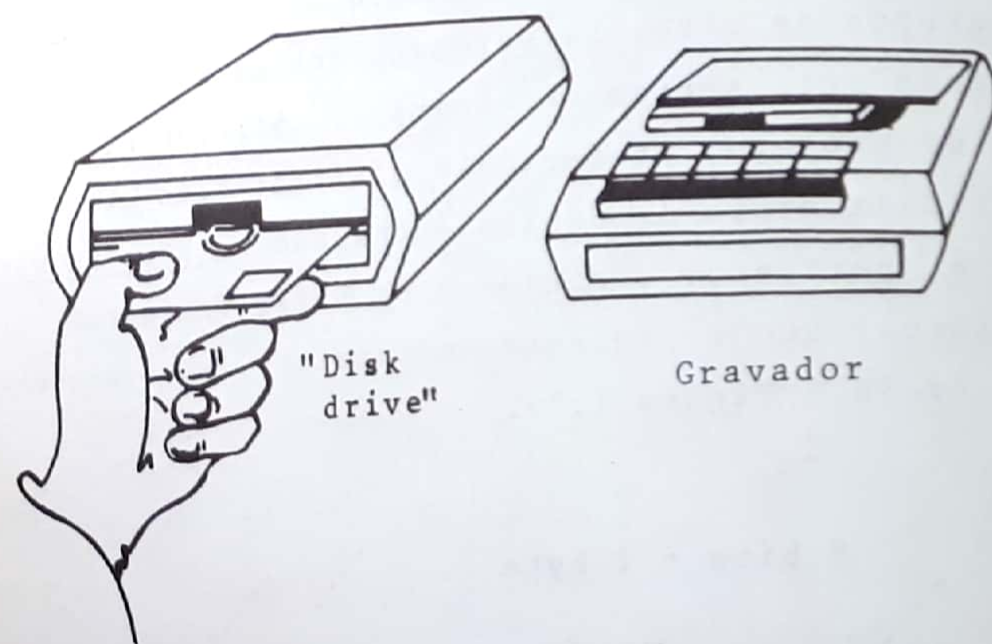


Figura 1.3 - Gravador de programas

Os dispositivos de entrada levam a informação para o computador.

Os dispositivos de saída são aqueles que retiram a informação da memória do computador para o uso humano.

Entre os dispositivos mais comuns de entrada/saída (E/S) destacamos as leitoras de cartões, exibição em um vídeo do aparelho de T.V., impressoras, traçadores, máquinas de escrever, etc. (Figura 1.4).

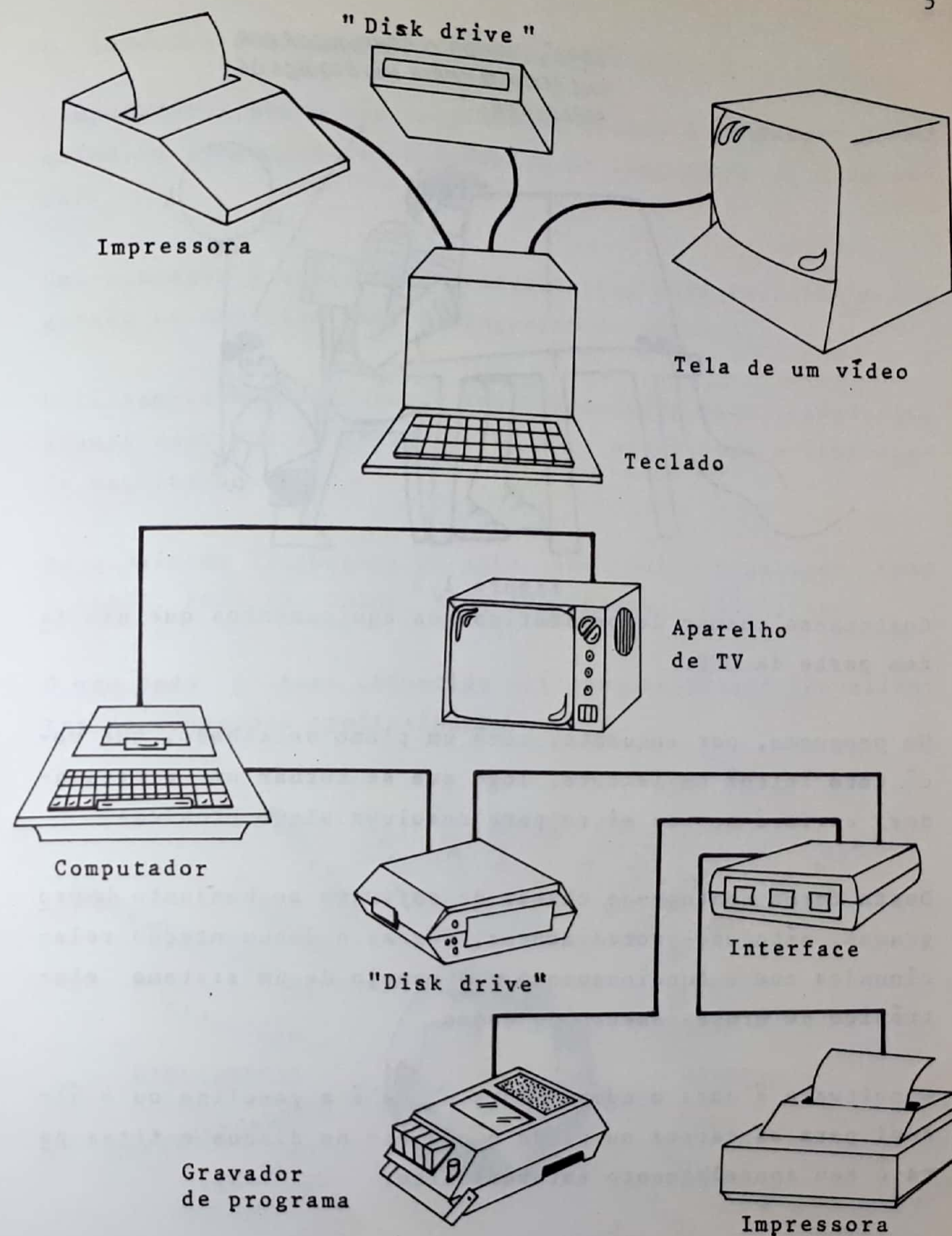


Figura 1.4 - Computador eletrônico

Portanto um sistema de computação consiste de uma UCP, de dispositivos de armazenagem, dispositivos de entrada/saída (E/S), console de controle, dispositivos de declaração de dados, sistema de operação e outros programas.

"PAPAI, QUANDO O COMPUTADOR DER
UMA RESPOSTA CERTA NÃO ESQUEÇA DE
AGRADECER..."



Figura 1.5

Costuma-se chamar de periféricos os equipamentos que não fazem parte da UCP.

Um programa, por enquanto, será um plano detalhado, que você caro leitor ou leitora, logo que se tornar um programador, enviará ao seu micro para resolver algum problema.

Desta forma costuma-se chamar de *software* ao conjunto de programas, métodos, procedimentos, regras e documentação relacionados com o funcionamento e o manejo de um sistema eletrônico de processamento de dados.

O software é para o computador, o que é a gasolina ou o álcool para os carros ou ainda o que são os discos e fitas para o seu aparelhamento estereofônico.

Sem o *software*, um computador é apenas uma massa não articulada de partes eletrônicas.

Por sua vez deve-se entender por *hardware* ao conjunto formado pelas máquinas de processamento de dados ou pelos elementos constitutivos das mesmas quer sejam do tipo mecânico, magnético, eletromecânico, elétrico ou eletrônico.

O *hardware* é assim o equipamento físico.

Computadores podem ser programados usando a linguagem de máquina, a linguagem de montador ou as linguagens de alto nível.

Uma montagem é usada para traduzir programas escritos em linguagem de montador para a linguagem de máquina.

Utilizam-se *compiladores* e *interpretadores* para traduzir programas escritos em linguagem de alto nível para a linguagem de máquina.

Hoje existem linguagens de alto nível muito populares como o BASIC, FORTRAN, COBOL, RPG, APL, Pascal, ALGOL, etc.

O uso cada vez mais difundido dos computadores criou milhares de ocupações profissionais.



Figura 1.6

Algumas dessas profissões estão diretamente ligadas ao computador e assim surgiram os analistas de sistema, programadores, operadores, perfuradores, diretores de centros de computação, vendedores de computadores, engenheiros eletrônicos, etc.

Porém a melhor forma de perceber isto é na Figura 1.6, onde se nota a variedade de setores onde é fundamental o uso de computadores.

As mudanças provocadas pelos computadores estão ocorrendo muito depressa e as tendências das mesmas se movimentam em cinco direções como mostrado na Figura 1.7.

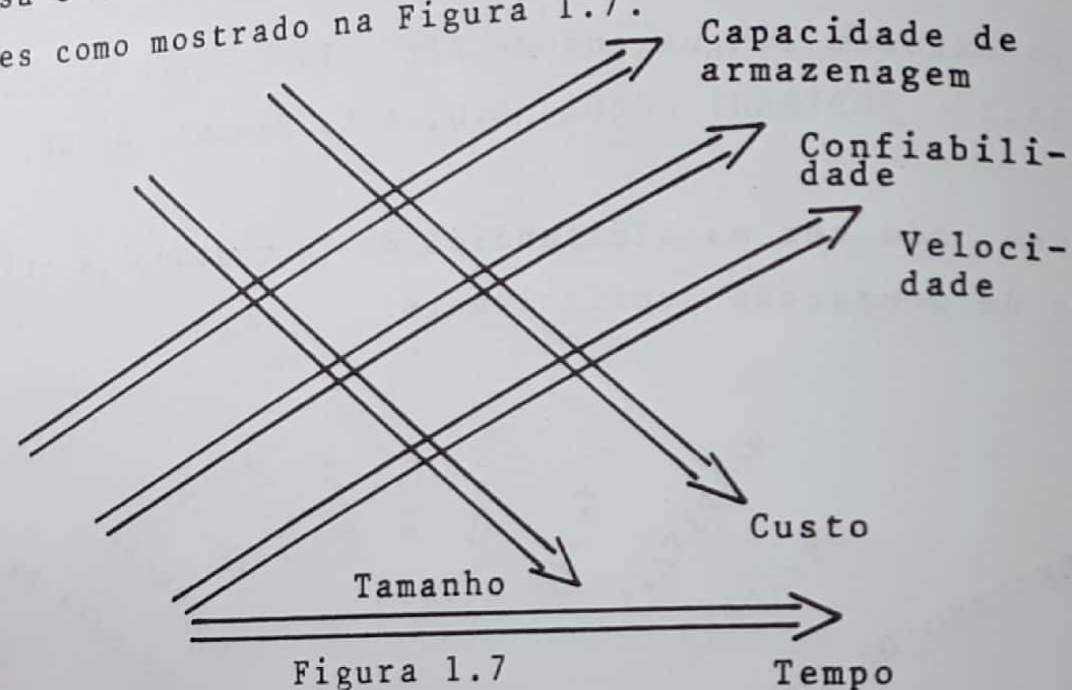
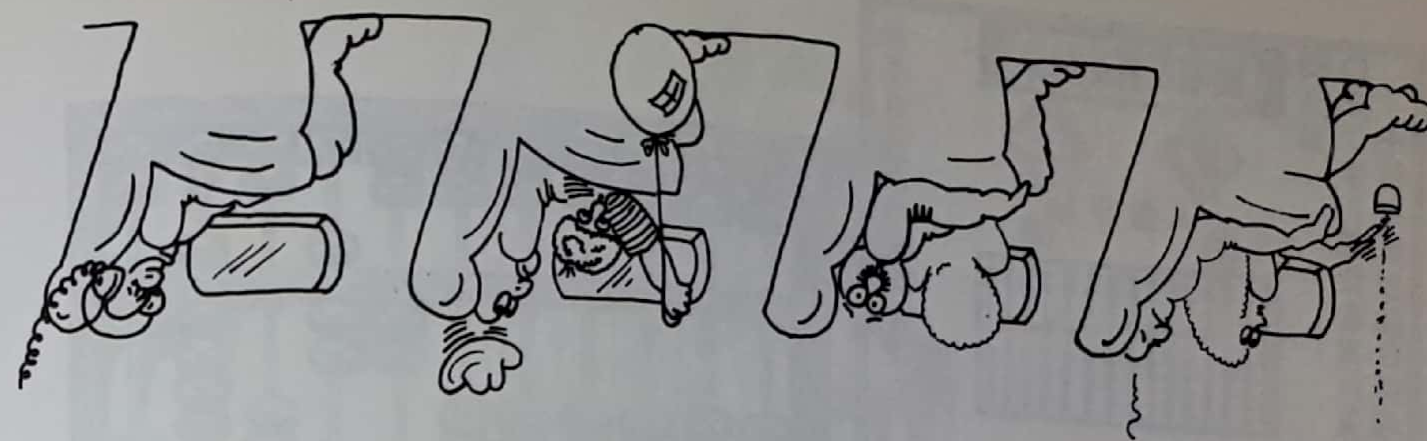


Figura 1.7

A tecnologia de computação é hoje cada vez mais dinâmica e pode-se dizer que o computador mais novo e moderno de *ontem* é uma peça de museu do *amanhã*.

Uma inovação neste ano será obsoleta no próximo ano.

Os computadores estão se tornando cada vez mais baratos e menores, enquanto ao mesmo tempo tornam-se mais confiáveis, mais rápidos e capazes de armazenarem ou seja guardarem mais coisas.



FALANDO O SEU CAPITÃO: A ATERISSAGEM ESTA HANNA É FEITA PELO SEU NOVO CO-PILOTO, O COMPUTADOR!!!

Figura 1.8

§1.2 - O COMPUTADOR E A SUA HISTÓRIA

O desenvolvimento de instrumentos para ajudar as pessoas a calcular começou com os primórdios da humanidade e elas usaram para este fim pedras, ossos, marcas na areia, nós nas cordas, etc.

Mais tarde (ufa) elas perceberam que tinham dedos e até hoje usam os mesmos para executar contas mais simples.

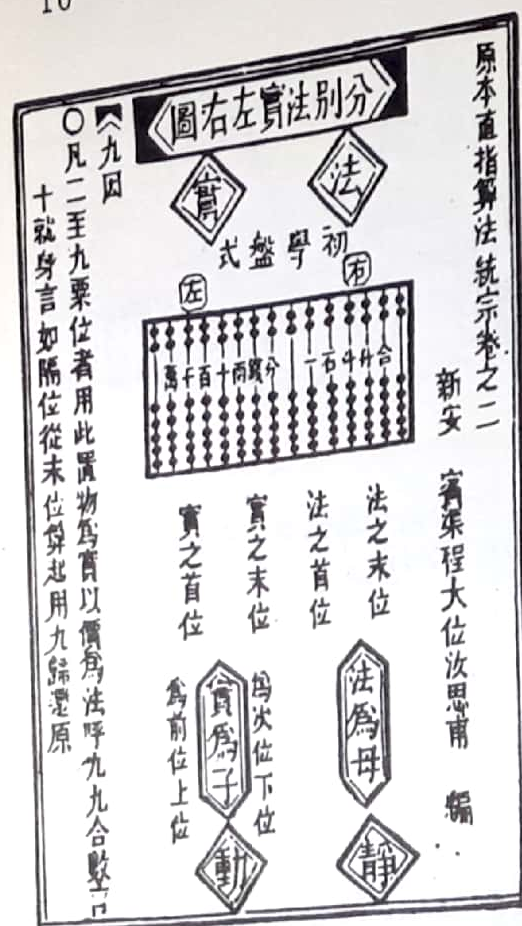
Um dos dispositivos mais simples, criado para o cálculo, foi o ábaco.

Lembra quando você era realmente uma criança e brincava com "contas" ou seja ábacos.

O ábaco, este instrumento tão antigo, foi usado nos últimos dois mil anos e mesmo hoje é ainda muito usado em alguns países orientais.

Você nunca ouviu falar do soroban japonês?

Foi apenas no século XVII que John Napier desenvolveu um aparelho bastante engenhoso que permitia multiplicar e dividir.



a) Uma representação primitiva do ábaco do Suan Fa
Thung Tsung, 1593

Figura 1.9

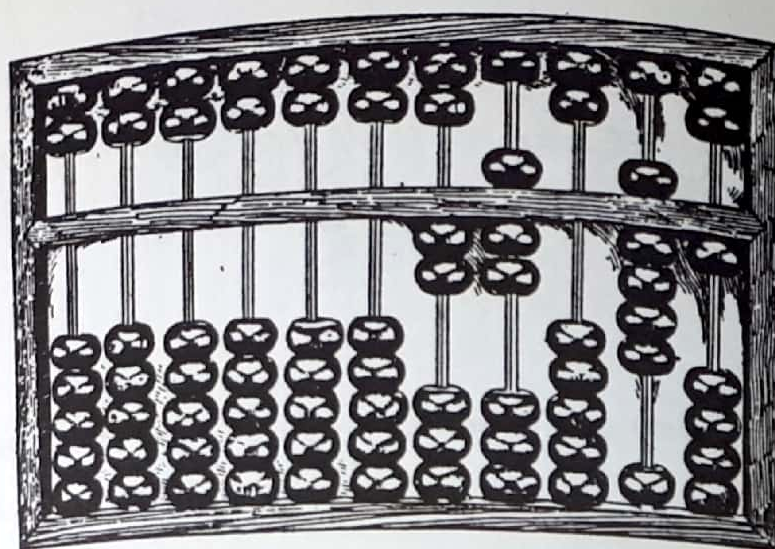
Porém a primeira máquina realmente prática, que permitia calcular, foi construída em 1642 e o seu inventor foi o matemático francês Blaise Pascal, na época com 19 anos.

A calculadora de Pascal estava limitada a executar apenas as operações de adição e subtração.

Mais ou menos 30 anos depois um famoso matemático alemão chamado Leibnitz desenvolveu uma máquina semelhante com a qual se podia também multiplicar e dividir.

Provavelmente o trabalho mais destacado no campo do cálculo mecânico foi feito por Charles Babbage.

Este matemático inglês projetou em 1833 a chamada "máquina das diferenças" e que viria a se tornar a precursora do moderno computador digital.



b) O ábaco chinês moderno



Figura 1.10

A máquina de Babbage nunca foi construída em vista de problemas de Engenharia, contudo o seu projeto permitiu a construção das máquinas de calcular.

De Babbage poderia se dizer que era um homem que viveu em uma época que não o entendia...

Foi no século XIX que surgiu a primeira máquina comercial de somar que foi inventada por William Burroughs (1884).

É interessante destacar que cerca de vinte anos após a morte de Charles Babbage já surgiram os cartões perfurados.

Os cartões perfurados foram inicialmente utilizados no controle de processo na indústria de tecelagem.

Em 1887, Herman Hollerith empregou um equipamento com cartões perfurados no trabalho do censo dos Estados Unidos.

Em 1907, James Powers desenvolveu um outro sistema de cartões perfurados que foi aplicado no censo dos Estados Unidos em 1910.

As máquinas desenvolvidas por Hollerith e por Powers foram melhoradas e usadas durante muitos anos.

Estas máquinas foram as precursoras dos sistemas de processamento de dados eletromecânicos que ainda hoje se usam.

Charles Babbage plantou a semente em 1833 e apenas em 1938 é que George Stibitz, nos laboratórios da Bell, conseguiu construir o computador eletromecânico.

Nos anos de 1937-1944, Howard Aiken da Universidade de Harvard liderou um grupo de engenheiros no projeto que levou a construção daquilo que veio a se chamar o computador Mark I.

Esta grande máquina usou um programa que permitia que se fizesse uma operação de adição em 0,3 segundos.

O Mark I foi usado por mais de quinze anos. Comparado com os modernos computadores era muito lento e tinha uma capacidade de memória muito pequena.

De qualquer forma foi o primeiro computador eletromecânico.

Em 1942, surgiu o ABC, construído por John V. Atanasoff.

Em 1943 um computador chamado COLOSSUS entrou em operação na Inglaterra.

Em 1946 desenvolveu-se na Universidade da Pennsylvania o primeiro ENIAC.

Na década de 1940 várias máquinas eletrônicas foram desenvolvidas para executar cálculos.

Foi somente em 1951 que os computadores passaram a ser algo diferente de "instrumentos" especificamente construídos para executar cálculos.

Nessa época (início da década de 1950) as máquinas UNIVAC I já existiam e foram muito utilizadas em aplicações de processamento de dados.

Logo após a guerra na Coreia, a IBM começou a fabricar máquinas e as primeiras foram as de números 650, 701, 702, 704, 709, 7090 e 7094.

Foi mais ou menos a partir de 1965 que os computadores assumiram uma grande importância em todos os países, principalmente nos industrializados.

As aplicações iniciais estavam nas áreas do governo e na administração, inclusive a escolar.

No período de 1965 a 1978 importantes computadores foram desenvolvidos pela IBM (sistema 360 e 370), pela Intel (microprocessador 8080), pela Radio Shack (TRS-80), pela Apple, pela Burroughs, pela Siemens, etc.

Os computadores da 1ª geração (aqueles de válvulas) eram muito volumosos e requeriam grandes quantidades de ar condicionado para dissipar o calor criado pelas milhares de válvulas.

Assim por exemplo o computador que se construiu na Universidade de Pennsylvania em 1946 custou cerca de meio milhão de dólares, pesava 30 toneladas e ocupava o espaço de uma casa de tamanho médio.

Suas 18500 válvulas especiais consumiam 130 kwatts.

A 2ª geração dos computadores era com transistores. Estes eram mais confiáveis e necessitavam menos energia.

O seu domínio foi aproximadamente de 1959 a 1963.

A 3ª geração de computadores, de 1964 até os primeiros anos da década de 1970, caracterizou-se pela *avançada miniaturização* (circuitos integrados).

Foi a Intel Corporation de Santa Clara a pioneira no desenvolvimento das técnicas de miniaturização, capazes de colocar, em um minúsculo pedaço de silício, o correspondente a milhares de válvulas de um computador.

Esta pequena placa de silício, com circuitos integrados de proporções reduzidas, denominada *microchip*, pode ser considerada na realidade, a responsável pelo grande salto tecnológico que desencadeou o processo de informatização da sociedade...

A 4ª geração dos computadores está trazendo violentas mudanças em todas as áreas e a que vamos destacar mais é a *área da EDUCAÇÃO*.

Introduziu-se o microprocessador e os circuitos tornaram-se microminiaturizados (Figura 1.11).

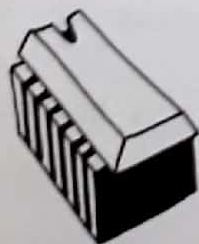
1ª Geração
Válvulas



2ª Geração
Transistores



3ª Geração
Circuitos
integrados



4ª Geração
Circuitos
micro mini-
aturizados

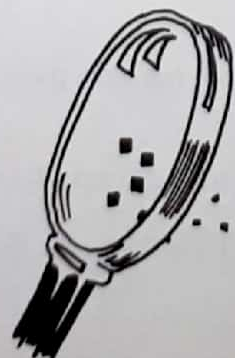


Figura 1.11

Já há quem discuta a 5ª geração e as outras que teremos após a mesma.

E é assim que já se começa a falar na memória de bolhas ("bubble memory") e na junção de Josephson.

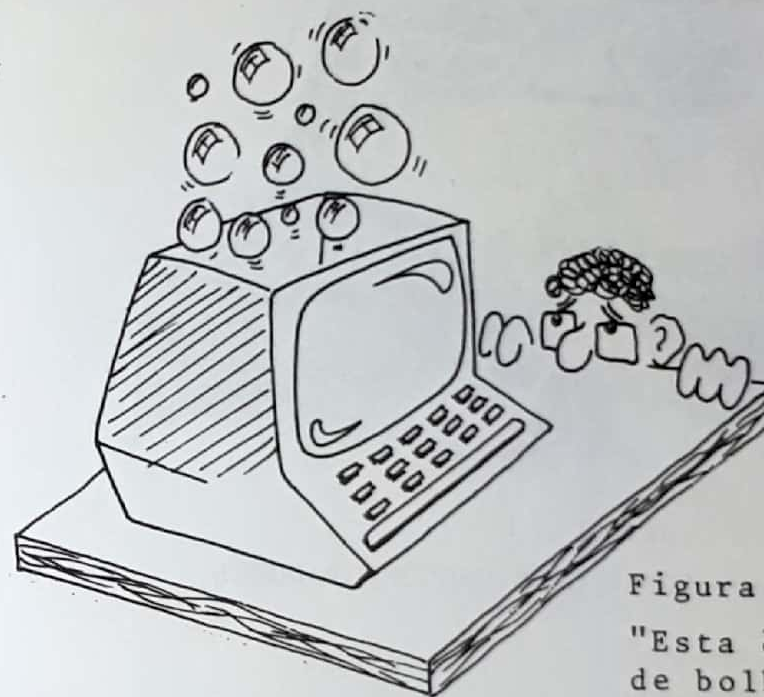


Figura 1.12

"Esta é a nossa nova memória de bolha..."

Computadores usando elementos de circuitos em junção de Josephson podem ser embutidos em um espaço não maior que o de uma bola de futebol, estando imersos em hélio e já (provavelmente) entendendo a voz humana.

Os seus "chips" integrados poderão conter até um bilhão de componentes.

Há quem diga que isto será possível no início do ano 2000.

Isto se Nostradamus estiver errado...

Ah! Ah! Ah!

Não fique obnubilado com esta última assertiva tá!



Figura 1.13

\$1.3 - UM OLHAR MAIS DETALHADO SOBRE OS MICROCOMPUTADORES E O SEU EQUIPAMENTO

Embora os computadores difiram muito nas suas particularidades, eles são organizados de uma forma similar e são constituídos de quatro componentes principais:

- 1) A UCP que controla todas as operações do computador e executa as operações aritméticas e lógicas.
- 2) A unidade de memória, na qual são armazenadas as instruções para o computador e os dados.
- 3) A unidade de entrada.
- 4) A unidade de saída.

A UCP (unidade central de processamento) é composta de uma unidade de controle e de uma unidade aritmética/lógica.

As unidades de entrada e saída possibilitam a comunicação com dispositivos externos e com o meio.

As instruções que guiam a operação de um computador são chamadas de PROGRAMA ARMAZENADO.

Sem um programa, um computador não pode executar a mais simples das tarefas.

Com um programa, ele toma vida e desempenha a tarefa especificada pelo programa.

Os programas, como já dissemos são chamados de *software*, enquanto o computador e os periféricos são chamados de *hardware*.

A informação deve chegar ao computador codificada de alguma forma.

Geralmente a finalidade de um computador é a de processar dados, os quais podem ser de contas a pagar, gastos com reparos, vendas de produtos, etc.

Entre alguns dos meios usados para introduzir uma informação dentro do computador destacamos cartões perfurados, caracteres escritos com tinta especial, caracteres que podem ser lidos por processo ótico, "floppy disks", fita de papel, fita magnética, etc.

A memória principal de um microcomputador é constituída por blocos ou bancos de memórias, cada um construído com pastilhas de semicondutores integrados em larga escala.

Existem diversos tipos de memórias de semicondutores como as do tipo ROM (read only memories) que só podem ser lidas, do tipo PROM (programable ROM) que só podem ser gravadas fora do micro e do tipo RAM (random access memories) que podem ser lidas e gravadas (Figura 1.14).

Uma pequena memória RAM pode armazenar 4000 bytes ou 4 K bytes.

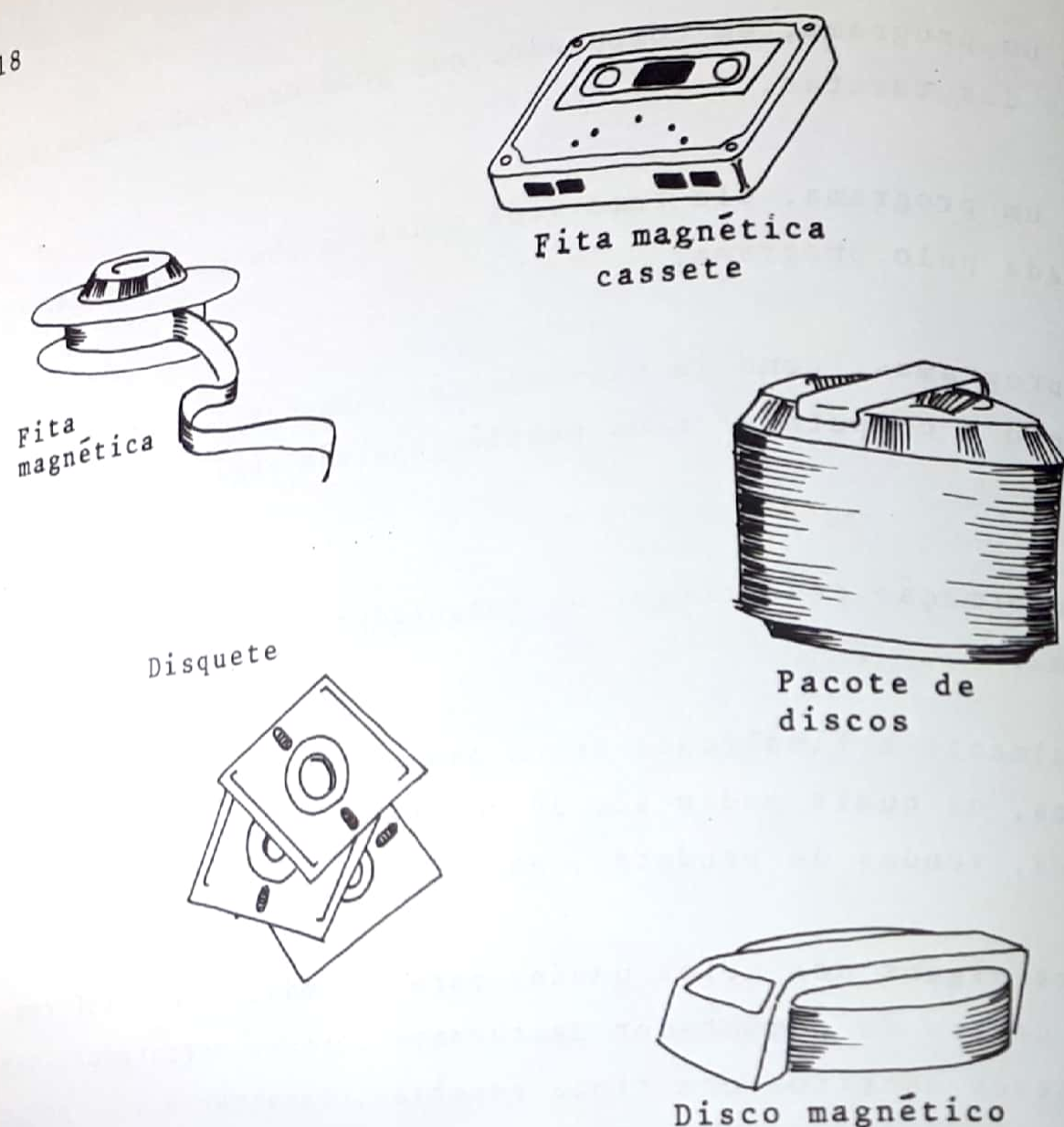


Figura 1.14 - Armazenamento em massa

Sabemos que $1\text{ K} = 1000$, porém em computação utiliza-se $1\text{ K} = 1024$, assim 4 K bytes correspondem a 4096 bytes .

A memória RAM permite ao micro "ler" diretamente qualquer byte da memória ou "escrever" (armazenar) novas informações.

Outra característica importante da memória RAM é que os dados armazenados são perdidos todas as vezes que o microcomputador é desligado.

As memórias do tipo ROM podem ter também cada um dos seus bytes acessados diretamente.

Entretanto, não é possível ter novas informações "escritas" nelas pois as mesmas são somente "lidas".

Tem como finalidade armazenar programas ou dados de caráter permanente.

Quando o microcomputador é desligado as informações contidas nas ROMs não são perdidas.

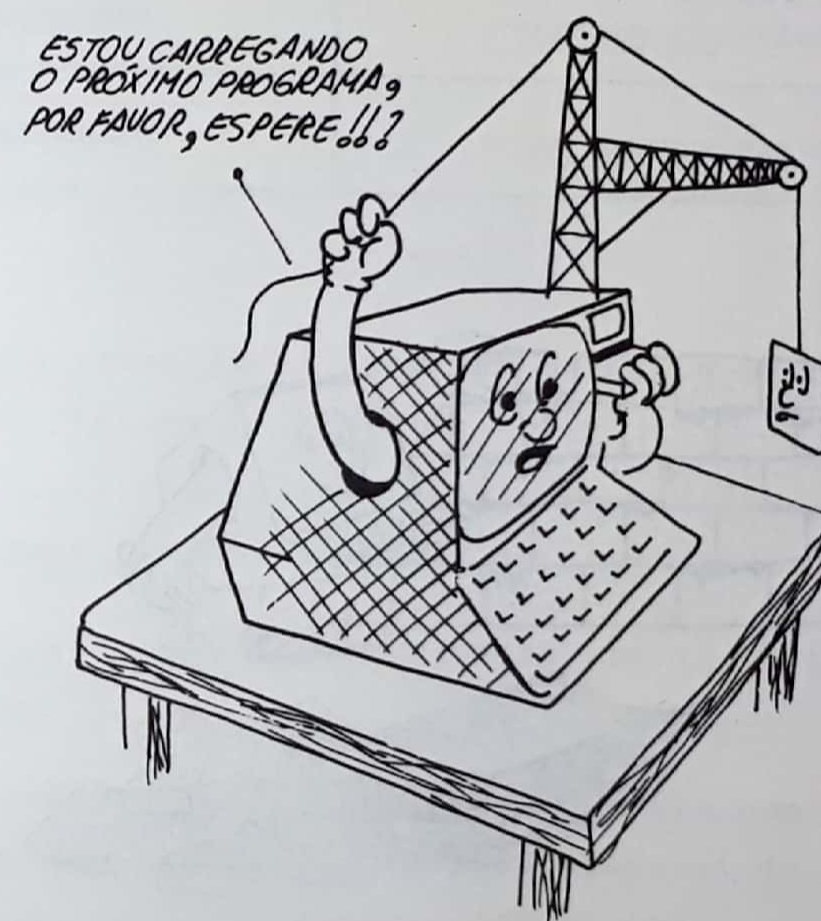


Figura 1.15 - A força de um micro repousa no seu "disquete"

Esses diversos tipos de memórias são combinados de forma conveniente, dependendo da função que tem, para constituírem os blocos de memórias.

Na memória principal residem os programas e dados que estão sendo presentemente processados.

Assim nos endereços que correspondem a memórias RAM são armazenados temporariamente os programas e dados do usuário.

Nos endereços correspondentes às memórias ROM e PROM são armazenados permanentemente os programas de supervisão do micro e programas de serviço, como por exemplo o interpretador BASIC. Por exemplo o CP-500 contém uma ROM de 16 K = 16384 bytes.

Há duas formas de acesso as memórias auxiliares:

- 1) sequencial
- 2) acesso direto

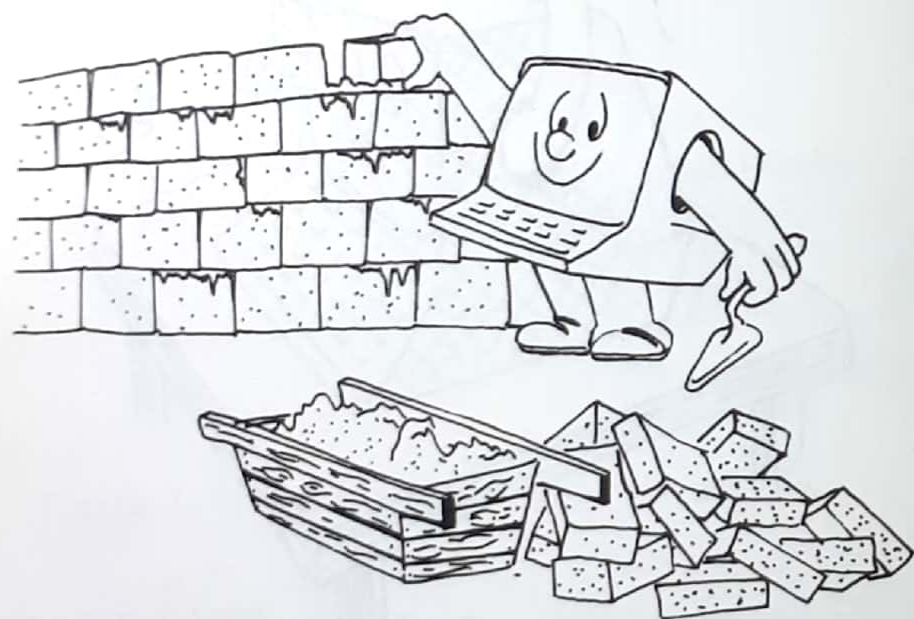


Figura 1.16

A velocidade de acesso às memórias principais é da ordem de nanosegundos e às memórias auxiliares é da ordem microsegundos.

Hum! Hum! Hum!

Quem falou que você sente algo pelo nanosegundo.

Bem não era para sentir, porém para perceber a sua pequenez (do nano segundo) veja a Tabela 1.1.

Unidade de tempo	Equivalente em forma de fração
1 segundo	1 segundo
1 milissegundo	$\frac{1}{1000}$ segundo
1 micro segundo	$\frac{1}{1\ 000\ 000}$ segundo
1 nanosegundo	$\frac{1}{1\ 000\ 000\ 000}$ segundo ← Oi ele!!!
1 picrosegundo	$\frac{1}{1\ 000\ 000\ 000\ 000}$ segundo

Tabela 1.1

As memórias auxiliares aumentam de muito as facilidades que o sistema oferece ao usuário.

Tornam possível armazenar *permanentemente* os programas e, eventualmente, dados, na forma de arquivos.

As principais memórias auxiliares são as fitas e os discos magnéticos.

Os microcomputadores geralmente utilizam como unidade de fita magnética os gravadores/reprodutores de som, principalmente os de fita cassete (de áudio).

Existem alguns, entretanto, que têm dispositivos especiais para a leitura e gravação de cartuchos de dados, que são cassetes especiais para processamento de dados, com fitas de maior densidade e velocidade.

As unidades de disco mais usuais em microcomputadores são as de *disco flexíveis*.

Os principais periféricos de microcomputadores são as impressoras e as unidades de vídeo e o teclado.



Figura 1.17 - Deixe o seu micro metido.
Dê a ele mais disquetes...

As impressoras são dispositivos que imprimem caracteres, em forma gráfica e legível, sobre o papel (formulário contínuo).

Fundamentalmente, as impressoras podem ser *de impacto* ou *sem impacto*.

As impressoras de impacto são parecidas com as máquinas de escrever comuns, no contexto de que os tipos batem sobre uma fita "entintada" que por isso marca o papel.

Existem as *impressoras de linhas* e as de *caracteres* que como o próprio nome indica, imprimem respectivamente, uma linha por vez e *caractere a caractere*.

As impressoras sem impacto mais comuns operam *termicamente*.

As impressoras térmicas requerem o uso de papel especial.

A unidade de vídeo não precisa de maiores descrições por ser muito conhecida. É o aparelho de T.V. da casa de cada um.

O teclado também é um dispositivo que dispensa outra apresentação pois está disponível em máquinas de escrever, telex, etc.

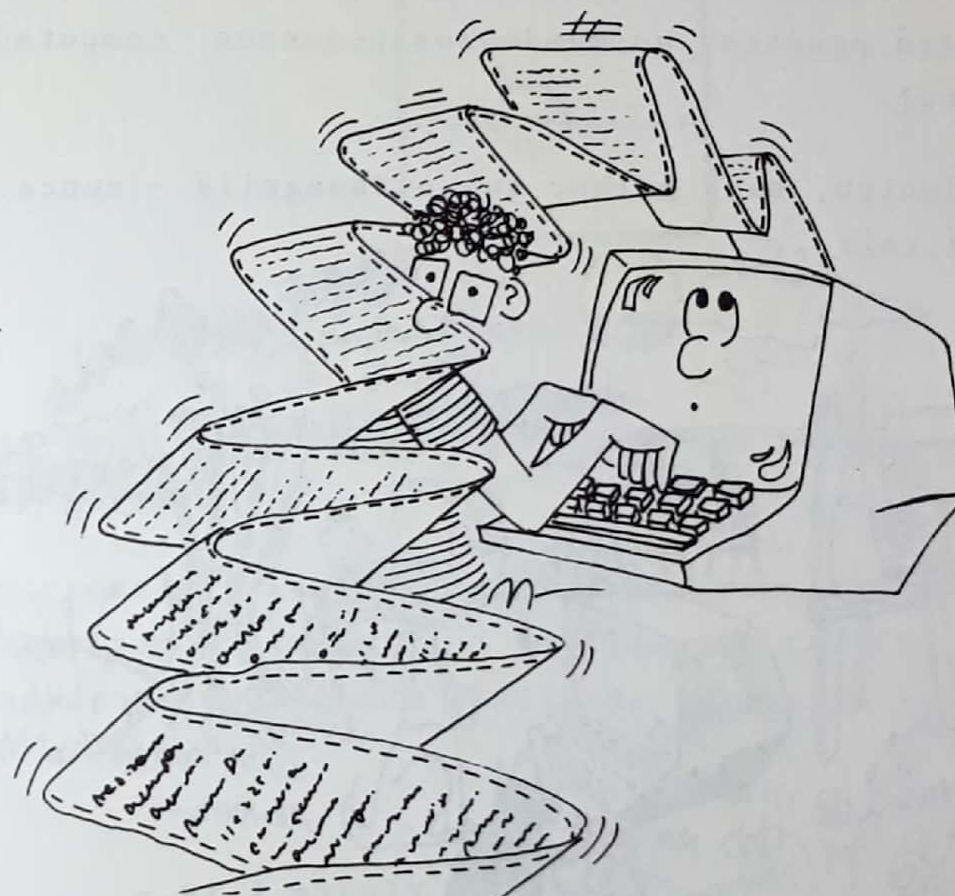


Figura 1.18 - Esperamos que você não faça este tipo de "mágica"!?!

§1.4 - O MÍNIMO QUE CADA UM DEVE SABER SOBRE OS MICROPROCESSADORES

Em poucos anos, a indústria dos microprocessadores cresceu muito e o custo dos mesmos foi muito reduzido de forma que este dispositivo está agora ao alcance de qualquer pessoa.

Hoje quase todo instrumento novo, controle ou algum sistema que possa "coleccionar" dados tem uma microprocessador dentro de si.

É por isso que achamos que não apenas engenheiros das diversas modalidades e pessoas ligadas ao trabalho técnico devem ter uma compreensão relativamente sólida dos princípios de funcionamento destes dispositivos.

O campo está aberto a qualquer um que queira aprender e com isto se poderá penetrar no mundo dos pequenos computadores (mini e micro).

Uma vez aí dentro, será melhor que o Shangrilã - nunca mais você quererá sair...



Figura 1.19 -
O Shangrilã dos micros

Inicialmente vejamos qual é o trabalho que o microcomputador deve executar.

Suponhamos que a tarefa seja a de relatar quanto de um certo líquido existe em um dado instante em um particular recipiente e então deve-se exibir um número ou uma mensagem que diga algo sobre isto.

O tanque tem um braço que "flutua" dentro do mesmo a medida que o nível do líquido se move para cima ou para baixo (Figura 1.20).

De que forma um microprocessador contribui para a operação deste sistema tão simples?

Microcomputador

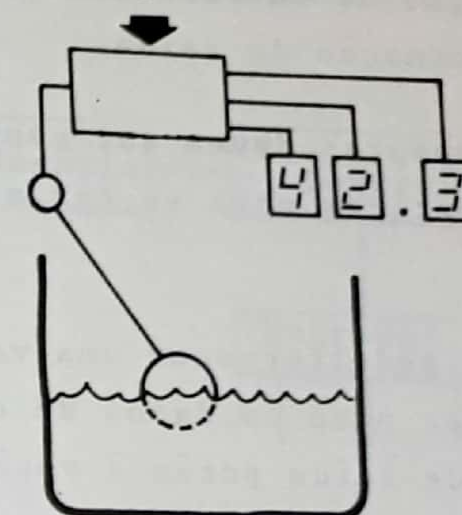


Figura 1.20

Começaremos com a descrição de como se parece um "chip" de microprocessador.

As entradas e as saídas de um microprocessador são voltagens que podem ser ou zero ou cinco volts. (Figura 1.21) e que aparecem em algumas dúzias de terminais em torno do seu bordo exterior.

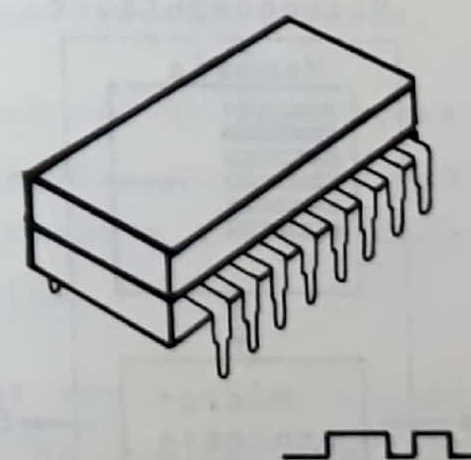


Figura 1.21

Desta maneira se alguém quiser utilizar os sensores como entradas e a exibição na tela de um aparelho de T.V. como saída, é necessário combinar as voltagens e executar o conveniente arranjo de fios.

Geralmente existe a necessidade de efetuar algumas modifi-

cações na informação de entrada antes que ela passe ou se transforme em informação de saída.

As entradas dos sensores devem ser convertidas em números ou mensagens alfabéticas para se ter a exibição "entendível" na saída.

O microprocessador pode fornecer uma variedade quase infinita dessas conversões para os dados de entrada e que mais tarde serão dados de saída porém é você que deve especificar exatamente o que quer.

Você poderá fazer isto com sequências de comandos ou instruções simples que são colocadas em uma memória permanente dentro do microcomputador.

Essas "fileiras" de comandos simples ou instruções é o que se chama de *programa*.

A programação é hoje a grande atividade não apenas dos engenheiros e técnicos mas das pessoas em geral (Figura 1.22).

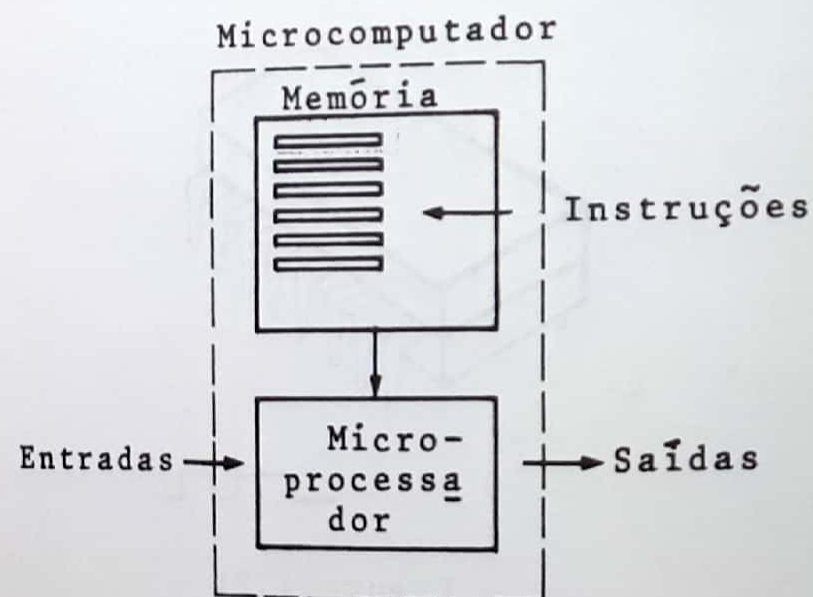


Figura 1.22

É por isto que você está lendo este livro, pois daqui a algumas páginas começará a aprender algo do "COMPUTADORÊZ".

Todos os computadores e já insistimos nisto, tem uma UCP, alguma memória, alguma forma de trazer os dados dentro dos

mesmos e após isto apresentar os resultados para o mundo exterior (Figura 1.23).

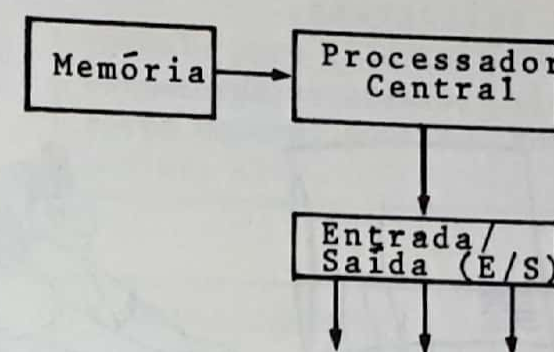


Figura 1.23

O que é peculiar a um microcomputador é que o processador é usualmente um "chip" e "talvez" a memória e os circuitos de E/S estejam sobre o mesmo "chip".

Microcomputadores não são apenas "pequenos" mas também são baratos no custo em relação aos "antigos" grandes computadores.

Tipicamente, pelo menos até uns três anos atrás, pensava-se que um microcomputador seria usado para o desempenho de somente uma ou algumas poucas tarefas.

Desta forma a sequência de instruções ou o programa era fixo, a amplitude do equipamento de E/S ao qual estava ligado o micro não era grande e a velocidade e a eficiência não eram tão importantes.

Um microcomputador seria assim encontrado dentro de um forno de microondas, de uma máquina de lavar roupa, de uma ferramenta de corte, de uma máquina registradora, etc.

O prof. José T. Senise escreveu em 1981 na Mauá em Revista, no seu primeiro número, um artigo muito interessante com o título "As microondas na cozinha".

Além das considerações técnicas sobre os fornos de microondas, as suas vantagens e o cuidado no seu uso ficou-nos na

cabeça um dado interessante: neste forno gasta-se ou melhor programa-se o mesmo para se assar um peru inteiro na base de 15 minutos por quilograma.



Figura 1.24 - Manda-se uma sequência de instruções a uma máquina e ela interrompe o seu trabalho de acordo com o programa recebido.

É claro que nos exemplos citados nós não precisamos de leitoras e perfuradoras de cartões de alta velocidade, de impressoras de linha, de fitas magnéticas ou ainda de memórias em disco.

Atualmente os microcomputadores se confundem com os mini e começam a substituir todos os grandes computadores que passam a ser grandes apenas no tamanho e maiores na idade da sua fabricação.

As tarefas que podem ser feitas pelos minis e micros que tem um console para a entrada, uma impressora de linhas e alguns disquetes de fita vão desde o cálculo de taxas, análise de vendas, verificação de estoques, etc. até os seguintes serviços (Tabela 1.2):

(1) PROJETO DE ENGENHARIA		
ENTRADAS	PROCESSAMENTO	SAÍDAS
Dimensão da ponte, terreno, rios, vao, etc.	Fórmula para calcular estruturas, materiais, força humana, equipamentos, etc.	Valores sobre materiais, pessoal, prazos, etc.
(2) SIMULAÇÃO		
ENTRADAS	PROCESSAMENTO	SAÍDAS
Condições de tráfego, número de veículos, número de semáforos, dimensões das vias, etc.	Cálculos: fluxo de carros, mudança de semáforos, etc.	Estatística sobre o comportamento do tráfego.
(3) CONTROLE DE PROCESSOS		
ENTRADAS	PROCESSAMENTO	SAÍDAS
Leitura por equipamentos: temperatura, pressão, nível de substâncias, etc.	Calcula-se e controla-se a mudança de condição, valores das substâncias, etc.	Valores calculados: gráficos da situação, avisos das condições críticas.

Tabela 1.2

Como se vê [principalmente da Tabela 1.2] o microcomputador cujo elemento fundamental é o microprocessador tem muitas aplicações e as suas dúzias ou talvez centenas de terminais podem ser facilmente conectados por fios que nos levam a sensores e a equipamentos de saída.

Temos assim sensores de pressão e temperatura, sensores dos níveis de líquido, detectores de luz, motores elétricos, válvulas, selenóides (Figura 1.25).

Neste ponto começa a surgir em você caro leitor ou leitora a preocupação de que um microcomputador precisa de um grande conjunto de instruções para se poder "ensiná-lo" a executar uma tarefa útil.

Naturalmente o que todos nós realmente desejariamos é que sem muito esforço esse conjunto de instruções pudesse ser codificado de uma forma universal e com o mínimo de erros e isto forçasse o micro a nos entender e a trabalhar.

A meta será atingida porém, infelizmente, não de uma forma muito simples.

Uai é daqui para frente que vai ficar complicado?

Devagar morena, calma brejeiro, tudo será explicado tim-tim por tim-tim e no fim você estará perfeitamente habilitado.

Bem para concluir diremos de uma forma mais técnica que o microcomputador usual é um quadro com um circuito impresso com dezenas de componentes no mesmo, precisando de uma fonte de energia separada (geralmente cara) e tendo como coração do sistema um pequeno "chip" chamado **MICROPROCESSADOR** que por sinal não é muito caro.

E aí vai a definição de CHIP

Chip é uma microplaqueta.

Nos circuitos microminiaturizados é um elemento isolado, de dimensões muito reduzidas podendo consistir de um ou mais transistores de diodos, formado por uma plaqueta de silício recortada de uma folha desse mesmo material (Figura 1.26).

Lamentavelmente, programar o microcomputador é ainda uma tarefa que consome tempo e é um processo no qual podem ocorrer

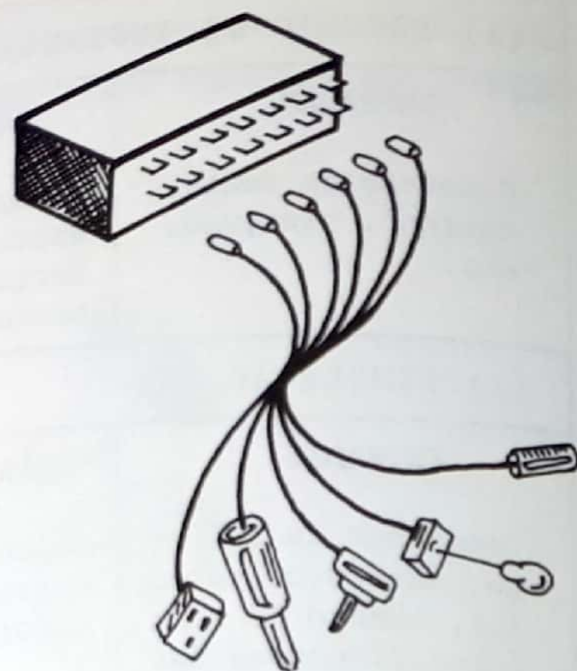


Figura 1.25

rer erros quase sempre feitos em uma linguagem diferente da que habitualmente estamos acostumados.

Uma das formas clássicas para programar um micro é tentar representar a solução do problema em um fluxograma (Figura 1.27).

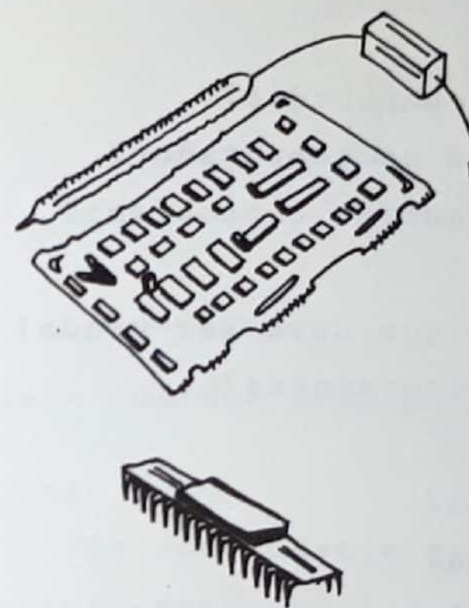


Figura 1.26

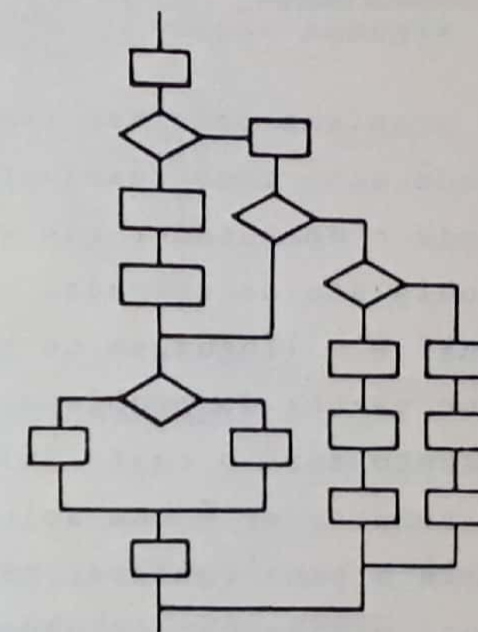


Figura 1.27

Neste livro, por incrível que pareça iremos fugir o mais que pudermos dos fluxogramas, apesar de dedicarmos a eles um parágrafo e mesmo assim acreditamos que os impetuosos e ávidos leitores aprenderão a programar.

§1.5 - PREPARAÇÃO DE UM PROGRAMA

Programação é um processo no qual um conjunto de instruções é produzido para um computador para fazer com que o mesmo execute uma específica tarefa.

O conjunto de instruções produzido é um *programa*.

Os passos para preparar um programa são:

- 1) Analisar o problema
- 2) Desenvolver o fluxograma
- 3) Codificar o programa
- 4) Executar o programa
- 5) Testar o programa
- 6) Documentar o programa.

No item 1) muitas perguntas devem ser respondidas e entre elas algumas são:

- a) O problema pode ser resolvido no computador?
- b) Você sabe como resolver o problema no computador?
- c) Pode o computador que você tem resolver o problema?
- d) Quais são as entradas e as saídas?
- e) Qual é a linguagem de programação que deve ser usada?
- f) Que partes do problema já foram programadas?
- g) Quanto será o custo disso?
- h) O computador é uma solução prática?
- i) Vale a pena resolver este problema?
- j) Qual é a parte dos dados que deve ser manipulada para produzir a saída desejada?

Depois que questões desse tipo forem respondidas é que se pode determinar melhor se um problema deve ou não ser resolvido no computador.

Na condição de que o problema será resolvido no computador você deve conhecer a 1ª e a 2ª leis da programação.

O computador é capaz de coisas incríveis!!!

Porém ele não tem nenhuma inteligência e nenhum julgamento de forma que ele não pode decidir o que tem que ser feito.

Esperamos que jamais ocorra a situação mostrada na Figura 1.28.

É o computador tão inteligente que ele sabe o que fazer sem ter recebido todas as explicações detalhadas?

A resposta é NÃO, NÃO e NÃO.



Figura 1.28

É isto que nos traz a 1ª LEI DA PROGRAMAÇÃO.

1ª Lei

"Se você deseja que o computador faça alguma coisa particular, você precisa ensinar-lhe os detalhes exatos para fazer esta coisa particular".

1º Corolário - Se você não o ensina, ele não o faz.

2º Corolário - O computador fará aquilo que lhe for ensinado, mesmo que você não perceba isto.

Se você já aceitou o fato de você precisar ensinar ou dizer ao computador exatamente o que fazer, provavelmente está querendo saber "como" executar isto e o que nos esperamos é que não ocorra o que está mostrado na Figura 1.29.

Basicamente existe um procedimento muito simples para se escrever um programa para o computador.

- I) Definir os resultados ou seja decidir exatamente que informação deve exibir.



Figura 1.29

II) Examinar os dados de entrada ou seja estar seguro que os dados contêm tudo que se quer para produzir a saída desejada.

III) Obter um plano geral (às vezes chamado de algoritmo) para converter os dados de entrada na saída desejada (veja a Figura 1.30).

Um grande número de técnicas é comumente usado com este intuito, incluindo fluxogramas (a descrição é dada adiante).

IV) Expressar a solução em uma linguagem que o computador possa entender.

A isto se chama "codificar o programa"

Este livro trata de programas codificados na linguagem BASIC (Figura 1.31).



Figura 1.30



Figura 1.31

Muitas pessoas e entre elas estão os "pequenos futuros gênios" não gostam do trabalho preliminar de definir e resolver o problema.

Elas preferem ir diretamente para a codificação.

Contudo elas irão perceber que a medida que o problema torna-se mais complicado o programa não pode ser codificado até que o algoritmo de solução tenha sido desenvolvido.

O fluxograma é uma forma bem conveniente para expressar o algoritmo.

Isto nos leva a 2ª LEI DA PROGRAMAÇÃO.

2ª Lei

"Quanto mais tempo você gastar na definição da sua saída e em desenvolver a sua solução global, tanto mais rapidamente transcorrerá a tarefa da programação".

Com isto você evitará a situação indicada na Figura 1.32.



Figura 1.32 - "Eliminando as mancadas"

1º Corolário da 2ª lei

Resista a pressa de codificar!

Vejamos agora algo sobre o item 2 ou seja sobre os fluxogramas.

Já sabemos o que é um algoritmo mas uma forma simples de defini-lo é dizer que é um conjunto de instruções para fazer alguma coisa.

O fluxograma é a demonstração gráfica dos passos requeridos

na solução de um problema e por isto é a forma gráfica de um algoritmo.

Em certas situações é a melhor forma de expressar o que o computador deve fazer ou o que a gente quer que ele faça.

Os fluxogramas são (as vezes) fáceis de preparar, fáceis de usar e livres de ambiguidades.

Há quem sofra de "fluxogramite" como está mostrado na Figura 1.33.



Figura 1.33

Também se costuma chamar o fluxograma como diagrama de blocos ou diagrama de fluxo.

Há quem utilize também o termo fluxograma.

Assim um fluxograma é basicamente um diagrama composto de símbolos, linhas direcionais e informação de como o computador será usado para resolver um problema.

O fluxograma de um programa mostra o que deve ser realizado.

Comumente os símbolos usados em um fluxograma são padronizados e alguns deles estão indicados na Figura 1.34.

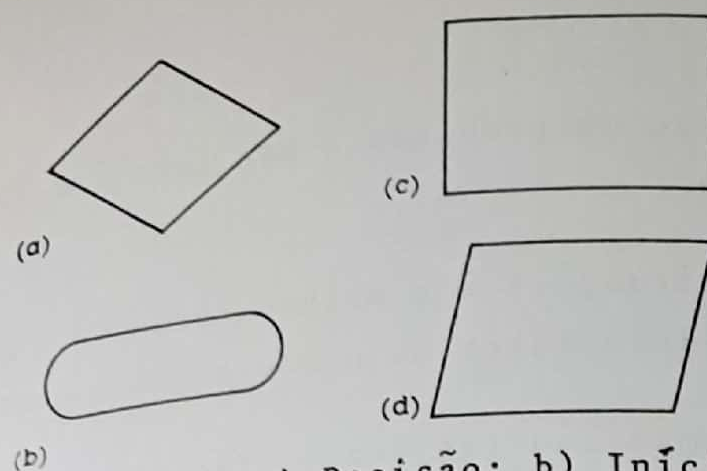


Figura 1.34 - a) Decisão; b) Início, parada ou fim; c) Processo de cálculo ou alguma função; d) Entrada ou saída.

Um problema genérico poderia ter por exemplo um fluxograma como o indicado na Figura 1.35.

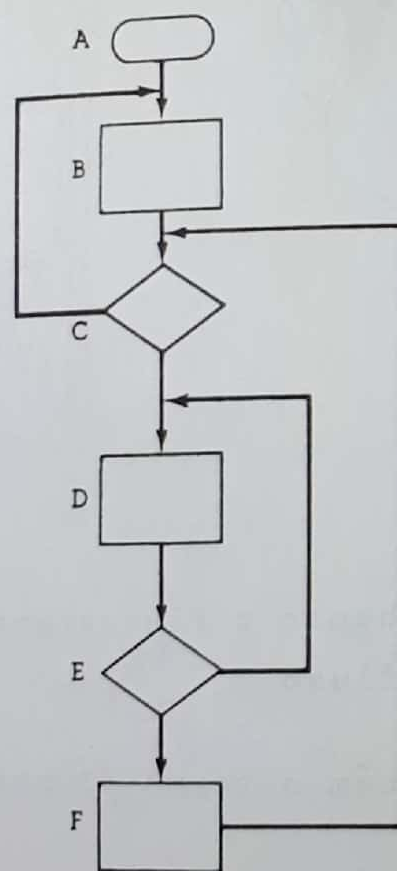


Figura 1.35

Bem para não tornar impossível o devaneio do fluxograma e para não deixar o caro leitor ou leitora muito bravos começando a chamar o diabo de meu bem e bomba atômica de tesouro aí vão dois exemplos de como se pode analisar ou acompanhar ou ainda descrever uma situação desde o início até o fim por um fluxograma. (Figura 1.36).

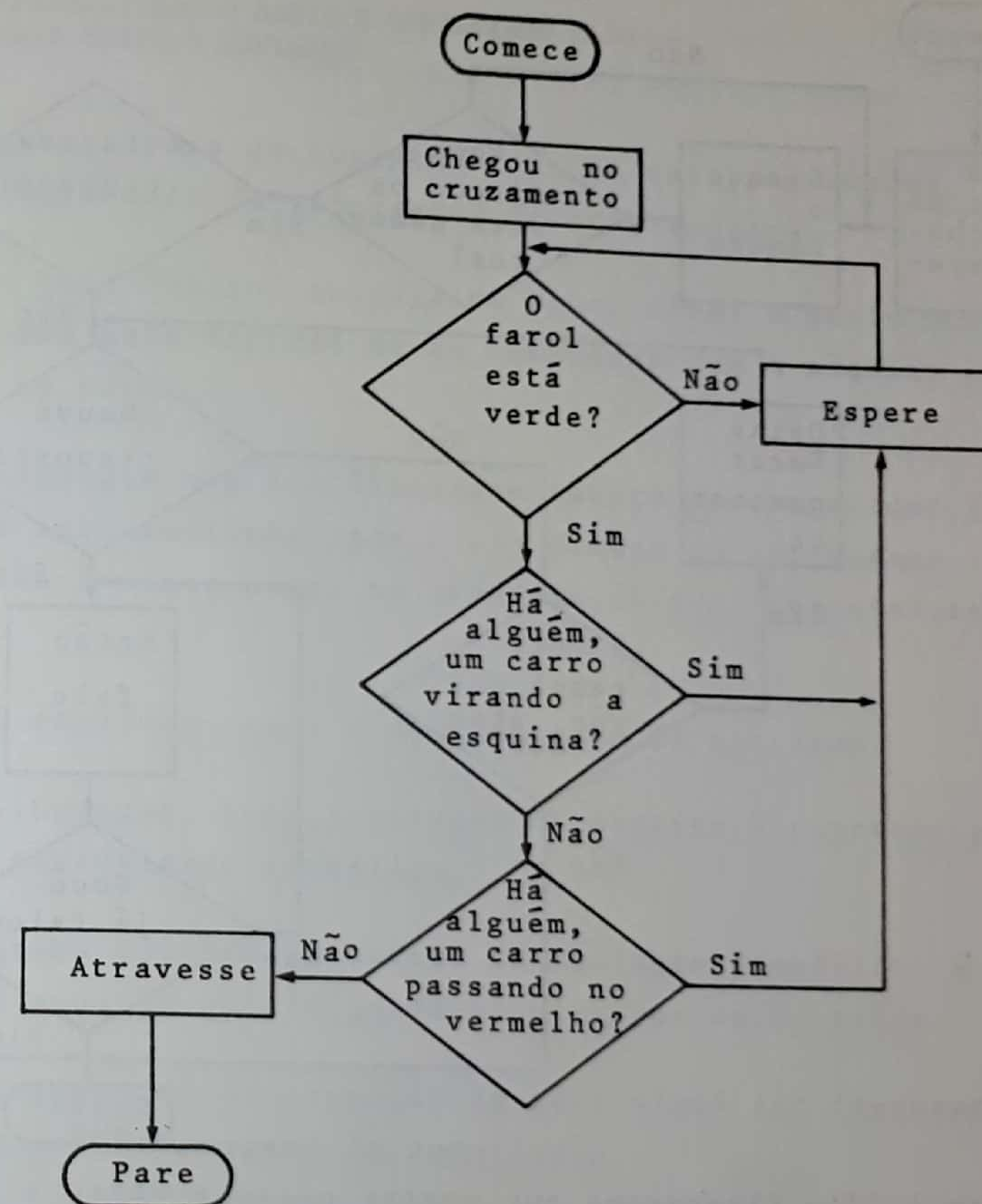


Figura 1.36 - Fluxograma que mostra como devemos proceder para atravessar uma rua num cruzamento.

Como este livro tem como finalidade a "apresentação" do BASIC procuraremos aprender a programar corretamente dispensando o uso de fluxogramas e desta forma não entrando na análise de problemas muito complexos, se bem que não queremos também cair no extremo oposto de resumir os nossos problemas a situação indicada na Figura 1.38.

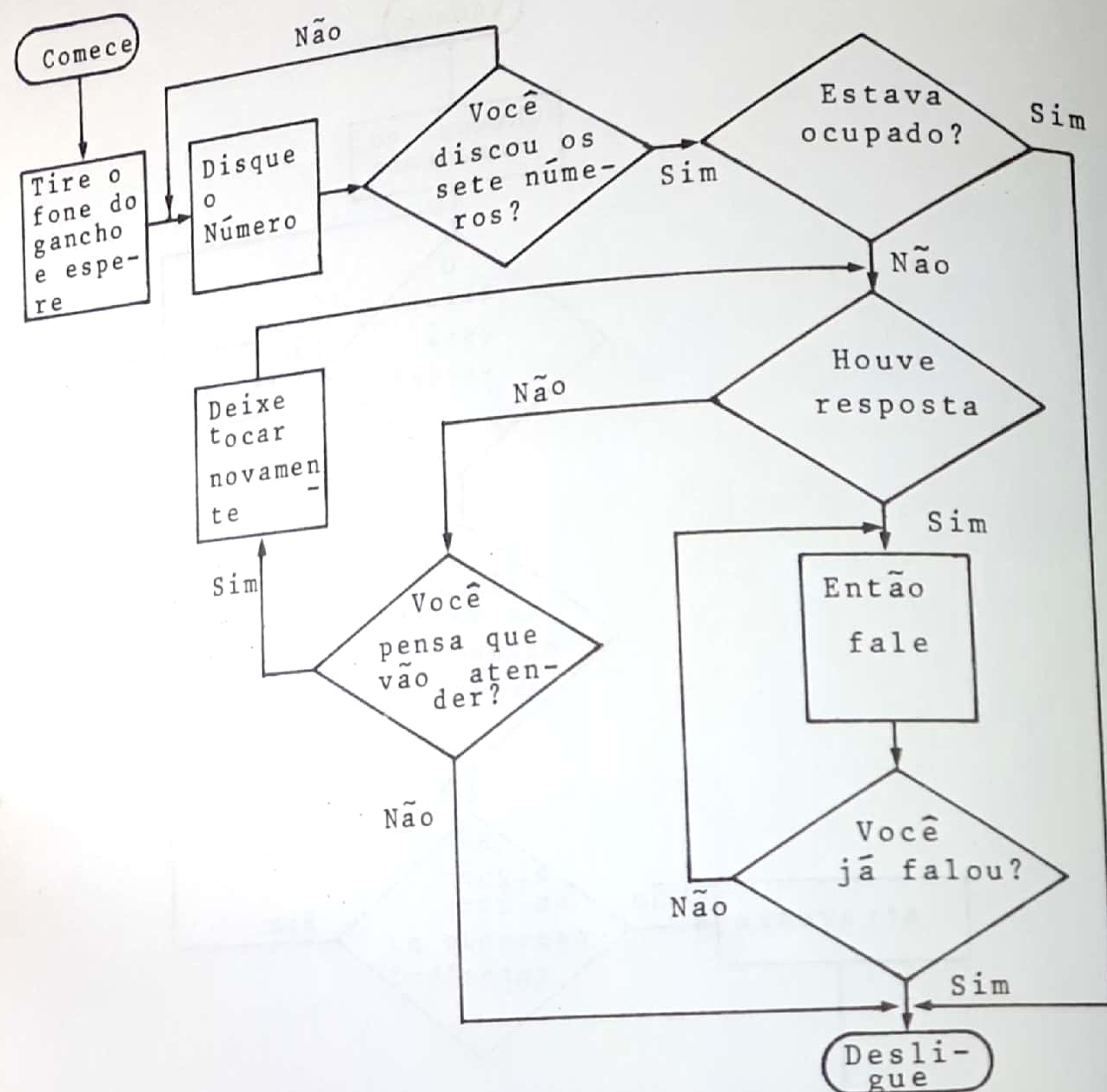


Figura 1.37 - Fluxograma que mostra passo a passo a forma de como poderíamos analisar uma chamada telefônica.



Figura 1.38

§1.6 - APRESENTANDO O BASIC E CONHECENDO ALGO MAIS SOBRE O SOFTWARE

Os programas para um computador podem ser programados usando uma variedade de linguagens de programação.

A forma mais "baixa" ou seja de menor nível e neste caso sendo a forma mais difícil de se comunicar com a máquina é o código de máquina.

Esta linguagem que é a linguagem básica do computador é difícil e inconveniente para o uso direto do ser humano e portanto não é nunca usada na programação dos computadores atuais.

O nível seguinte seria o da linguagem de montagem.

Nessa linguagem, cada instrução de máquina é representada por um equivalente simbólico.

Um programa escrito em código de montagem simbólico é mais tarde traduzido em código de máquina por um montador.

As linguagens de programação de alto nível são frequentemente chamadas linguagens de compilação.

Estas linguagens são menos dependentes da máquina que as linguagens de montagem.

Linguagens de alto nível são largamente utilizadas por pessoas que não tem um conhecimento detalhado de computadores ou da programação em linguagem de máquina.

Os programas escritos nessas linguagens são traduzidos para o código de máquina por um compilador (que é um tradutor) ou por um interpretador.

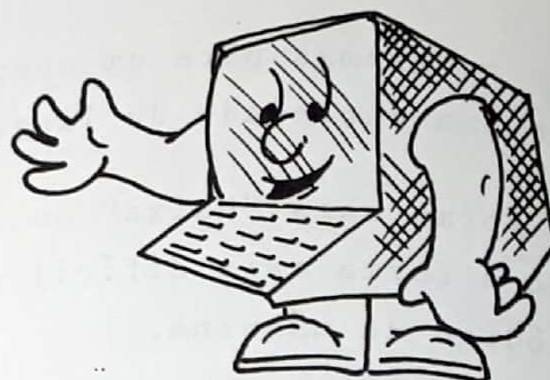
Entre as linguagens de alto nível mais populares citamos o BASIC, COBOL, PL/I, FORTRAN, RPG, APL, Pascal, PL/C, WATFIV,

ADA, Algol, etc.

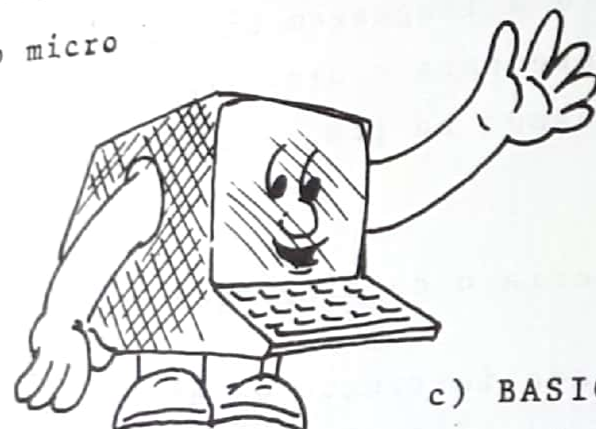
O nosso livro é sobre a linguagem BASIC.



a) Código do micro



b) Linguagem de máquina



c) BASIC

Figura 1.39 - "Com o BASIC animaremos o nosso micro".

Por quê o nosso livro é sobre BASIC e não sobre alguma língua oriental (Figura 1.40)?



BEM QUE O PAPAI DIZIA
QUE SERIA UM ERRO
TRAZER ALGUMA
COISINHA DO JAPÃO...

Figura 1.40

Em primeiro lugar e não tendo nada a ver com a piadinha da Figura 1.40 porque todas as linguagens de alto nível são fáceis para aprender e usar (você é que vai dizer isto daqui a algum tempo) e isto é particularmente verdadeiro para o BASIC.

Em segundo lugar porque hoje o BASIC está disponível em quase todos os sistemas de tempo compartilhado, na maior parte dos minicomputadores, nas máquinas de média e grande escala e em quase todos os microcomputadores.

BASIC é um acrônimo para

Beginner's All - purpose Symbolic Instruction Code

e foi primeiramente planejada para uma computação conversacional num processo interativo ou seja em um sistema de computação com tempo compartilhado para pessoas que não tivessem nenhuma experiência com computadores ou em escrever programas.

Esta pelo menos foi a idéia original de John Kemeny e Thomas Kurtz desenvolvida em 1964 no Dartmouth College.

Como já dissemos a linguagem foi inventada para o uso de todas as pessoas, de qualquer raça, de qualquer profissão, de qualquer sexo e qualquer idade.

Portanto em terceiro lugar e talvez o mais importante a linguagem BASIC não tem discriminações!!!

Lá vai uma propaganda para levantar a sua moral:

BASIC é fácil de aprender

BASIC é fácil de usar

BASIC é fácil de lembrar

BASIC tem umas poucas regras de sintaxe e um pequeno número de instruções

BASIC pode ser usada em quase todas as aplicações

BASIC é disponível (como já dissemos) em quase todos os tipos de computadores.

Ufa, ufa, você meu caro leitor já está convencido que se é BASIC é bom...

Um programa escrito em BASIC é comumente submetido diretamente ao computador de um terminal, usualmente um teclado com um acoplamento com aparelho de T.V.

Nos sistemas BASIC a *compilação* (tradução) ou seja a interpretação das instruções ocorre a medida que as mesmas vão sendo tecladas.

Para usar o sistema BASIC, o usuário simplesmente tecla o programa (ou entra via disco ou fita) no terminal e manda um comando de execução tal como RUN.

As respostas do programa são produzidas imediatamente.

Pode-se perceber com facilidade que a possibilidade de solução de problemas interativamente pelo BASIC é de um valor extraordinário para os usuários.

A linguagem BASIC é constituída de umas quarenta instruções do tipo GOTO, READ, FOR-NEXT, PRINT, CHR\$, END, etc.

É a mais fácil de todas as linguagens de alto nível.

Pode ser aprendida em horas e ser dominada em alguns dias.

Cada instrução em um programa BASIC tem um número de linha que serve para identificar a linha e especificar a ordem na qual as instruções são executadas pelo computador.

De 1964 para frente a linguagem BASIC tem estado em contínuo desenvolvimento e muitas implementações na linguagem são hoje disponíveis, feitas por fabricantes ou elaboradas em Escolas.

É com este intuito por exemplo que a Itautec acaba de assinar um convênio com a Escola Politécnica da Universidade de São Paulo (junho/1983).

Novos aspectos tem sido acrescentados a linguagem BASIC original e é por isso que frequentemente se houve dizer de "super BASIC", "BASIC simplificado" e mesmo de "BASIC básico".

Nos dividimos o sistema BASIC do livro em três partes, a 1ª para que se possa atingir o iniciante.



Figura 1.41 - "Se é BASIC é bom"

Mais tarde o iniciante será "veterano" e como ele deve conhecer também as minúcias, reservam-se para ele nessa época a 2ª e 3ª partes ou seja os capítulos 3, 4 e 5 respectivamente.

A linguagem BASIC é usada hoje no Brasil nas Faculdades que tenham no seu curso a disciplina Ciência da Computação ou algo similar e esperamos (sonhando de olhos abertos) que isto ocorra o mais breve possível nas escolas do 1º e 2º grau por decisão do Ministério de Educação e Cultura.

O sistema BASIC é sem dúvida, até agora, a melhor forma para alguém que tenha a possibilidade de usar microcomputadores ou que queira (que é o seu caso) aprender como os micros resolvem problemas, mas que não quer ser um PROFISSIONAL DA PROGRAMAÇÃO.

Será que todo esse papo o encorajou?
 Esperamos que sim e que você não se torne segundo a definição de Renato Sabbatini um *computaca* ou seja panaca através do computador.
 Boa sorte é pouco nesse caso.

Você irá se esforçar e assim o eventual temor que possa vir do "computo-papão" irá rapidamente desaparecer.



Figura 1.42

§1.7 - SISTEMAS DE PROCESSAMENTO DE INFORMAÇÕES

O declínio do custo dos micro e dos minis, da memória e dos terminais inteligentes está permitindo não apenas a indústria e ao comércio, mas sim a todos ter o seu micro ou então instalar redes de processamento.

Tempo compartilhado (time-sharing) é o uso simultâneo de um sistema de computadores de múltiplos terminais.

Um sistema em time-sharing faz com que as facilidades de um

computador sejam disponíveis a todos os usuários, praticamente ao mesmo tempo (Figura 1.43).

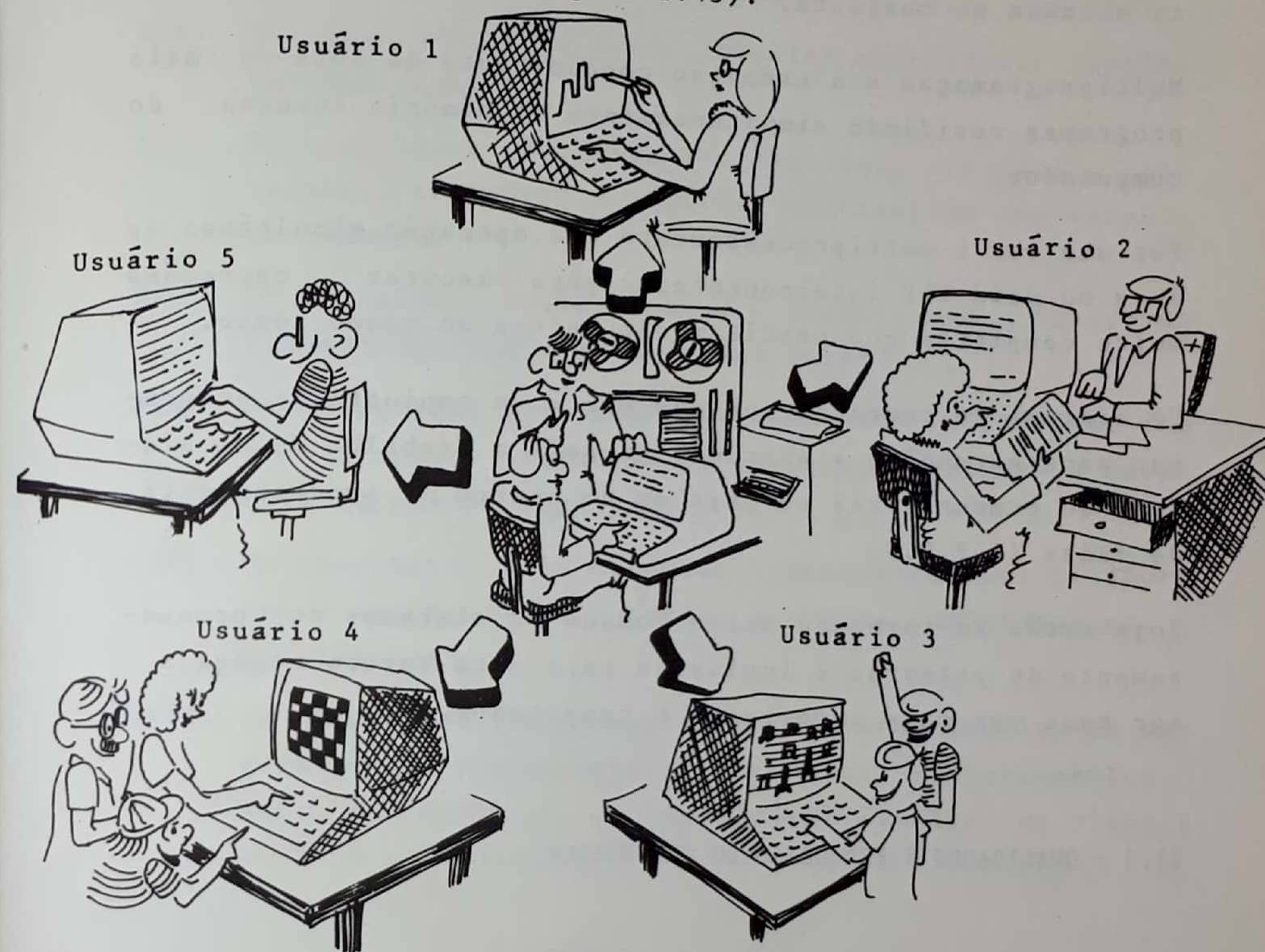


Figura 1.43 - Sistema "time-sharing"

Um sistema em tempo real controla um processo em andamento e envia as suas saídas não mais tarde que no instante em que elas são necessárias para um controle efetivo.

Sistemas em tempo real aparecem em vários setores e um exemplo típico é a reserva de lugares feita pelas companhias de aviação ou o atual serviço de bancos nas transações com outros bancos ou entre as suas filiais.

Processamento em lotes ou bateladas (batch) é um método de processamento de dados no qual as transações são acumuladas por um período predeterminado de tempo e então prepara-se

uma entrada no computador para ser processada como uma única unidade ou conjunto.

Multiprogramação é a execução concomitante de dois ou mais programas residindo simultaneamente na memória interna do computador.

Por sua vez o multiprocessamento é a operação simultânea de duas ou mais UCP interconectadas para executar operações muito complexas que precisam ser feitas ao mesmo tempo.

Um sistema de operação por sua vez é um conjunto de programas para gerenciar a execução ou seja o trabalho de um sistema de computadores ou seja de um centro de processamento de dados (C.P.D.).

Hoje estão se tornando muito comuns os sistemas de processamento de palavras e inclusive para esta tarefa começa a ser cada vez mais importante o computador.

§1.8 - QUALIDADES E FRAQUEZAS DO COMPUTADOR

Pode o computador ser bastante útil?

Nós já vimos que o computador tem vários usos, mas como qual quer outra coisa ele tem suas qualidades e suas fraquezas ou "defeitos".

Qualidades

- I) Um computador lida com quantidades enormes de informação de maneira muito rápida.
Ele pode repetir o mesmo cálculo inúmeras vezes ou colocar os mesmos dados em vários arquivos simultaneamente.
- II) Um computador é incansável, preciso e consistente. Ele

não troca dígitos ou os coloca em lugar errado.
Para ele é proibido ter devaneios ou estar fatigado.

- III) Um computador pode fazer cálculos matemáticos simples ou complexos.
- IV) Um computador pode separar, combinar, recolocar, armazenar, restaurar e arquivar informações com espantosa rapidez.
- V) Um computador lembra falhas e precisa ser instruído somente uma vez acerca do que fazer.
- VI) Um computador pode imprimir esmerados relatórios, cheques, contas, etiquetas ou qualquer papel que a natureza do seu serviço pedir.
- VII) Um computador pode executar inúmeras tarefas de secretaria concernentes a cartas e textos (processamento de palavras).
- VIII) Um computador pode simular ou elaborar modelos de organização, estratégias de trabalho, processamentos, etc. Ele pode ser um inestimável auxiliar em planejamento e previsões.

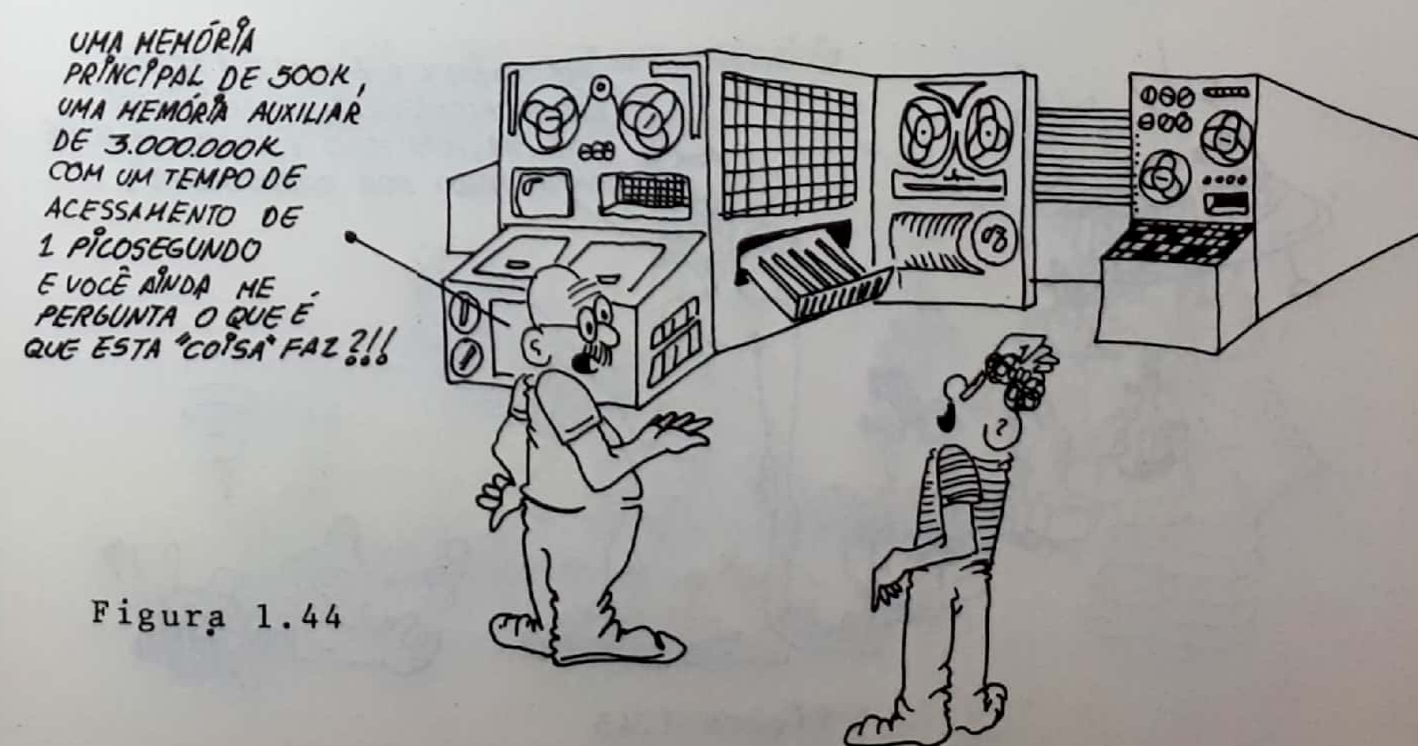


Figura 1.44

Fraquezas

I) Um computador não pode pensar. Embora ele possa ser programado (instruído) a fazer algumas coisas espantosas.

Ele não "pensa" de nenhum modo
Pensamento é ainda prioridade do ser humano.

II) Um computador não é tolerante ou adaptável à pessoas. Ele não reconhece erros nos dados de entrada a não ser que ele tenha sido programado especificamente para fazê-lo para cada tipo de erro.

Ele somente pode reconhecer (e somente se o programa pedir para reconhecer) coisas como números cujo caractere foi pedido ou um ponto decimal esquecido (sete dígitos entraram onde somente seis são permitidos).

O mais simples empregado é muito mais flexível em lidar com erros humanos - reconhecendo dados incompletos, omissões, pequenos erros contábeis, etc.

O empregado aprende por experiência, o computador NÃO!!!

Ele continua fazendo o mesmo erro SEMPRE, SEMPRE, SEMPRE...



Figura 1.45

III) O computador não trabalha ou melhor não funciona quando tem qualquer defeito.

Ele é usualmente ou oito ou oitenta ou reforçando mais ou TUDO ou NADA.

Todo robô tem dentro de si um micro e aproveitamos aqui para repetir as três leis da Robótica do genial Isaac Asimov:

1ª Lei - Um robô jamais irá injuriar um ser humano ou através da sua condição estática permitir que o ser humano tenha dano ou fique em perigo.

2ª Lei - Um robô precisa obedecer as ordens dadas a ele por seres humanos exceto quando tais ordens entram em conflito com a 1ª Lei.

3ª Lei - Um robô precisa proteger a sua própria existência na medida que esta proteção não entre em conflito com a 1ª e a 2ª Leis.

Deus queira que tudo o que seja inventado daqui para frente se encaixe perfeitamente nestas leis e não ocorra nunca algo semelhante ao mostrado na Figura 1.46.

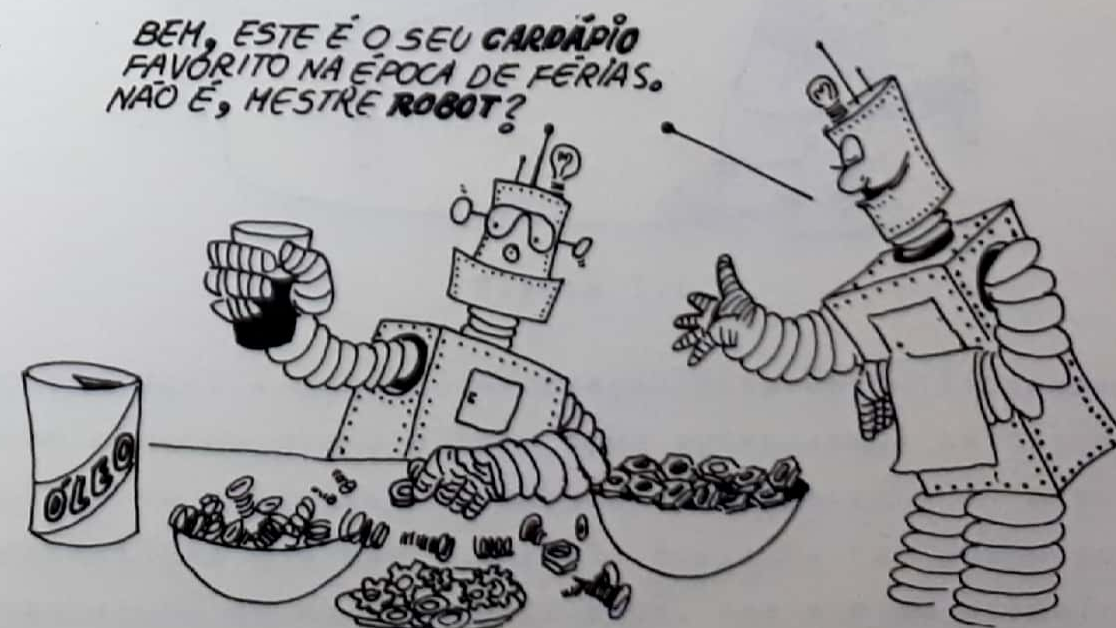


Figura 1.46

A linguagem basic elementar

§2.1 - A ETIQUETA DO COMPUTADOR

No instante em que você ligar o seu micro (de preferência na voltagem certa) ele obedecerá a um conjunto bem definido de regras para poder brincar com o "mestre" que é você. Surpreso!!!

Se você impetuoso leitor(a) seguir estas regras o uso do micro será bem simples.

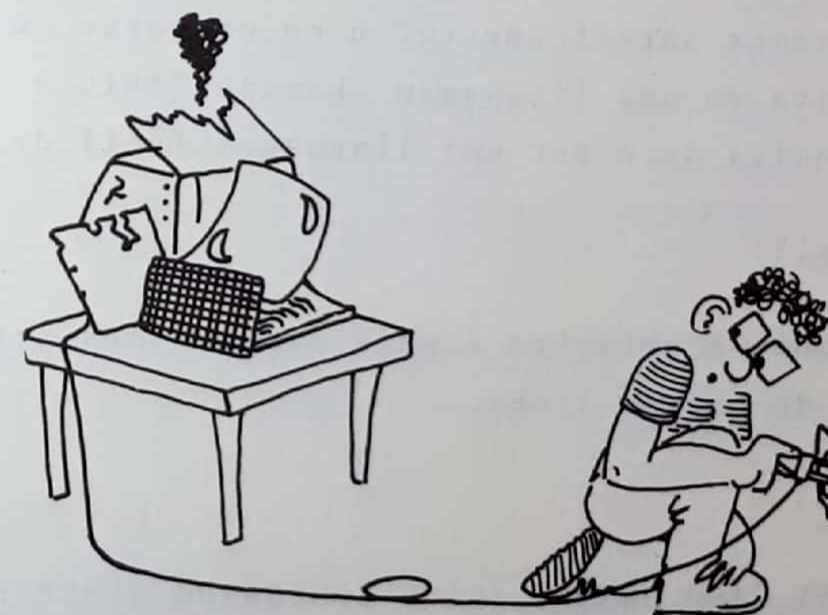


Figura 2.1

Começamos aqui a nossa conversação através da linguagem BASIC com o nosso micro e logo após aprendermos as noções fundamentais da etiqueta do computador chegaremos a maravilha das maravilhas que não é a Sonia Braga ou tão pouco uma lata inteirinha de sorvete para você, mas sim ao primeiro programa escrito e executado no seu micro.

Comumente para que o micro comece a operar é necessário apertar a tecla ON.

A seguir deve-se apertar a tecla ENTER ou NEW LINE e depois de alguma divulgação comercial do fabricante da máquina que você está usando aparecerá na tela do seu aparelho de T.V. um cursor brilhante que pode ser do tipo

READY



Vamos nos preparar agora para escrever um programa.

Mas o que é realmente um programa?

→ Um programa é uma sequência de instruções que o micro armazena até que você lhe envie o sinal para executar o mesmo.

Como já dissemos anteriormente, a maior parte dos programas é hoje escrita em uma linguagem chamada BASIC e como o próprio nome indica deve ser uma linguagem fácil de aprender.

Que bom hein!?!

Escrevamos pois o primeiro e mais simples dos programas ou seja aquele de uma só linha.

Exemplo 2.1

```
10 PRINT "ALÔ MAMÃE, ESTOU COMEÇANDO A MEXER NO MICRO"
```

Examinemos com atenção o que foi escrito.

- 1) Tudo o que está escrito depois do PRINT está entre aspas?
- 2) Não tem aspas demais?

Se as duas perguntas foram verificadas aperte agora a tecla ENTER, NEW LINE ou CR (tecla RETURN no MAX XI).

O cursor reaparecerá se tudo está em ordem e o micro está no fundo lhe dizendo

"Bom, qual é a próxima instrução?"

Se algo está errado existe a necessidade de corrigir a linha e no começo você terá muitas dificuldades nesta parte.

Podemos agora, se o nosso programa é de apenas uma linha, dizer ao micro para que execute o mesmo.

O comando BASIC para esta finalidade é simplesmente a tecla RUN.

Desta forma "escreve-se" o RUN e aperta-se a tecla ENTER novamente.

Caso você não tenha feito nenhum erro lerá:

ALÔ MAMÃE, ESTOU COMEÇANDO A MEXER NO MICRO

Repetindo:

- 1) *Comandos* são executados assim que teclarmos os mesmos e apertarmos o ENTER.
- 2) *Instruções* são chamadas dentro dos programas e são executadas somente depois que se entra com o comando RUN.

Mensagem especial

Espere, não comece a pular as páginas deste livro para logo começar a brincar com o seu micro.

Não esqueça que "quem tem pressa, come crú...".

§2.2 - EXPANDINDO UM POUCO O NOSSO ELEMENTAR PROGRAMA

Já temos um programa de uma linha.

Você já percebeu que nos usamos 0 para o número zero para

distingui-lo da letra O.
Na tela do seu aparelho de T.V. em muitos casos a letra O e o zero Ø não são a mesma coisa (Figura 2.2 a) e b).)

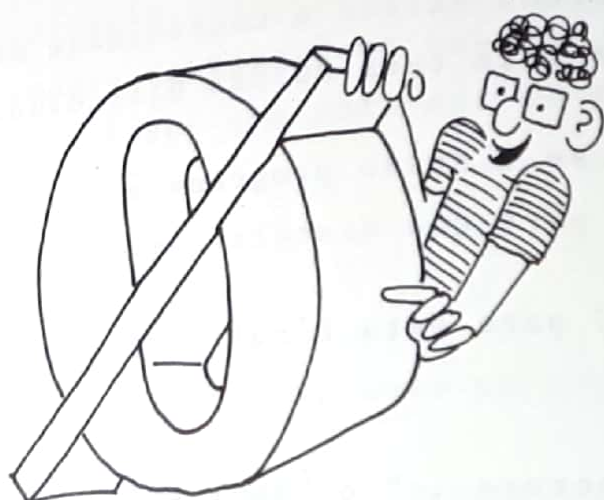


Figura 2.2a - Número zero (Ø)



Figura 2.2b - Letra O

Se expandirmos o programa do Exemplo 2.1 com a 2ª linha

```
2Ø PRMT "OH MAGNÍFICO MICRO SEREI O SEU SENHOR"
RUN
```

obteremos uma mensagem de erro que pode aparecer da seguinte forma.

?SN ERRO na 2Ø (ou outra forma equivalente)

É tradicional representar o espaçamento entre as linhas de um programa por números que crescem de 1Ø em 1Ø.

Não é proibido entretanto numerar as linhas com 1,2,3,4,etc.

A vantagem de ir de 1Ø em 1Ø é que se esquecermos alguma instrução é fácil inserir uma nova digamos 65 entre as instruções de números de linha 6Ø e 7Ø (Figura 2.3).

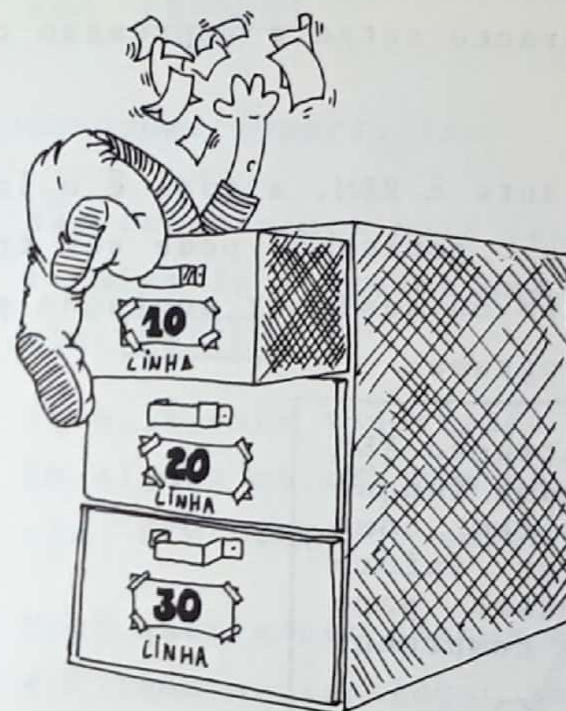


Figura 2.3

Cada instrução deve ser precedida pelo seu número de linha

Digamos que se queira para o nosso programa (Exemplo 2.1), agora já com duas linhas, pular uma linha entre as mesmas.

Para que isto ocorra o programa ficará:

Exemplo 2.2

```
1Ø PRINT "ALÔ MAMÃE, ESTOU COMEÇANDO A MEXER NO MICRO"
15 PRINT
2Ø PRINT "OH MAGNÍFICO MICRO SEREI O SEU SENHOR"
```

Teremos na tela ou na impressora se colocarmos LPRINT no lugar do PRINT

```
ALÔ MAMÃE, ESTOU COMEÇANDO A MEXER NO MICRO
OH MAGNÍFICO MICRO SEREI O SEU SENHOR
```

Insistindo mais um pouco sobre a linha 15.

Nela está dito PRINT
Mas PRINT o que????
Bem na realidade NADA.

Isto é o que segue o PRINT e é justamente o que se imprime ou seja NADA.

Porém no processo de se inscrever ou imprimir "nada" automaticamente se abre um espaço em branco entre a impressão ordenada nas linhas 10 e 20.

Uma outra instrução muito importante é REM, a qual é o início de REMark que do inglês para o português pode ser traduzida como observação, comentário ou qualquer anotação para facilitar a compreensão do programa.

Figura 2.4 - A instrução REM não executa nenhuma função no programa, ela tem como finalidade colocar informações sobre o mesmo para facilitar a sua compreensão.



Por quê é importante a instrução REM?

Se você ou alguém mais olhar pela primeira vez um programa complicado, pode-se explicar os detalhes do mesmo, com várias instruções REM colocadas em pontos estratégicos.

Assim o nosso programinha do Exemplo 2.1 poderá ficar:

Exemplo 2.3

```
5 REM *** ESTE É O MEU PRIMEIRO PROGRAMA NO COMPUTADOR"
   ENTER
10 LPRINT "ALÔ MAMÃE ESTOU COMEÇANDO A MEXER NO MICRO"
   ENTER
15 LPRINT   ENTER
20 LPRINT "OH MAGNÍFICO MICRO, SEREI O SEU SENHOR" ENTER
RUN   ENTER
```

Observações importantes

- 1) Não repetiremos mais daqui para frente que sempre que se termina de escrever uma instrução deve-se apertar a tecla **ENTER**.

Isto já está memorizado não é?

Em alguns micros tem-se no lugar da tecla ENTER a tecla NEW LINE, ou então a tecla RETURN como no MAXXI.

- 2) Você quer saber porque colocamos os asteriscos na linha # 5 (que sofisticação hein, no lugar de "número" estamos colocando # que as vezes é confundido com "libra")?

A resposta é... eles apareceram apenas para decoração ou seja para dar mais "personalidade" ao que se escreveu.

Não esqueça porém que qualquer coisa que for batida ou seja teclada na linha que segue o REM é ignorado pelo computador.

Esta é uma instrução não executável e dessa forma narrativas ou mensagens que o usuário deseja incluir na listagem do programa devem aparecer logo após a palavra chave REM.

Como se vê esse negócio de programação está começando a ficar complicado e se você esquecer algo no seu já "enorme" programa como se pode obter uma listagem do que o mesmo contém?

Ora isto é muito fácil e temos para isto um novo comando BASIC.

Não esqueça que comandos BASIC são instruções que o micro executa no chamado "modo direto", isto é, tão logo que se dê a entrada, através da tecla ENTER, NEW LINE ou RETURN.

Os comandos podem ser executados a qualquer momento.

E assim que basta escrever LIST ou LLIST e com isto o micro fornece uma listagem ordenada de um programa da área de trabalho (na memória) para a tela do vídeo ou impressora (alternativa LLIST).

Onde é o fim de um programa?

O fim de um programa como naturalmente deveria acontecer é a última instrução que nos queremos que o microcomputador execute.

Muitos micros necessitam ainda que se coloque neste ponto a instrução END de forma que o micro saiba que o programa terminou.

Quando o uso do END é opcional, por compatibilidade e para formar um costume, deve-se usar sempre o mesmo, exceto se a implementação não tiver a instrução END.

A instrução END também indica o fim lógico do programa, isto é, o fim da execução.

Nos sistemas BASIC onde não é obrigatória, a execução terminará quando for encontrada a instrução com o maior número de linha.

Observação importante

Outros comandos são particularmente úteis como: NEW, CLEAR, CONT, EDIT, BREAK.

O NEW é um comando que inicializará a memória, isto é, apagará todos os programas e dados deixando a memória limpa para receber novos programas e dados.



O NEW diz ao micro que todos os programas anteriormente armazenados no micro devem ser apagados para a introdução de um novo programa.

Figura 2.5

O comando CLEAR limpa apenas os dados, isto é libera as áreas da memória que contêm valores permitindo sua nova utilização.

O comando CONT permite a execução da continuação de um programa que tenha sido interrompido.

O comando EDIT seleciona uma linha para a área de edição.

É muito importante para se poder fazer correções em uma dada linha.

Com o comando BREAK ou seja a tecla BREAK é possível interromper a execução de qualquer programa, por meio de um simples toque.

O comando BREAK só é acionado quando um programa está em execução e não aparece na tela do vídeo.

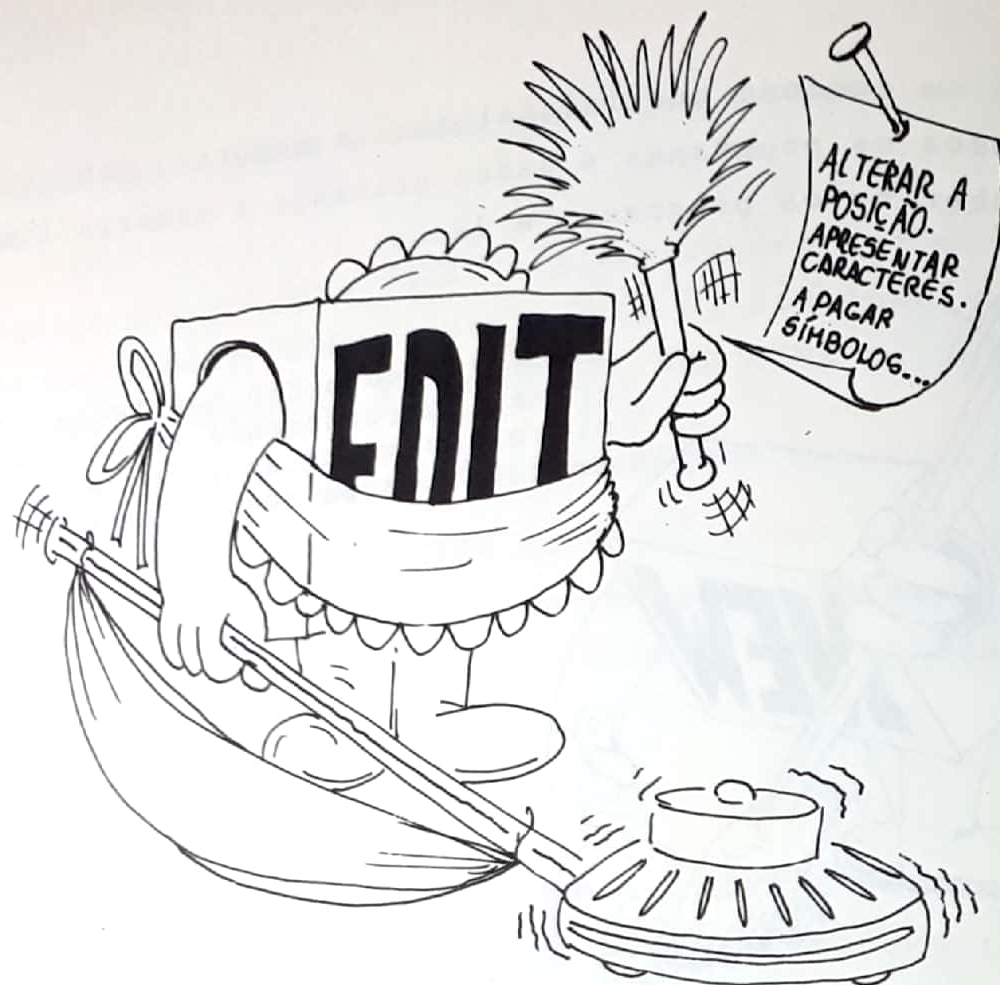


Figura 2.6 - Não esqueça do comando EDIT que vai ser tão útil para "arrumar" os seus programas.

Para terminar, um papinho sobre o teclado.

Os caracteres de um micro isto é do seu teclado englobam não só símbolos (letras, dígitos, etc.) como também os sinais compostos (palavras-chave, nomes de funções, etc.).

Em alguns micros uma tecla chega a ter 6 significados distintos e esta diferenciação no seu uso é obtida pela tecla SHIFT e pela modalidade na qual se utiliza a máquina.

Por exemplo o TK-85 tem a modalidade K (para palavras-chave), a modalidade L (para letras), a modalidade F (para funções) e a modalidade G (para gráficos).

\$2.3 -

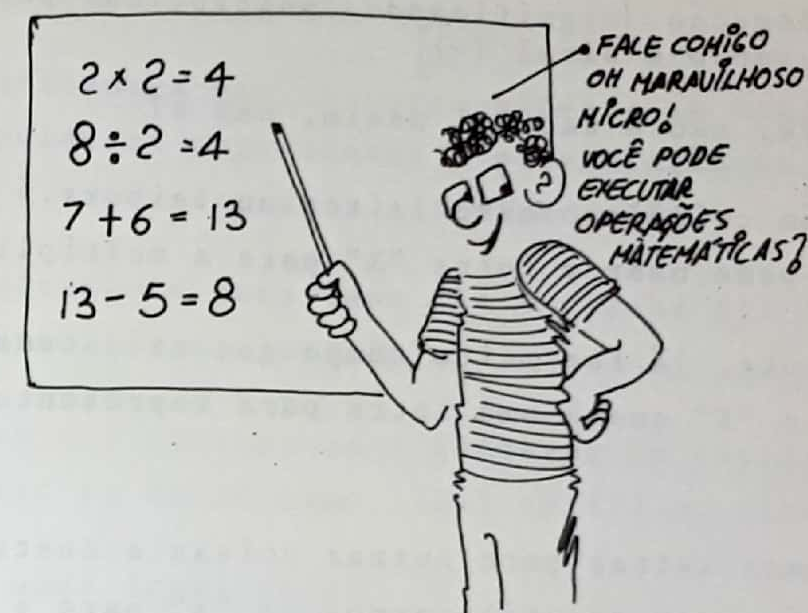


Figura 2.7

Para começar vamos informando que o BASIC usa as quatro operações aritméticas fundamentais, mais uma quinta que é justamente uma modificação de duas das outras.

1. Adição, usando o sinal (+)
2. Subtração, usando o sinal (-)

Veja que até agora não temos nada de especial em relação ao que acontece no 1º grau.

Aliás até agora a estimada e já velha professora que nos alfabetizou diria que o micro não é de nada!

Ah! Ela não faria isto, pois entre outras coisas era prudente...

3. Multiplicação, usando o sinal (*) que é um asterisco

Oh droga, isto é muito fácil para que se possa acreditar que é verdade!

4. Divisão, usando o sinal (/) que é uma barra

Bem, ao menos é um sinal mais simples que o velho sinal (÷)

5, Negação [significando "multiplicar por menos um"
usando o sinal (-)]

Como se vê, não é tão mal assim, não é?

Seja porém cuidadoso caro leitor ou leitora.

Você não pode usar a letra "X" para a multiplicação.

Infelizmente, já faz muito tempo que os matemáticos decidiram usar o "X" que é uma letra para representar a multiplicação.

Nos usaremos letras para outras coisas e desta forma a confusão será menor se utilizarmos um "*" para a multiplicação.

CONFUSÃO É UMA COISA QUE O MICRO NÃO TOLERA

Dessa forma para os micros, "*" é o único sinal que significa multiplicação e aliás depois de utilizá-lo você se sentirá com uma terrível vontade de abolir para sempre o uso do "X" como um sinal de multiplicação.

Naturalmente o arguto e perspicaz leitor ou leitora já deve estar "encucado" com o que aconteceu com o seu velho e favorito sinal de igual (=).

Espere um pouco mais (pois caso contrário come crú...) e aprenderás que a linguagem BASIC tem um uso todo especial para este sinal.

Expressões matemáticas como $13 + 2*7 - 14/7$ podem estar apenas no membro da direita do sinal de igual.

O membro da esquerda do sinal de igual está reservado para o "nome da variável".

Este é o nome que nos daremos ao resultado da expressão matemática.

E a potenciação como é representada?

Bem, aí os fabricantes não chegaram ainda a um acordo e a forma mais popular de representar a potenciação ou seja a exponenciação é a de usar dois asteriscos (**).

Há quem use entretanto para esse fim a flecha (↑) ou ainda o sinal de acento circunflexo (^).

Dessa forma nos que aprendemos a escrever na escola 7^3 teremos que escrever em BASIC como $7**3$ ou $7↑3$ ou ainda 7^3 .

Aqui não está nada legal hein!

E vem mais "complicação" adiante!?!

Podemos agora colocar a "besta" num trabalhinho matemático visto que já conhecemos quase todos os símbolos ou seja sinais matemáticos (oh pânico deixe-me).

Exemplo 2.4

Qual é a distância de São Paulo até a cidade de Patópolis se um avião voando a velocidade de 900 quilômetros por hora fez a viagem em 11 horas?

Solução:

Representemos a velocidade por V e o tempo gasto por T e ensinemos ao micro que para obter a distância D entre as duas cidades basta multiplicar V por T.

Neste caso deveremos antes saber o que é uma instrução de atribuição.

Instrução de atribuição é aquela que pode atribuir (e poderia ser de outra forma hein!?) valores a variáveis numéricas ou alfanuméricas (estas como veremos adiante não são variáveis extra-terrenas ou seja E.T.).

Em BASIC esta instrução é a LET

Alguns micros são tão desenvolvidos que não requerem o uso do LET quando se programam equações ou expressões matemáticas.

Nós para ficarmos seguros colocaremos quase sempre o LET antes de uma variável que será seguida de um sinal de igual com o sentido de atribuição do valor da expressão matemática ou alfanumérica que vem no membro da direita.

Nesta altura dos acontecimentos podemos escrever o nosso programa.

```
10 REM ** D=DISTÂNCIA, V=VELOCIDADE, T=TEMPO **
20 LET V=900
30 LET T=11
40 LET D=V*T
```

Já olhou bem o programa? Sim, bem mande então o RUN.

Hum, hum, hum, ... dum, dum, dum, - ru, ri, ro, ...

Seguramente este é um micro do tempo de Babbage pois está demorando muito para dizer que são 9900 quilômetros.

Tudo indica que está tudo OK e o micro não trabalha!!!!?

Não, ele trabalha e até muito bem e faz a conta 900 vezes 11 com uma rapidez extraordinária porém nós (você e eu) esquecemos de dizer-lhe para que nos dê a resposta.

Desculpe, mas te peguei!

Então basta agora acrescentar ao programa a instrução:

```
50 LPRINT "A DISTÂNCIA (EM QUILOMETROS), E: "; D
RUN
```

Obteremos na impressora (mais uma vez, se quiser saída na tela do vídeo use o PRINT)

A DISTÂNCIA (EM QUILOMETROS), E: 9900

Vamos explicar melhor como isto aconteceu.

A mensagem que apareceu entre aspas foi impressa exatamente como foi especificado e a letra D pediu o valor da distância percorrida.

A vírgula diz ao micro que nos queremos imprimir itens separados na mesma linha ou seja em zonas distintas.

Dependendo do modelo de micro podemos ter mais ou menos zonas de impressão por linha.

É comum termos aqueles que tem 5 zonas de 14, 15 ou 16 caracteres cada uma.

Desta forma o uso da vírgula (,) causa a impressão com vários espaços entre os valores das expressões.

Já o uso do ponto e vírgula (;) e cuidado com essa diferença, causa geralmente a impressão dos valores sem espaço entre eles.

Serão dados mais detalhes sobre isto quando tratarmos do formato de saída.

Bem agora, pode-se perguntar como seria a instrução de número de linha 50 se quiséssemos que saísse impresso

- a) A DISTÂNCIA E 9900 QUILOMETROS
- b) A DISTÂNCIA E DE 9900 QUILOMETROS

Resposta

- a) 50 PRINT "A DISTÂNCIA E "; D, "QUILOMETROS"
- b) 50 PRINT "A DISTÂNCIA E DE"; D; "QUILOMETROS"

Como você vê e irá se acostumar com isto, é possível eliminar todos os espaços extras em uma saída editando a separação com pontos e vírgulas no lugar de vírgulas.

Importante - É bom não esquecer que a nossa tradicional vírgula decimal é substituída no BASIC pelo ponto decimal.

TRABALHO PARA VOCÊ

- 1) Escreva um programa que ache o tempo que você leva no seu lindo Miura-Targa que pode andar em média 140 km/h para ir de Megalópolis a Ociosópolis que distam entre si 700 km e são interligados por uma linda estrada pavimentada.
- 2) O comprimento de um círculo é obtido multiplicando-se o seu diâmetro por $\pi = 3,14151927$ (pi). Escreva um programa que dê este comprimento para um círculo que tenha 13 m de diâmetro (este é o diâmetro da linda piscina do seu clube).

LEMBRANÇA INESQUECÍVEL - NÃO É A DO SEU ANIVERSÁRIO OU A DOS SEUS PAIS

Você pode usar qualquer uma das 26 letras de A até Z para indicar as variáveis e não apenas as letras D, V e T como foi feito no Exemplo 2.4.

- 3) Se a área de um círculo é obtida multiplicando-se π vezes o quadrado de raio, ache a área da piscina do seu clube (veja o problema 2).

§2.4 - NOTAÇÃO CIENTÍFICA E REGRAS DE PRECEDÊNCIA ENTRE OS OPERADORES ARITMÉTICOS

O que existe em maior quantidade, estrelas no céu, grãos de areia na praia ou formigas no seu sítio?

Neste nosso mundo matemático fomos agraciados com uma quantidade razoável de números grandes e pequenos.

Milhões de uns e talvez bilhões de outros para que se possa representar os mesmos.

o nosso magnífico micro dispõe do que se chama *notação exponencial* ou a *notação científica padrão* e aplicamos a mesma quando o tamanho do número começa a fugir da nossa (sua) "imaginação".

É ISTO QUE ME DEIXA "TÍMIDO".
OS SEUS PREÇOS SÃO TODOS EM
NOTAÇÃO CIENTÍFICA!



Figura 2.8

Assim o número 13 bilhões (13 000 000 000) pode ser escrito como 13E09.

Já se o número for cinco milionésimos $0,000005$ então precisamos deslocar o ponto decimal seis casas para a direita.

Assim em notação exponencial teríamos $5E-06$.

O micro coloca sempre os números grandes e pequenos na forma científica e por isto é bom se acostumar a mesma.

Exemplo 2.5

Que tal, para desenvolver a cabeça fazer o seguinte programa: se um carro percorre por ano aproximadamente $150\ 000$ km, quantos percorrerão 2000000 de carros?

Se a cada $50\ 000$ km deve-se trocar os 4 pneus de um carro quantos pneus serão vendidos no ano?

Hah, vamos ajudá-lo e aí vai o programa:

Solução:

```
10 REM ** PROGRAMA DE QUILOMETROS RODADOS E PNEUS VENDIDOS **
20 LET N=2E06
30 LET Q=150 000
35 REM ** D E A DISTÂNCIA TOTAL PERCORRIDA PELOS N CARROS **
40 LET D=N*Q
45 REM ** T E O NÚMERO DE TROCAS DE PNEUS POR CARRO **
50 LET T=Q/50000
55 REM ** M E O NÚMERO TOTAL DE PNEUS QUE SERÃO COMPRADOS **
60 LET M=4*T*N
70 LPRINT "O NÚMERO TOTAL DE QUILOMETROS E D=";D
80 LPRINT "O NÚMERO TOTAL DE PNEUS E M=";M
90 LPRINT "*** QUE BOM SER FABRICANTE DE PNEUS!!! ***"
```

Introduzindo o comando RUN teremos a saída

```
O NÚMERO TOTAL DE QUILOMETROS E D= 3E+11
O NÚMERO TOTAL DE PNEUS E M= 2.4E+07
*** QUE BOM SER FABRICANTE DE PNEUS!!! ***
```



Figura 2.9 - Usando a notação científica ou seja a exponencial dificilmente você vai ter problemas para representar uma medida.

Os parênteses e a ordem das operações

O uso dos parênteses desempenha uma função muito importante na programação de computadores, assim como ocorre na Matemática.

Eles são usados em computação da mesma forma que na Matemática porém existem *importantes exceções*.

- I) No BASIC, parênteses podem incluir operações que devem ser executadas.

As operações que estão dentro dos parênteses são executadas antes daquelas que não estão dentro dos parênteses.

- II) Podemos ter em uma expressão vários níveis de parênteses e as operações são sempre executadas do nível de parênteses mais interno para o mais externo.
- III) Quando existir um "empate" na ordem de execução das operações depois de terem sido "removidos" todos os parênteses o micro trabalha ao longo da linha do programa da esquerda para a direita fazendo primeiro a *exponenciação*, depois a multiplicação ou a divisão e final

mente começando da esquerda para a direita a adição ou a subtração.

Observação importante

Adiante aprenderemos o uso de algumas funções como INT, RND, ABS, SQR, SIN, etc. e veremos que elas serão executadas antes da exponenciação, multiplicação, divisão, etc.

Exemplo 2.6

Para converter a temperatura em graus centígrados ou seja graus Celsius (que é a escala usada no Brasil) para graus Fahrenheit (que é a escala comumente usada nos Estados Unidos da América do Norte) utiliza-se uma fórmula de conversão muito simples ou seja

$$F^{\circ} = \frac{9}{5} C^{\circ} + 32$$

Suponhamos que hoje tem-se a temperatura de 35°C (que aliás é uma boa temperatura para se "chupar" um sorvete) e queremos converter a mesma em graus Fahrenheit.

O programa para esta finalidade é:

```
10 REM *** CONVERSÃO DE CELSIUS PARA FAHRENHEIT ***
20 LET C=35
30 LET F=(9/5)*C+32
40 LPRINT C;"GRAUS CELSIUS=";F;"GRAUS FAHRENHEIT."
RUN
```

Teremos na saída

35 GRAUS CELSIUS= 95 GRAUS FAHRENHEIT.

Em primeiro lugar observe estimado leitor ou leitora que a linha 40 consiste de uma instrução LPRINT (poderia também ter sido PRINT) seguida por quatro expressões separadas - duas variáveis numéricas (C e F) e dois grupos de palavras entre aspas chamados "literais" ou STRINGS.

Deve-se a seguir observar como foram colocados os parênteses ou seja com o 9/5 dentro dos mesmos estamos seguros que estamos multiplicando o seu quociente vezes C e então adicionamos 32.

Caro leitor(a) veja o que ocorre se na linha 30 você remover os parênteses.

Não acontece nada de especial não é?

Realmente como a operação começa da esquerda para a direita e divisão e multiplicação tem a mesma "força" ou hierarquia então neste caso não há nenhuma necessidade de parênteses.

Se porém quiséssemos a temperatura Celsius sendo dada a Fahrenheit, então a expressão de conversão seria:

$$C^{\circ} = \frac{5(F^{\circ} - 32)}{9}$$

e neste caso quando escrevermos o programa para obter a temperatura C° quando a Fahrenheit é 100 os parênteses são necessários.

```
10 REM *** CONVERSÃO DE FAHRENHEIT PARA CELSIUS ***
20 LET F=100
30 LET C=5*(F-32)/9
40 LPRINT F;"GRAUS FAHRENHEIT=";C;"GRAUS CELSIUS."
RUN
```

Teremos como saída:

100 GRAUS FAHRENHEIT = 37.7778 GRAUS CELSIUS.

Não esqueça portanto leitor ou leitora que os parênteses disciplinam a ordem de avaliação de expressões aritméticas compostas.

1ª Regra geral

Avalia-se a expressão no sentido

esquerda → direita

2ª Regra geral
Parênteses embutidos, em vários níveis devem ser avaliados no sentido

interno → externo

3ª Regra geral
Funções (veremos isto bem adiante) serão avaliadas antes de se efetuar as operações com o seu resultado.

4ª Regra geral
Entre dois operadores aritméticos consecutivos existe uma precedência, que será estabelecida segundo a seguinte escala de prioridade (Tabela 2.1).

Nível de precedência	Operador	Sinal
1	Potenciação	** ou ↑ ou ^
2	Multiplicação/Divisão	* /
3	Adição/Subtração	+ -

Tabela 2.1

5ª Regra geral

Operadores com a mesma precedência (nível 2 ou 3) serão avaliadas pela 1ª Regra geral.

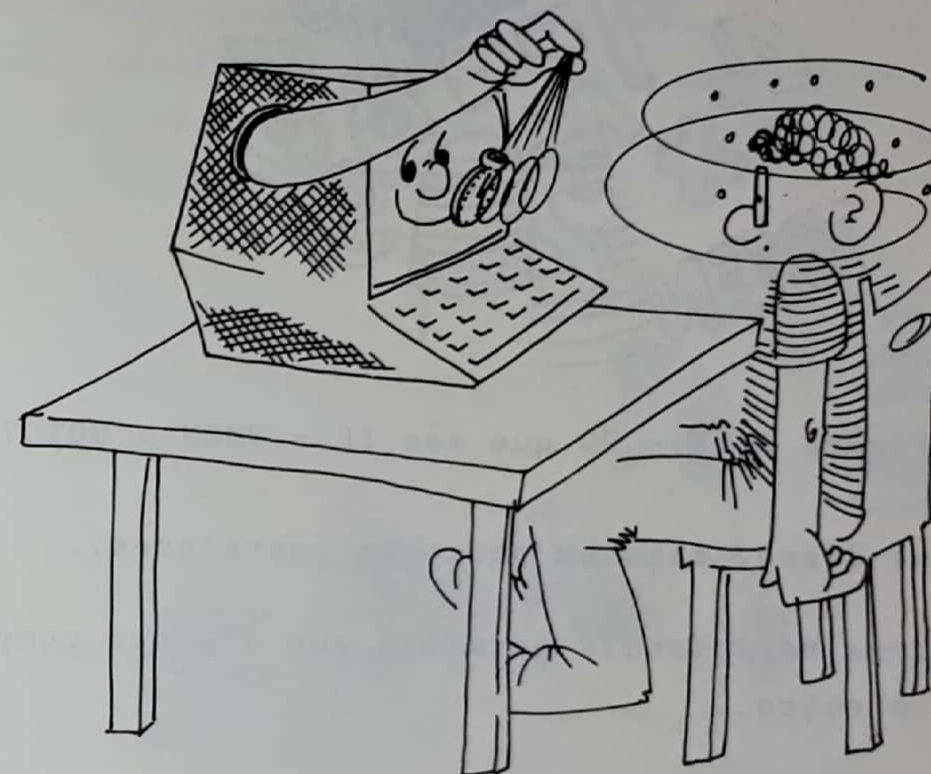


Figura 2.10 - Lembre bem as 5 regras da ordem de avaliação das expressões aritméticas compostas e assim estará sempre livre de cair na armadilha da "magia dos números".

§2.5 - OPERADORES RELACIONAIS - INSTRUÇÕES IF-THEN - GOTO

Se (IF) você gostou dos tópicos precedentes então (THEN) você seguramente gostará do resto (e que resto!) do livro!!

Isto vai acontecer porque passo a passo estamos nos encaminhando para as instruções IF - THEN e GOTO que farão com que o seu micro tome *decisões* e *ações*.

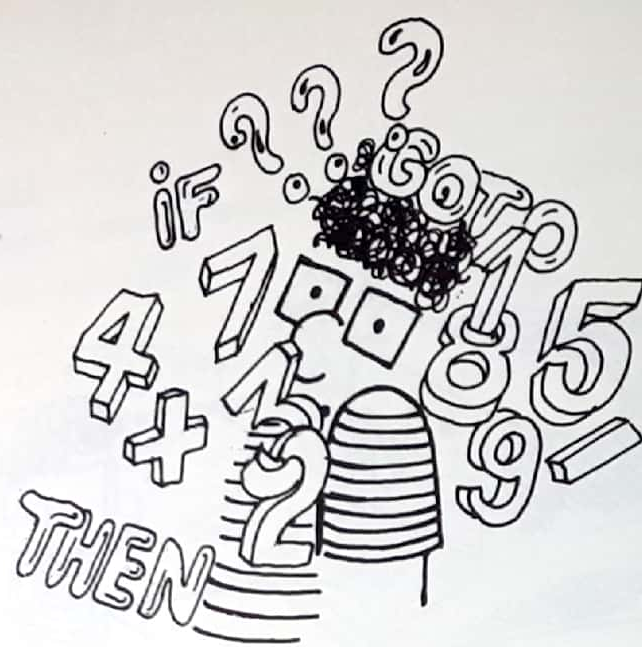


Figura 2.11 - "O que são IF - THEN e GOTO?"

Porém antes disso, mais um pouco de operadores...

Os operadores relacionais permitem que o micro compare um valor com o outro.

Existem apenas três operadores relacionais:

1. Igual, usando o símbolo (=)
(Como é que você advinhou isto?)
2. Maior do que, usando o símbolo (>)
3. Menor do que, usando o símbolo (<)

Combinando estes três símbolos, vem mais três operadores

4. Diferente de ou não igual, usando o símbolo < >
5. Menor do que ou igual a, usando o símbolo < =
6. Maior do que ou igual a, usando o símbolo > =

A instrução IF-THEN combinada com os operadores relacionais e aritméticos nos dará, a partir de agora, um grande poder de ação sobre um sistema de lógica.

A instrução **IF-THEN** é uma instrução condicional porque a ação tomada depende se a instrução condicional é considerada verdadeira ou falsa.

A forma mais simples da instrução IF-THEN é:

```
IF < condição > THEN <número de linha>
```

A "porção condicional" da instrução compara duas quantidades que precisam estar separadas por algum dos operadores relacionais há pouco estudados.



Figura 2.12 - O iniciante geralmente acha que a "construção" IF-THEN é um conceito difícil. Experimente não fazer a lição seu "raposão" e então você entenderá bem o IF-THEN.

Por sua vez a instrução GOTO é uma instrução de desvio incondicional e a sua forma geral é

```
GOTO < número de linha >
```

O desvio pode ser feito para qualquer instrução do programa seja executável ou não executável, seja desvio para trás ou para frente.

No caso de desvio para uma instrução executável, a execução do programa continua com esta instrução *inclusive*.

No caso de desvio para uma instrução não executável, a execução prosseguirá com a próxima instrução executável.

O número de linha para desvio tem que ser necessariamente um número de linha do mesmo programa.

Comumente, a execução do programa continuará com a instrução para a qual foi feito o desvio.

Exemplo 2.7

Suponhamos que você deseja fazer um programa para saber o que é melhor em termos de retorno de investimento ter 16% de 200 mil cruzeiros ou 14% de 230 mil cruzeiros.

Um programa usando a instrução IF-THEN resolve o seu problema.

```
10 LET X=.16*200000
20 LET Y=.14*230000
30 IF X>Y THEN 90
40 IF X=Y THEN 70
50 LPRINT ".14 E MELHOR"
60 GOTO 100
70 LPRINT "AMBOS SÃO IGUAIS"
80 GOTO 100
90 LPRINT ".16 E MELHOR"
100 .END
```

Em alguns micros como o TK-85 e o CP-200 é necessário após o THEN o GOTO e depois o número da linha.



Teremos como saída:

.14 E MELHOR

Não esqueça que se você quer a saída na tela do vídeo substitua o LPRINT por PRINT.

Repetindo, como se vê a instrução IF-THEN é uma instrução de "ramificação" condicional.

Quando se tem o IF-THEN o programa se "desviará" para uma outra parte do mesmo sob a condição de que ele passe no teste ao qual é submetido.

"PEQUENO GÊNIO :
OU VOCÊ VEM COMER
IMEDIATAMENTE
OU LHE MOSTRAREI
O QUE É UMA
INSTRUÇÃO DE
RAMIFICAÇÃO OU
ALTERNATIVA".



Figura 2.13

Se ele "falhar" no teste ao qual está sendo submetido, simplesmente *continuará na próxima linha*.

Já o GOTO é uma instrução de desvio incondicional e no nosso caso particular, há pouco analisado, o micro *obedecia incondicionalmente* o desvio para o número de linha 100.

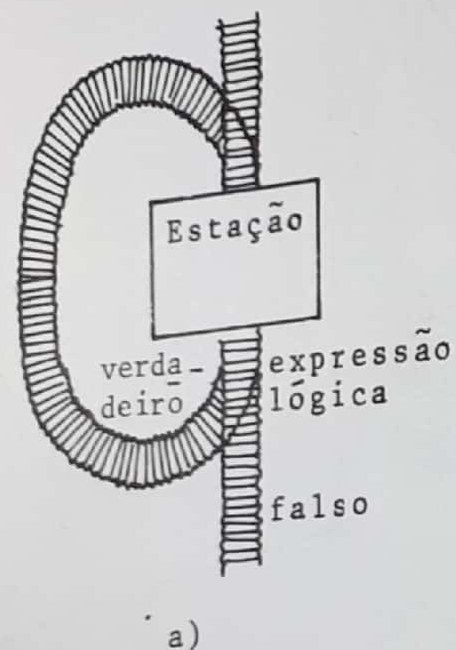
Quem não brincou de autorama em alguma fase da sua vida?

Bom, quase todos brincaram e os que não brincaram devem fazer isto o mais breve possível porém no momento fiquem atentos aos diversos casos das instruções IF-THEN e GOTO que são indicados com o auxílio de uma linha férrea.

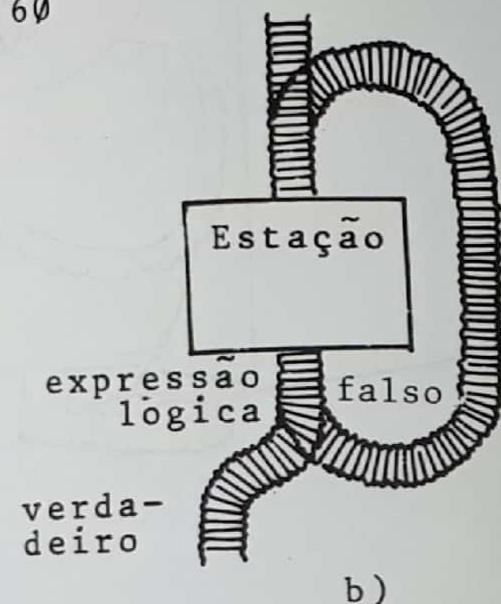
Além destes seis casos dificilmente você terá mais algum, portanto fique sempre atento a Figura 2.14 que ela é uma grande "dica" para fazer futuros programinhas.

Alguns tipos de "loops" ou seja laços usando IF-THEN e GOTO.

```
10
20
30
40 IF expressão lógica THEN 10
```



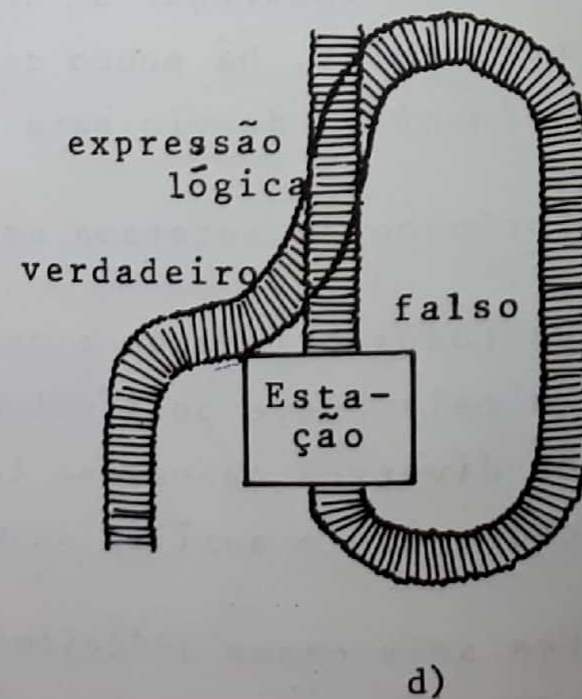
```
10
20
30
40 IF expressão lógica THEN 60
50 GOTO 10
60
```



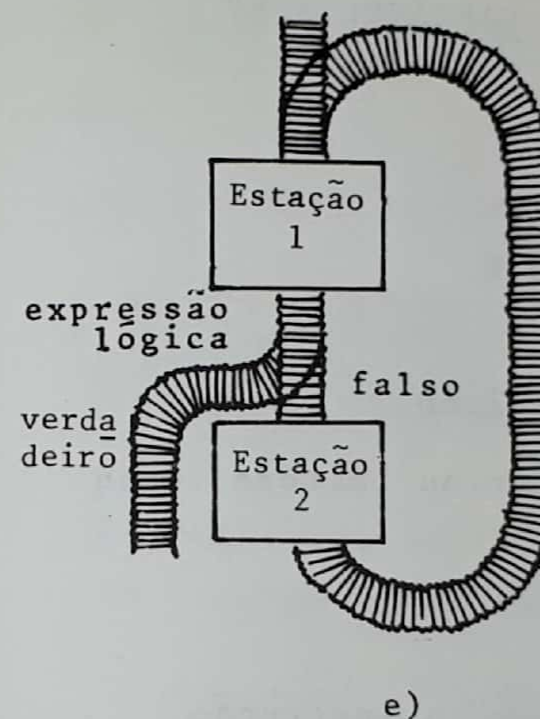
```
10 IF expressão lógica THEN 30
20 GOTO 70
30
40
50
60 GOTO 10
70
```



```
10 IF expressão lógica THEN 60
20
30
40
50 GOTO 10
60
70
```



```
10
20
30
40 IF expressão lógica THEN 80
50
60
70 GOTO 10
80
```



```
10
20
30
40 IF expressão lógica THEN 60
50 GOTO 90
60
70
80 GOTO 10
90
```

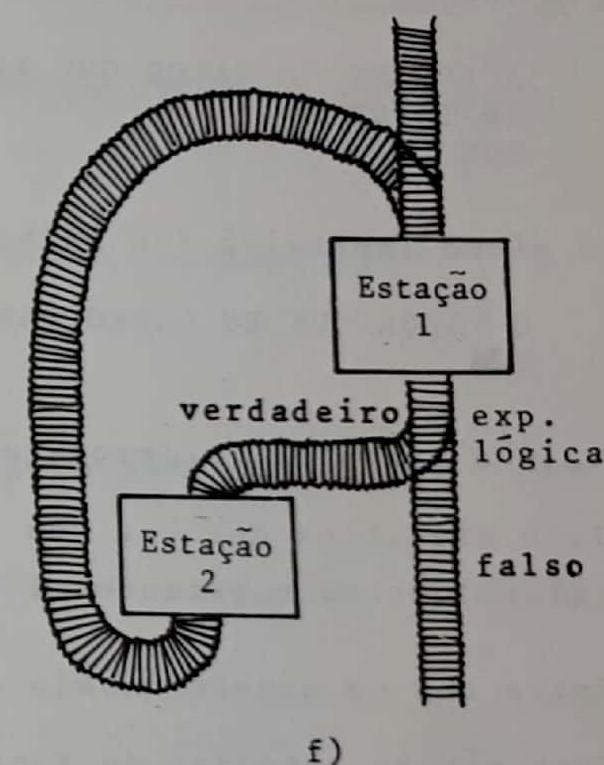


Figura 2.14

Para finalizar o IF-THEN e o GOTO que tal a metáfora visual da Figura 2.15.



O garfo é a "metáfora da estrada" para a instrução IF-THEN.

§2.6 - O MICRO "FALA" E ESCUTA - A INSTRUÇÃO INPUT

Exemplo 2.8

Digamos que se tenha o seguinte programa residente isto é aquele que está dentro do microcomputador:

```
5 LPRINT "O VALOR QUE EU QUERO DAR A VARIÁVEL X E"
10 INPUT X
RUN
```

O micro imprimirá (ou exibirá):

O VALOR QUE EU QUERO DAR A VARIÁVEL X E
? ■

Veja o ponto de interrogação na tela do vídeo.

Isto significa que é a sua vez caro leitor ou leitora e eu (micro) estou esperando...

Entre com um número e veja o que ocorre.

Faça várias entradas de X para "sentir bem" a instrução

INPUT.

Assim a função da instrução **INPUT** é a de ler dados a serem fornecidos pelo teclado.

A execução da instrução provocará o aparecimento de um ponto de interrogação "?" no vídeo.

O usuário deverá fornecer um valor (no nosso caso até agora numérico) de forma que deste momento em diante a variável numérica passará a ter o valor lido.

Portanto a função básica da instrução INPUT é a *entrada de dados* ao passo que o PRINT escrevia os dados na tela e o LPRINT provoca a saída de dados através da impressora.

Exemplo 2.9

Entre com o valor de X através da instrução INPUT e verifique se o mesmo é solução da equação $13X - 169 = 0$.

```
10 INPUT X
20 IF 13*X-169 = 0 THEN 50
30 LPRINT X;"NÃO É A SOLUÇÃO"
40 GOTO 10
50 LPRINT X;"É A SOLUÇÃO E NÃO ESQUEÇA QUE É UM NÚMERO DE
   SORTE"
RUN
```

O teu micro precisa neste ponto do GOTO

Uma possível "conversação" com o micro poderia ser:

```
1 NÃO É A SOLUÇÃO
9 NÃO É A SOLUÇÃO
18 NÃO É A SOLUÇÃO
13 É A SOLUÇÃO E NÃO ESQUEÇA QUE É UM NÚMERO DE SORTE
```

Observe neste exemplo a necessidade da linha 40.

Sem a mesma, o programa imprimirá

É A SOLUÇÃO E NÃO ESQUEÇA QUE É UM NÚMERO DE SORTE

mesmo quando a solução não seja o particular valor de X.

A sequência de números de linha que o micro segue quando o X é a solução é 10, 20, 50 e quando X não é a solução a sequência de números de linha é 10, 20, 30, 40, 10.

Exemplo 2.10

Vamos praticar a multiplicação.

Façamos antes um fluxograma:

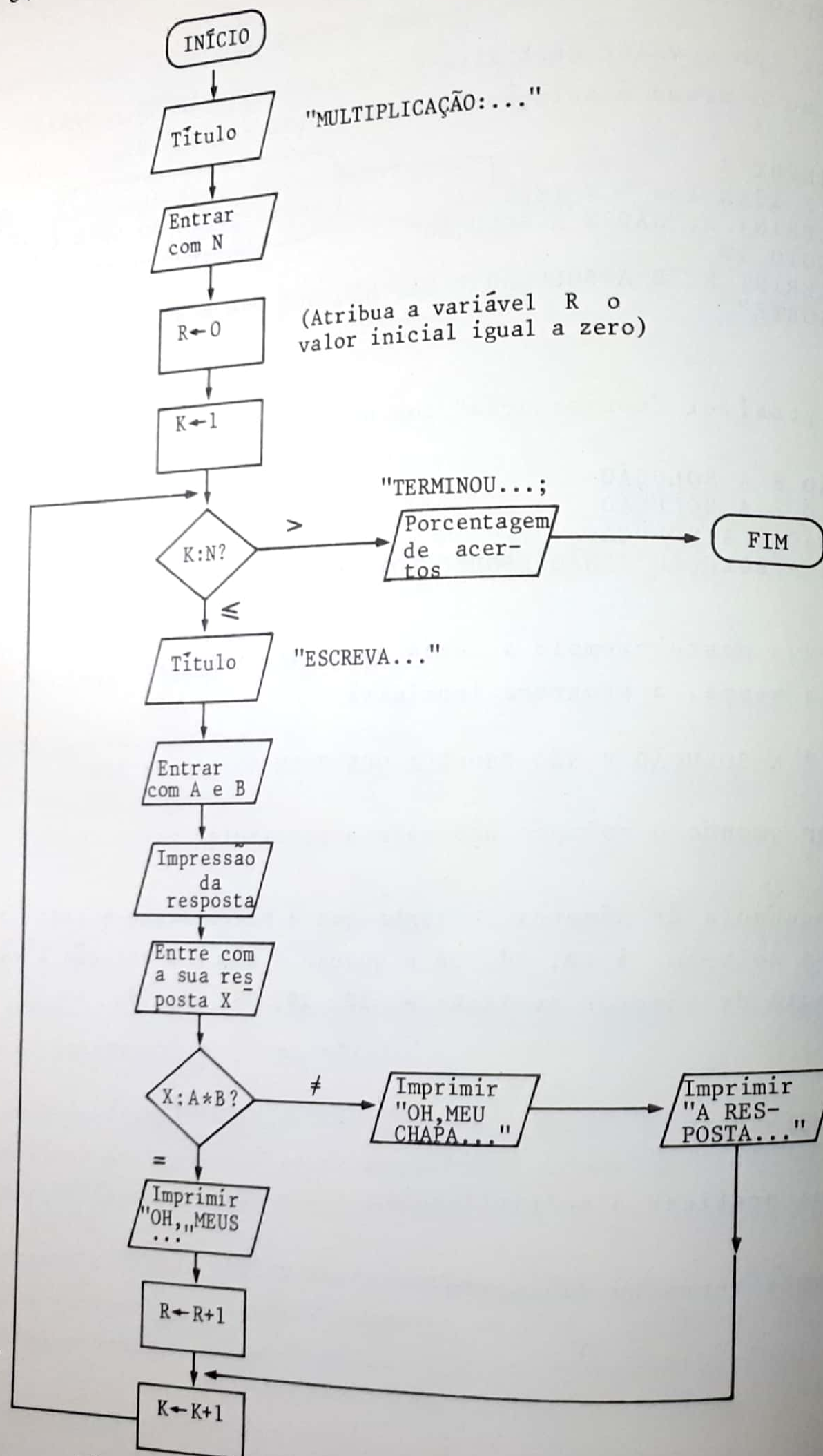


Figura 2.16 - Fluxograma para Exemplo 2.10

E aí vai o programa com uma possível saída conversacional.

```

10 PRINT "MULTIPLICAÇÃO: QUANTOS PRODUTOS SERÃO FEITOS";
20 INPUT N
30 LET R=0
40 LET K=1
50 IF K>N THEN 190
60 PRINT "ESCREVA OS 2 NÚMEROS";
70 INPUT A,B
80 PRINT "ESCREVA O PRODUTO.....";A;"*";B;"=";
90 INPUT X
100 IF X=A*B THEN 140
110 PRINT "OH, MEU CHAPA ISTO ESTA.....ERRADO!!!"
120 PRINT "A RESPOSTA E .....";A*B
130 GOTO 170
140 PRINT "OH, MEUS PARABENS!!!!!!!!!! ESTÁ CERTO!!!!!"
150 LET R=R+1
170 LET K=K+1
180 GOTO 50
190 PRINT "TERMINOU: A SUA PORCENTAGEM DE ACERTOS E...";
    R/N*100;"%"
200 END
RUN
  
```

0 seu micro precisa aqui do GOTO?

Uma possível conversação com a máquina poderia ser:

```

MULTIPLICAÇÃO: QUANTOS PRODUTOS SERÃO FEITOS? 3
ESCREVA OS 2 NÚMEROS ? 1,2
ESCREVA O PRODUTO..... 1 * 2 = ? 2
OH, MEUS PARABENS!!!!!!!!!! ESTÁ CERTO!!!!!!
ESCREVA OS 2 NÚMEROS ? 2,3
ESCREVA O PRODUTO..... 2 * 3 = ? 5
OH, MEU CHAPA ISTO ESTÁ ..... ERRADO!!!!
A RESPOSTA É ..... 6
ESCREVA OS 2 NÚMEROS ? 4,5
ESCREVA O PRODUTO ..... 4 * 5 = ? 20
OH, MEUS PARABENS!!!!!!!!!!!!!!!!!!!!!! ESTÁ CERTO!!!!!!
TERMINOU: A SUA PORCENTAGEM DE ACERTOS E... 66.6667 %
  
```

Você acha que o fluxograma da Figura 2.16 ajudou para entender o programa propriamente dito?

Estás na dúvida, não é?

Porém o seu micro é maravilhoso merecendo já um beijinho, tá ou melhor dá logo...



Figura 2.17

Continuando a exploração de todas as sutilezas do nosso es-
tonteante micro não podemos deixar de ressaltar que ele não dei-
xou de ser uma poderosa máquina de calcular.

Exemplo 2.11

Executar os seguintes cálculos

$$\begin{cases} A = X + Y + Z \\ B = X * Y + Z \\ C = X * Y + Z * 3 \end{cases}$$

sabendo que $X=4$, $Y=7$ e $Z=13$ sem utilizar a instrução INPUT.

Solução:

```
10 LET X=4
20 LET Y=7
30 LET Z=13
40 LET A=X+Y*Z
50 LET B=X*Y+Z
60 LET C=X*Y+Z*3
70 PRINT "X=";X;"Y=";Y;"Z=";Z;"A=";A;"B=";B;"C=";C
RUN
```

Teremos a seguinte saída na tela do vídeo

X=4 Y=7 Z=13 A=95 B=41 C=67

Note que $\text{ } \text{ } \text{ }$ é uma forma de representar espaço em branco.

Elimine caro leitor os espaços em branco na linha 70 e você em alguns micros poderá ter problemas.

Verifique isto no seu micro.

Por outro lado, fazendo inclusive uma simplificação você pode eliminar as instruções de números de linha 10, 20 e 30 usando agora uma instrução INPUT. Que tal a idéia?

De qualquer forma como treininho faça o programa que calcule

$$\begin{cases} a = x + \frac{y}{z} \\ b = \frac{x}{y} + z \\ c = \frac{x}{y} + \frac{z}{3} \\ d = x^2 + y^3 + z^4 \\ e = x^2 + 3y^3xz \\ f = xy^2 + xy^3z \\ g = \frac{x+3}{(y-4)(z+4)} \\ h = -x^3(-y)^3(-z)^2 \end{cases}$$

Adote que $x=4$, $y=7$ e $z=13$ e não esqueça que você já conhece a instrução INPUT.

Resposta:

Se você fez o programa certinho deve obter:

a) 4,53846	b) 13,5714	c) 4,90476	d) 28920
e) 13393	f) 18032	g) 0,137255	h) 3,70989.10 ⁶

Cuidado com a escrita do micro e a que você habitualmente aprendeu em Matemática.

§2.7 - OS OUTROS USOS DO PRINT

A instrução PRINT pode ser usada para diversas finalidades e uma delas é para imprimir resultados de cálculos definidos dentro da própria instrução PRINT.

Exemplo 2.12

Suponha que você precisa fazer os seguintes cálculos:

$$\begin{cases} 133 + 48 = \\ 33 - 42 = \\ 964 \times 640 = \\ \frac{13}{4,5} = \\ 12,3 \times 6^{4,2} = \end{cases}$$

Para se obter os resultados desejados basta escrever um programa como o que está abaixo:

```
10 LPRINT 133+48
20 LPRINT 33-42
30 LPRINT 964*640
40 LPRINT 13/4.5
50 LPRINT 12.3*6**4.2
RUN
```

Teremos na saída:

```
181
-9
616960
2.88889
22810.8
```

Meu caro "micromaniaco" nunca mais a sua "mania" ficará assim:



Figura 2.18

Como "treininho" vê o que sae com o seguinte "programinha":

```
1) 10 LPRINT 7*12*14*41*42
   20 LPRINT 1983/64/13
   30 LPRINT (24*23/83)/1999
   40 LPRINT (9.5+1.8)/(7.7-9.3)+7.6
   50 LPRINT ((4.8-7.6+1.3)/4.4)*(8.35/(5.5+4.2+3))
   60 LPRINT (13/(27+1.941))/(7/(9+1.941))
   RUN
```

Resposta:

```
780202.
702084.
752722.
1057501.
3.32697E-03
2.38341
2.02507E+06
```

Ainda dentro do mesmo "espírito" utilizando apenas a instrução PRINT execute os seguintes cálculos

a) $\frac{99,8}{7,7} \times \frac{44,4}{1,2555} =$

(não esqueça que a vírgula decimal é igual ao ponto decimal do BASIC).

$$b) \frac{\frac{7,7 + 13,7}{8}}{\frac{4,4}{1,7 - 3,8}} \times \frac{1942}{1,5 - 29} =$$

$$c) \frac{\frac{2,5}{8,9}}{\frac{9,6}{4,5}} =$$

$$d) \frac{6,66 (5^{3,11} + 4^{2,22})}{7,89 (6^{4,33} + 2^{5,44})} =$$

Quando você fizer o programa as respostas serão:

- a) 65E'857
- b) 9851'06
- c) 2E'8601
- d) 2905090'0

Exemplo 2.13

Vejamos agora um programa no qual na mesma linha do PRINT estão incluídas mensagens literais e valores numéricos.

```
10 LET A=2670
20 LET B=42
30 LET C=4890
40 LPRINT "O VALOR DE C É";C
50 LPRINT "O VALOR DE B*C É";B*C
60 LPRINT "O VALOR DE W=A*B*C É";A*B*C
70 END
RUN
```

Teremos como saída:

```
O VALOR DE C É 4890
O VALOR DE B*C É 205380
O VALOR DE W=A*B*C É 5.48365E+08
```

Exemplo 2.14

```
10 LET A=2.7
20 LET X=19.1
30 LET G=33.4
40 LET Z=45.6789
50 LET K=888
60 LET D=11.1111
```

```
70 LPRINT A,X;G,Z;K,D
80 LPRINT A;X;G;Z;K;D
90 LPRINT A;X,G;Z,K;D
100 END
RUN
```

```
2.7          19.1 33.4          45.6789 888 11.1111
2.7 19.1    33.4    45.6789    888 11.1111
2.7 19.1          33.4 45.6789    888 11.1111
```

Nos Exemplos 2.13 e 2.14 vírgulas e pontos e vírgulas foram utilizadas na instrução LPRINT.

É interessante destacar que as saídas exibidas foram obtidas no CP-500 e pode ser que no seu micro elas sejam diferentes. É possível isto?

De qualquer forma não esqueça que a vírgula faz avançar para a próxima zona do vídeo ou do papel (formato aberto) ao passo que o formato produzido pelo ponto e vírgula é um formato fechado.

Você já teclou estes dois últimos exemplos?

Bem, então já está nítida a diferença entre os delimitadores vírgula (,) e ponto e vírgula (;).

Ótimo, vamos em frente!

Exemplo 2.15

Como uma última e variada aplicação da instrução PRINT ou LPRINT vejamos como se pode imprimir o seguinte desenho:

```

      *
    * *
  *   *
 *   *
 *
 * *
*   * *
*   *
* * *
* *
*

```

Figura 2.19

Solução:

O programa para obter o desenho da Figura 2.19 é:

```

10 LPRINT "  *  "
20 LPRINT " * *  "
30 LPRINT " * *  "
40 LPRINT " * *  "
50 LPRINT " * *  "
60 LPRINT " * * * "
70 LPRINT " * * * "
80 LPRINT " * * * "
90 LPRINT " * * * "
100 LPRINT " * * * "
110 LPRINT " * * * "
120 END
RUN

```

TRABALHO PARA VOCÊ

- 1) Usando não mais de 11 instruções PRINT ou LPRINT faça a inicial do seu nome e do seu sobrenome.

Sugestão: Convém fazer antes um quadriculado para "perceber" a posição correta dos asteriscos e dos espaços em branco ().

- 2) Agora que você já fez o exercício 1, o imediatamente anterior está apto, tendo muita paciência para esboçar o rosto do seu amigo ou amiga (veja a sugestão da Figura 2.20) e fazer um esboço do mapa da América do Sul, (veja a Figura 2.21).

Muita paciência e boa sorte bicho!

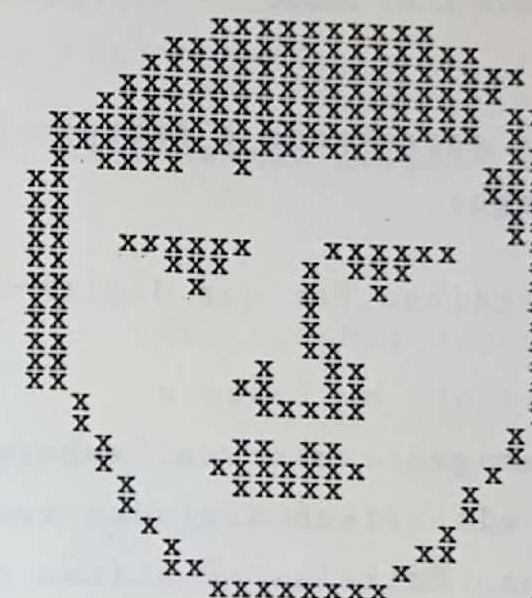


Figura 2.20

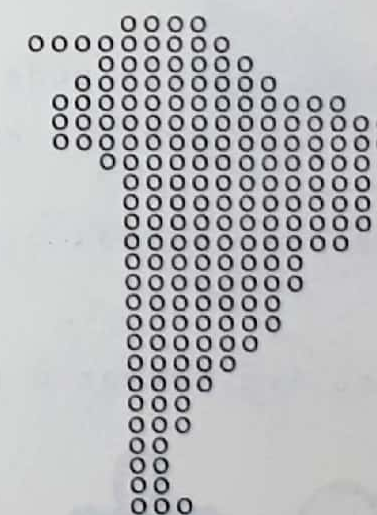


Figura 2.21

§2.8 - O USO DA FITA CASSETTE E DO DISCO

Logo, logo você estará escrevendo e fazendo "rodar" programas sofisticados e longos.

Torna-se então muito tedioso ter que digitá-los toda vez que quiser utilizá-los.

Impressionar os seus amigos com o seu "super-micro" torna-se uma tarefa difícil se eles ficam sentados vendo no aparelho de T.V. a Guerra nas Estrelas nº 4 (não chegou ainda?) enquanto você fica mais de uma hora para teclar todo o seu "original programa".

Deve haver uma forma mais simples para isto.

Praticamente todos os micros permitem que nos gravemos e armazenemos qualquer programa em uma fita cassete de boa qualidade.

Geralmente um total de 4K de memória pode ser embutido em uma fita ou carregado dela para o micro em apenas 3 minutos.

A maior parte dos programas são curtos e levam muito menos tempo.

Agora você já percebeu como irá formar a sua "biblioteca" de programas favoritos!!!



Figura 2.22

Porém não force demais as suas fitas para não acontecer o que se indica na Figura 2.22.

Neste momento é interessante definir alguns comandos do BASIC.

SAVE ou CSAVE → Este comando tem como função salvar (armazenar) um programa que se encontra na área de trabalho (na memória) para um arquivo do sistema (que tem o nome do programa) em disquete. Quando está em cassete é que se usa o comando CSAVE.

LOAD ou CLOAD → Este comando tem como função carregar na memória um programa a partir de um arquivo do sistema (que tem o nome do programa em disquete). Quando está em cassete é que se usa o comando CLOAD.



Figura 2.23

Com todos esses recursos não se "enrole" e venha a sofrer um acidente no trabalho.

§2.9 - ENFIM O PAR DE INSTRUÇÕES FOR-NEXT!!!

A maior diferença entre o *micro* e uma *máquina de calcular* é a capacidade do micro de fazer a mesma coisa muitas e muitas vezes e bem mais rápido que qualquer foguete que sobe

ao céu levando um satélite ou uma estação orbital.

Esta capacidade, mais que qualquer outra, separa bem o micro de uma simples máquina de calcular.

O laço FOR-NEXT é de importância fundamental para se poder colocar o nosso micro no trabalho em qualquer área.

A sua simplicidade (do par FOR-NEXT) e as suas variações são o centro da eficiência e do desconcertante poder do micro (Figura 2.24).

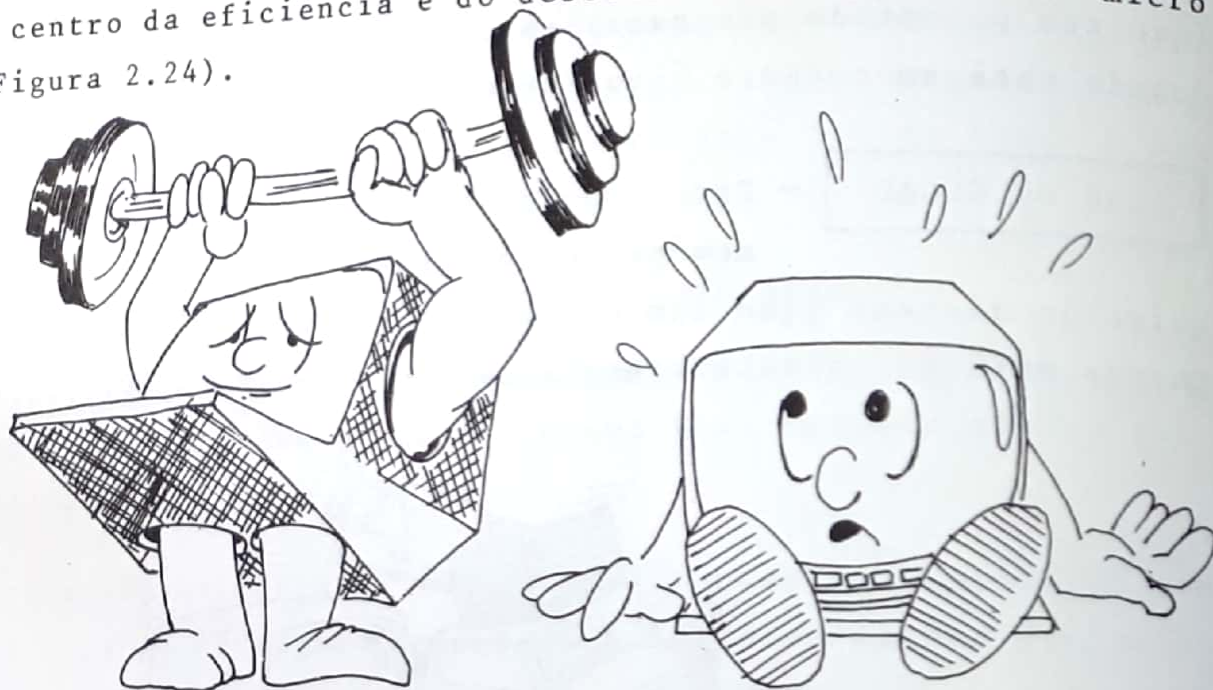


Figura 2.24 - "Parece que o micro da esquerda conhece bem o par FOR-NEXT".

Definição:

FOR-NEXT

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

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

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



Figura 2.25 - Esta é uma imagem de um ciclo ou seja de um "LOOP" que só termina quando fecha o parque de diversões.

Exemplo 2.15

Faça com que o seu micro escreva em 5 linhas consecutivas.

"SOCORRO O MEU MICRO ESTÁ FURIOSO" usando apenas uma instrução PRINT ou LPRINT.

```
7 FOR N=1 TO 5
10 LPRINT "SOCORRO O MEU MICRO ESTÁ FURIOSO"
20 NEXT N
30 END
RUN
```

Teremos a saída:

```
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
```


O laço FOR-NEXT (o laço em Inglês significa LOOP) criado nas linhas 7 e 20 faz com que o micro passe em ciclos através das linhas 7, 10 e 20 exatamente 5 vezes e então faz o resto do programa.

Cada vez que o micro atinge a linha 20 ele "vê" o NEXT N.

A palavra NEXT (próximo) faz com que o valor de N seja aumentado de exatamente uma unidade e o micro incondicionalmente é mandado de volta para a instrução FOR N iniciando um novo ciclo, laço ou LOOP.

A instrução NEXT é condicional para N enquanto o mesmo é menor que 5, porque a linha 7 diz FOR N=1 TO 5.

Este TO é que significa até 5.

Depois de ter sido executada a 5ª volta consecutiva o teste embutido na instrução falha ou seja não é verificado e a execução do programa move-se para a 1ª instrução depois do NEXT N.

Desta forma o par de instruções FOR-NEXT tem condição de por fim ao laço ou seja a repetição nele é controlada.

Definição:

A função STEP

Há ocasiões em que se deseja um incremento para o FOR-NEXT diferente de um.

Para esta finalidade utiliza-se a função STEP.

Se no programa do Exemplo 2.15 tivermos na linha 7

```
7 FOR N=1 TO 5 STEP 2
```

veremos que a linha 10 será impressa somente 3 vezes pois teremos N=1, N=3 e N=5.

Na primeira passagem, através do programa, quando o NEXT N for atingido o mesmo é aumentado ou incrementado de 2 no lugar de 1 como foi no caso inicial.

Na segunda passagem pelo LOOP, N vai com o valor 3 e na terceira e última passagem vai com N igual a 5.

Os "LOOPS" FOR-NEXT podem ser incrementados por qualquer número decimal e mesmo por números negativos.

Porque alguém vai querer incrementos com números negativos, pode parecer no momento um pouco vago porém também isto será entendido com o passar do tempo.

Neste interim, se fizermos uma nova mudança na linha 7, tendo agora a instrução

```
7 FOR N=5 TO 1 STEP -1
```

e o resto do programa como antes, novamente teremos cinco passagens pelo LOOP, diminuindo o N em cada passagem de 5 até 1.

Desta forma a linha 10 outra vez aparecerá impressa cinco vezes.

Mude agora o passo (STEP) de -1 para -2,5 e aperte ou tecle o RUN novamente.

Surpresa! A linha 10 será impressa exatamente duas vezes.

Oh, obediente e "esperto" micro você é demais!!!

Num programa mais complexo, geralmente teremos muitos laços FOR-NEXT e muitas vezes um embutido dentro do outro sendo a sua execução sempre do mais interno para o mais externo.

Na realidade não existe nada de mágico no laço FOR-NEXT e você já aprendeu uma forma de fazer a mesma coisa que se consegue com este par de instruções, não faz muito tempo!

Você acha que existe equivalência entre o programa abaixo e

aquele que inicialmente serviu para exemplificar o par de instruções FOR-NEXT?

Exemplo 2.16

```
7 LET N=1
10 LPRINT "SOCORRO O MEU MICRO ESTÁ FURIOSO"
15 LET N=N+1
20 IF N<6 THEN 10
30 END
RUN
```

Vê-se claramente que na linha 7 se "inicializa" o valor de N, dando-lhe um valor de começo igual a 1.

Na linha 15 incrementa-se o N de 1 em relação ao seu valor anterior.

Na linha 20 usa-se um dos nossos tradicionais operadores relacionais (<) para verificar se o novo valor de N está dentro dos limites que foram estabelecidos.

Caso não esteja o teste "falha" e o programa continua...

Bem a saída que você obteve foi a seguinte:

```
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
SOCORRO O MEU MICRO ESTÁ FURIOSO
```

No caso do Exemplo 2.16 deve-se notar que no processo de incrementar e testar, nós não enviamos mais o programa de volta para a linha 7 como foi feito no caso do par de instruções FOR-NEXT (Exemplo 2.15).

O que ocorreria se tivéssemos feito isto?

Resposta

Nós reinicializaríamos o valor de N como sendo igual a 1 e teríamos um laço ou seja um LOOP sem fim.

O oposto de *incrementar* é *decrementar*.

Desta forma ao se decrementar torna-se algo menor.

Como treino faça um programa análogo ao último usando decremento no lugar de incremento.

Será que você vai conseguir?

Vamos fazer o FOR-NEXT trabalhar mais!!!

É claro que não é muito excitante, para não dizer que é muito chato fazer a mesma coisa dia após dia e é dentro deste contexto que podemos afirmar que o uso ou a finalidade para o qual se destina o FOR-NEXT é muito mais nobre.

Exemplo 2.17

Suponhamos que a distância entre Micrópolis e Bytopolis é de 1600 km e que pode-se voar entre as duas cidades em diferentes aviões desde aqueles mais lentos que fazem 400 km/h até aqueles muito rápidos que fazem 1800 km/h.

Adotando que se pode variar a velocidade de 200 em 200 km/h qual é o tempo gasto no percurso entre as duas cidades?



Figura 2.26

Solução

```
10 REM ** TABELA DE TEMPO X VELOCIDADE **
20 CLS
30 LET D=1600
40 LPRINT "MICROPOLIS PARA BYTOPOLIS"
50 LPRINT
```



```

60 LPRINT "VELOCIDADE", "TEMPO", "DISTÂNCIA"
65 LPRINT "(KM/H)", "(HORAS)", "(KM)"
70 LPRINT
80 FOR V=400 TO 1600 STEP 200
90 LET T=D/V
100 LPRINT V,T,D
110 NEXT V
120 END
RUN

```

Teremos a seguinte saída:

MICROPOLIS PARA BYTOPOLIS

VELOCIDADE (KM/H)	TEMPO (HORAS)	DISTÂNCIA (KM)
400	4	1600
600	2.66667	1600
800	2	1600
1000	1.6	1600
1200	1.33333	1600
1400	1.14286	1600
1600	1	1600

Observação importante

Na linha 20 temos CLS (Clear Screen).

Esta é uma instrução para limpar a tela ou seja apagar todos os caracteres que estão na tela.

A instrução CLS é muito útil e com ela nos vamos "piscar" bastante na tela do seu vídeo.

É claro que neste caso convém, quando se usa o CLS usar no programa o PRINT e não o LPRINT.

Vamos agora testar a sua criatividade!!!

"Probleminha" para você

- 1) Você tem 3 rolos de arame cada um de 1000 m e deseja saber qual deve ser a largura (L) e o comprimento (C) de um terreno retangular que você quer cercar para colocar os seus lindos novinhos para ter área interna máxima.

A cerca terá três fios.

Elabore um programa que indique os valores ótimos de L e C com erro inferior a 2 m.

Sugestão: Utilize o método da "força bruta" ou seja varie os valores de L e C de 2 em 2 metros começando de convenientes valores de L e C e não esqueça que a área de um retângulo (A) é dada por $A = L \times C$.



Figura 2.27

"Não derrame lágrimas de crocodilo se o último "probleminha" não deu certo. Dê uma olhada no Exemplo 2.18. Quem sabe irá ajudar uma na tentativa com sucesso".

Exemplo 2.18

Agora, antes de você ver, já será testada a sua habilidade no "teorema da transferência máxima de energia de um circuito de corrente contínua".

Uma fonte de corrente contínua pode ser um gerador ou uma bateria e uma carga pode ser uma lâmpada ou um resistor.

Sabe-se que a potência máxima é enviada a uma carga elétrica quando a resistência da carga é igual a resistência interna da fonte.

Vamos verificar isto para o circuito da Figura 2.28 fazendo a resistência da carga variar de 1 a 20 ohms.

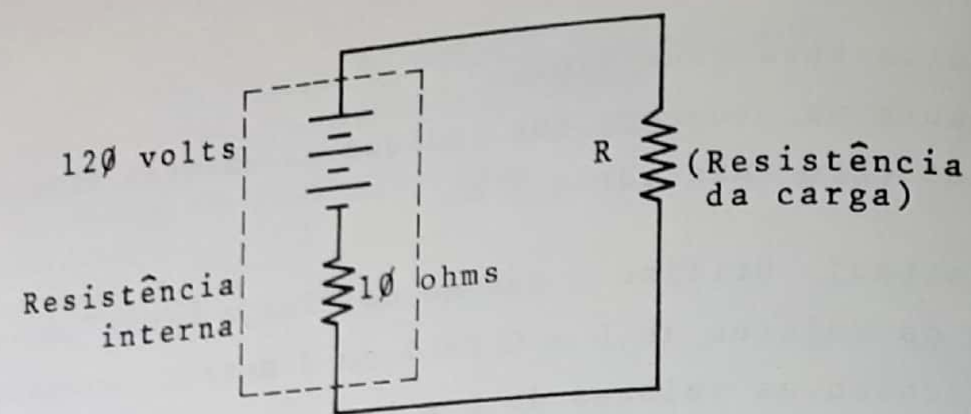


Figura 2.28

Deve-se ressaltar que:

- 1) A potência total do circuito é $I^2(10+R)$ onde I é a intensidade de corrente elétrica no circuito.
- 2) A potência perdida na fonte é $10I^2$ ou seja 10 vezes o quadrado da corrente.
- 3) A potência entregue a carga é RI^2 .
- 4) A intensidade da corrente I é obtida da lei de Ohm ou seja $I = \frac{120}{10+R}$

Solução:

```

10 REM ** PROCURA DA CARGA ÓTIMA **
20 CLS
30 LPRINT "CARGA","CIRCUITO","FONTE","CARGA"
40 LPRINT "RESISTÊNCIA","POTÊNCIA","POTÊNCIA","POTÊNCIA"
50 LPRINT "(OHMS)","(WATTS)","(WATTS)","(WATTS)"
60 LPRINT
70 FOR R=1 TO 20
80 LET I=120/(10+R)
90 LET C=I**2*(10+R)
100 LET F=I**2*10
105 REM ** Q É A VARIÁVEL CARGA **
110 Q=I**2*R
120 LPRINT R,C,F,Q
130 NEXT R
140 END
RUN

```

Ter-se-á a seguinte saída:

CARGA RESISTÊNCIA (OHMS)	CIRCUITO POTÊNCIA (WATTS)	FONTE POTÊNCIA (WATTS)	CARGA POTÊNCIA (WATTS)
1	1309.09	1190.08	119.008
2	1200	1000	200
3	1107.69	852.071	255.621
4	1028.57	734.694	293.878
5	960	640	320
6	900.001	562.5	337.5
7	847.059	498.27	348.789
8	800	444.445	355.556
9	757.895	398.892	359.003
10	720	360	360
11	685.714	326.531	359.184
12	654.546	297.521	357.025
13	626.087	272.212	353.875
14	600	250	350
15	576	230.4	345.6
16	553.846	213.018	340.828
17	533.334	197.531	335.803
18	514.286	183.673	330.612
19	496.552	171.225	325.327
20	480	160	320



Figura 2.29 - "É O USO DOS LAÇOS FOR-NEXT QUE SEPARA OS HOMENS DAS CRIANÇAS E OS MICROCOMPUTADORES DE SIMPLES CALCULADORAS".

\$2.10 - COMO É BOM CONHECER-TE TAB!

Depois destas últimas páginas o leitor ou leitora avesso à Matemática merece um pouco de descanso.

Recordemos as três formas que já aprendemos para montar o nosso formato de saída através da instrução PRINT.

Assim podemos:

- I) Mandar imprimir o que quisermos desde que esteja na instrução PRINT e entre aspas.
- II) Separar os "itens" da instrução PRINT com pontos e vírgulas de forma que a impressão saia bem próxima uma da outra e na mesma linha.
- III) Separar os "objetos" da instrução PRINT com vírgulas para imprimir as mesmas na mesma linha, geralmente em quatro zonas diferentes de impressão.

Existe uma quarta forma, que é usar a *função TAB*.

O efeito da tabulação é análogo ao que existe nas máquinas de escrever.

A função TAB provoca o salto do cursor na linha de impressão, para uma posição especificada, na mesma linha, onde será iniciada a colocação dos dados.

A função TAB não consegue fazer o cursor saltar para trás.

A função TAB é extremamente útil quando na saída temos colunas de números com cabeçalhos.

A forma geral da função de tabulação da saída é dada por:

TAB(< expressão aritmética ou simplesmente um número inteiro >)

Deve-se observar qual a pontuação ou delimitador que segue uma função TAB, pois determinará a colocação do valor que segue, sem espaço se for (;) ou na próxima zona se for (,).

Conforme o tipo de vídeo pode-se ter como extremo da esquerda TAB(0) e como extremo da direita TAB(63) ou então uma posição menor.

No seu pequeno, porém esperto micro, aliás o esperto é você, repita os seguintes programas:

Exemplo 2.19

```
a) 10 PRINT TAB(13):"ESTA":TAB(20);"E";TAB(27);"A FUNÇÃO TAB"
    20 END
    RUN
```

Teremos a saída

ESTA É A FUNÇÃO TAB

Se dentro do PRINT tivermos variáveis e a função TAB então em geral a pontuação após as variáveis é irrelevante, porém após a função TAB é levada em consideração para a colocação do valor que vem a seguir.

b) Este programa desenha um triângulo na tela do vídeo.

```
10 PRINT TAB(10), "*****"
20 FOR X=1 TO 5
30 PRINT TAB(X+10); "*"; TAB(16); "*"
40 NEXT X
50 PRINT TAB(16); "*"
60 END
    RUN
```

Teremos na saída:

```
*****
 *      *
 *      *
 *      *
 *      *
 *      *
 *      *
 *      *
```


Faça agora as seguintes modificações no seu programa e veja o seu triângulo piscar

```

5 CLS
10 .
20 .
30 .
40 .
50 .
60 GOTO 5

```

Execute esse programinha cerebral leitor.

Pisca ou não?

c) Programa para obter a marca do Zorro

```

100 LPRINT TAB(15); "*****"
110 LET X=22
120 LET X=X-1
130 LPRINT TAB(X); "*"
140 IF X>15 THEN GOTO 120
150 LPRINT TAB(X); "*****"
160 END

```

Ter-se-á seguinte saída:

```

*****
      *
      *
      *
      *
      *
      *
      *
      *
*****

```

TRABALHO PARA VOCÊ

1) Sacha Cerebral, um entusiasta dos micros de apenas 7 anos escreveu o seguinte programa:

```

10 FOR A=1 TO 4
20 PRINT TAB(A);A
30 NEXT A
40 FOR B=1 TO 4
50 PRINT TAB(B),B
60 NEXT B
70 END
RUN

```

Repita este programa no seu sagaz micro e assim definitivamente você sentirá a função TAB e a influência do ponto e vírgula ou da vírgula depois da TAB.

- 2) Que tal você obter na tela do vídeo, em homenagem ao autor do livrinho que está usando a inicial do seu nome. Não deixe de usar a função TAB. Após isto tente a inicial do próprio nome.
- 3) Escreva um programa em BASIC que produza a seguinte saída.

```

MICRO
MICRO
MICRO
MICRO
MICRO
MICRO
MICRO
MICRO

```

- 4) Que tal você agora, para realmente ficar seguro sobre a função TAB "produzir" um programa para obter um triângulo equilátero estando o mesmo centrado em TAB(15).



Figura 2.30 - É isso aí "cara" É preciso também gastar um pouco de massa cinzenta e fazer uns "calos" na "poupança"...

§2.11 - O FILHO, O NETO, O BISNETO, ETC. DO PAR FOR-NEXT

Nós já tivemos algumas informações sobre o que se pode fazer com as instruções FOR...TO...STEP, NEXT e agora vamos mostrar como podem aparecer os vários laços ou "LOOPS" e qual é a forma que deve ser utilizada para que se possa colocar um dentro do outro.

Examinaremos daqui para frente laços FOR-NEXT aninhados, embutidos ou seja um dentro do outro.

Exemplo 2.20

```

10 FOR X=1 TO 3
20 LPRINT "LAÇO DO X"
30 FOR Y=1 TO 2
40 LPRINT "      LAÇO DO Y"
50 NEXT Y
60 NEXT X
70 END

```

Diagrama de aninhamento:

- 1ª malha e é a mais interna (LAÇO DO Y)
- 2ª malha, laço ou LOOP mais externo (LAÇO DO X)

O resultado é:

```

LAÇO DO X
  LAÇO DO Y
  LAÇO DO Y
LAÇO DO X
  LAÇO DO Y
  LAÇO DO Y
LAÇO DO X
  LAÇO DO Y
  LAÇO DO Y

```

A saída demonstra de forma clara a operação do laço FOR-NEXT aninhado ou seja embutido.

O aninhamento é usado da mesma forma que a sua mãe guarda, as vases, os copos uns dentro dos outros para economizar espaço ou como funcionam as engrenagens de um relógio no qual depois de uma volta completa do ponteiro dos segundos, o ponteiro dos minutos se desloca um pouco e só depois da volta completa do ponteiro dos minutos é que o ponteiro das horas se desloca para a próxima hora.

Hi, acho que o nosso moderno leitor não está entendendo nada pois o seu relógio é digital e não tem ponteiros...

Porém com aninhamento queremos dizer que é possível encaixar ou embutir objetos uns dentro dos outros e para retirá-los deve-se começar do mais interno para o mais externo.

Num modelo físico ou no mundo real pode não ser sempre assim...

Bem aí vão mais "programinhas" bem simples que tornarão evidente a vantagem de se usar instruções FOR-NEXT embutidas umas dentro das outras.

Exemplo 2.21

Vamos com o nosso prestativo micro imprimir uma parte da inesquecível "tabuada"!!!

```

10 FOR X=1 TO 5
20 FOR Y=1 TO 3
30 LPRINT X;"*";Y;"=";X*Y,
40 NEXT Y
50 LPRINT
60 NEXT X
70 END
RUN

```

Devemos ter a seguinte saída:

1 * 1 = 1	1 * 2 = 2	1 * 3 = 3
2 * 1 = 2	2 * 2 = 4	2 * 3 = 6
3 * 1 = 3	3 * 2 = 6	3 * 3 = 9
4 * 1 = 4	4 * 2 = 8	4 * 3 = 12
5 * 1 = 5	5 * 2 = 10	5 * 3 = 15

Depende do micro; pode ser que não saia isto?



Como se vê a variável Y toma os valores 1, 2, 3 cinco vezes devido ao LOOP da variável X que é feito cinco vezes com X tomando em cada volta respectivamente os valores 1, 2, 3, 4 e 5.

Agora espertinho (a) leitor (a) que tal elaborar toda a tabuada fazendo uma saída especial em que no cruzamento da linha e da coluna esteja o resultado da multiplicação desejada.

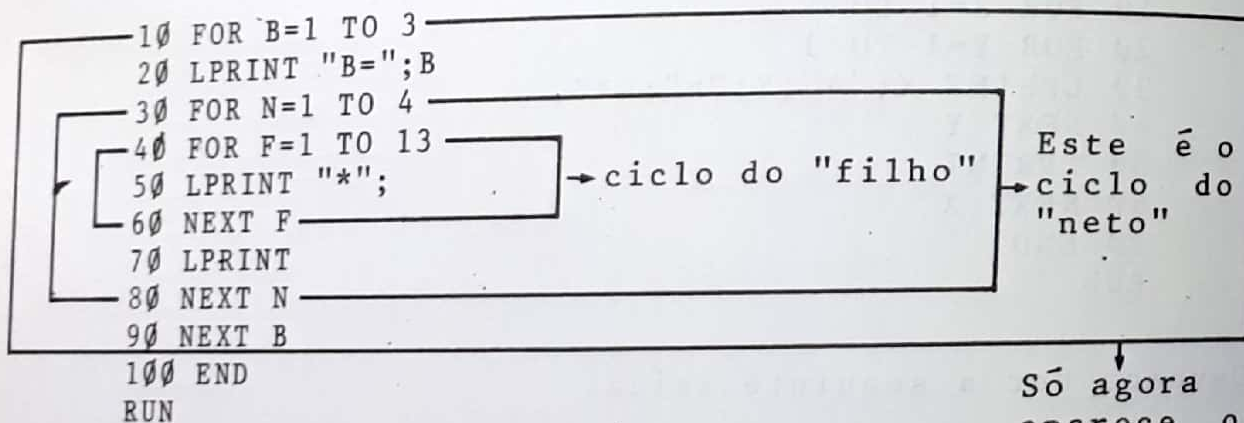
Não deixe o seu micro triste e ocioso, ele quer mostrar a você do que é capaz...



Figura 2.31

Exemplo 2.22

Lá vai um filho, neto e bisneto de algum pai (ou melhor talvez fosse falar de alguma mãe)!!!



Teremos a seguinte saída:

```

B=1
*****
*****
*****
*****
B=2
*****
*****
*****
*****
B=3
*****
*****
*****
*****

```

Observação importante:

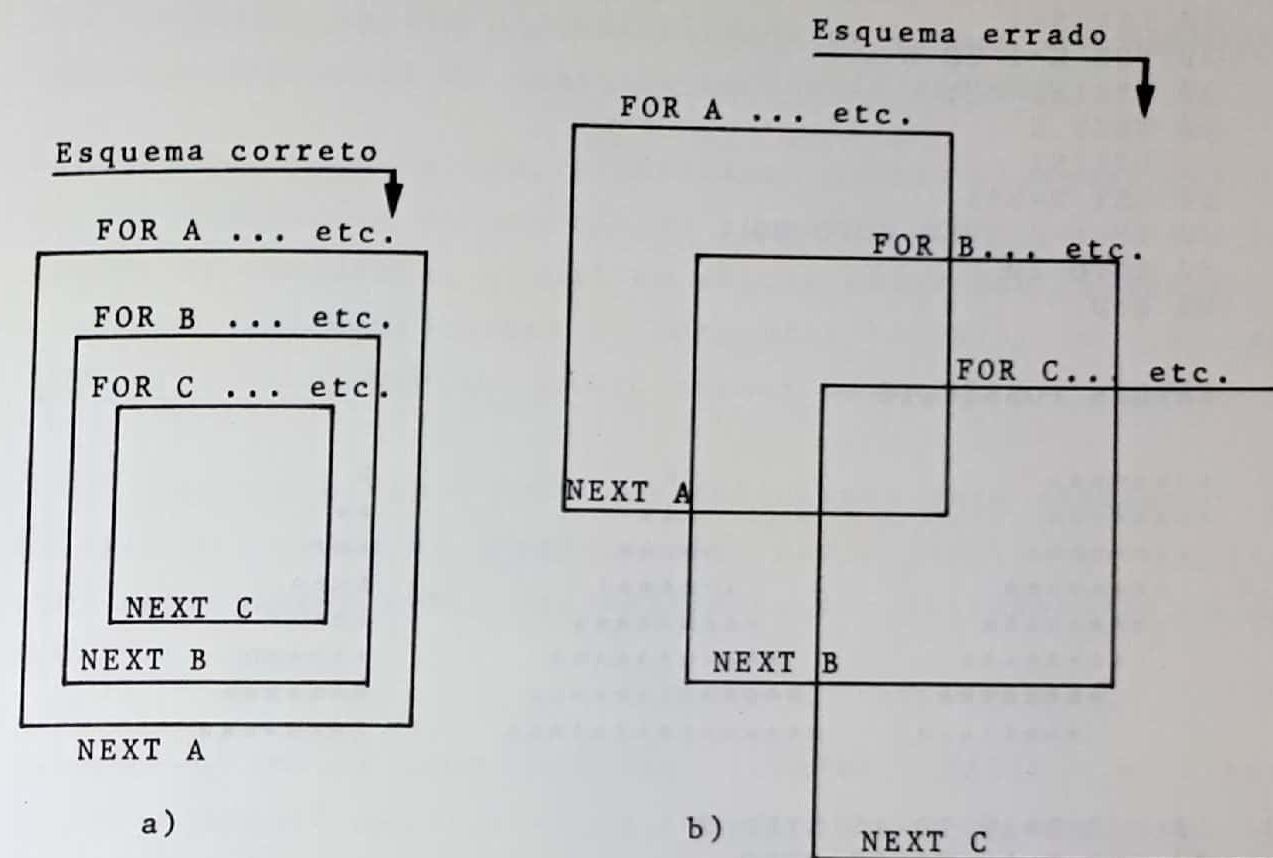


Figura 2.32 - FOR-NEXT aninhados

Trabalhinho para você ou melhor ajude o Juquinha M. Iolomole que esqueceu o seu micro no sítio e precisa na segunda-feira dizer para a sua professora Dona Benevolente o que representam os programas abaixo.

- 1)


```

10 FOR X=1 TO 3
20 FOR Y=1 TO 2*X
30 FOR Z=1 TO 2*(X+Y)
40 LPRINT "*";
50 NEXT Z
60 LPRINT
70 NEXT Y
80 LPRINT
90 NEXT X
100 END

```
- 2)


```

10 FOR X1=1 TO 3
20 FOR X2=5 TO 6
30 LPRINT X1,X2
40 NEXT X2
50 NEXT X1
60 END

```


114

3) 5 REM - QUE TIPO DE SAIDA PRODUZ O SEGUINTE PROGRAMA? -

```
10 LET N=1
20 FOR K=1 TO N
30 LPRINT "*";
40 NEXT K
50 LPRINT
60 LET N=N+1
70 IF N>8 THEN GOTO 90
80 GOTO 20
90 END
```

SAIDAS POSSÍVEIS

```

      *
    ***
  *****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****
*****

```

```
4) 10 FOR X=10 TO 15 STEP 2
    20 FOR Y=15 TO 10 STEP -2
    30 IF Y=X THEN 999
    40 IF X<Y
    50 PRINT Y
    60 NEXT Y
    70 NEXT X
    999 END
```

Observação importante:

Vamos falar um pouco mais sobre a formação de nomes de va
riáveis numéricas simples.

Regras fundamentais

- (1) O primeiro caractere tem que ser obrigatoriamente uma letra (maiúscula de A até Z).
- (2) Opcionalmente, o primeiro caractere pode ser seguido de um dígito decimal (0 a 9).
- (3) Caso o nome tenha dois caracteres não pode haver espaço em branco entre eles.

Considerando-se que são 26 letras do alfabeto e 10 os dígitos decimais, existe a possibilidade de termos 260 combinações na formação de nomes de *variáveis simples*.

Entretanto, hoje em dia, ainda temos muitos sistemas que devido a limitações de memória só permitem criar um número limitado de variáveis, o qual em alguns casos não ultrapassa a centena incluindo nomes de conjuntos (arrays) ou seja de variáveis coletivas que discutiremos adiante.

Esse limite todavia é mais que suficiente para atender à maioria dos casos e, praticamente, não é preciso ter qualquer preocupação com a possibilidade de ultrapassá-lo em al gum programa.

Isto é tão remoto como você não aprender o BASIC e estar "estraçalhando" daqui a algum tempo, sempre no bom sentido, o seu micro!!!

Exemplo 2.23

Elabore um gráfico para a curva "quadrática" $y=x^2$ locando ao menos 25 pontos ou seja para x variando de 1 até 25.

Solução:

Com certeza este gráfico não caberá na tela do vídeo porém sairá "lindinho" na folha impressa.

Um possível programa pode ser:

```
10 FOR X=1 TO 25
20 LPRINT TAB(X*X/10); "*"
30 NEXT X
40 END
RUN
```

Teremos a saída:

Dividindo por 10 nos locais o gráfico dentro da escala de 0 até 63 colunas. Talvez você tenha que dividir por 20. Depende do seu micro. Veja bem isto.

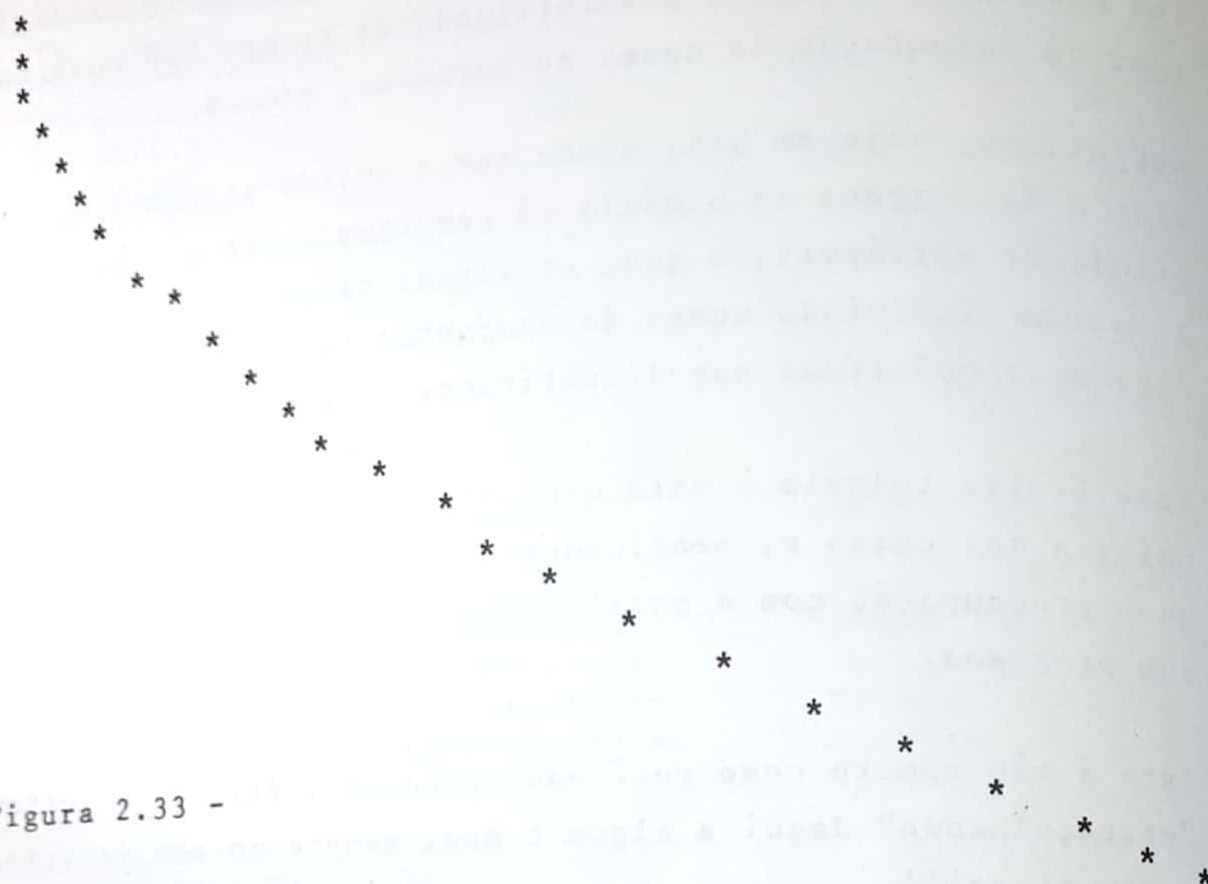


Figura 2.33 -

Como uma aplicação modifique o seu programa usando a curva cuja equação é $y=x^2$.

Convém mudar o fator de escala?

Considerações sobre o Exemplo 2.23.

Você lembra bem que TAB(X) significa que deve-se mover para a x-ésima posição antes de imprimir o asterisco sendo que as posições das colunas são numeradas com 0,1,2,3,...

Desta forma ao variarmos X de 1 até 25 nos variaremos $\frac{X^2}{10}$ de 0,1 até 62,5.

Desde que TAB usa apenas a parte inteira deste argumento, os asteriscos serão impressos em posições determinadas pelos números da 3ª coluna da Tabela 2.2.

Exemplo 2.24

Você entrou em uma "lanchonete" e vê a sua frente as possíveis escolhas (Tabela 2.3).

X	$\frac{X^2}{10}$	TAB(X*X/10)
1	0,1	0
2	0,4	0
3	0,9	0
4	1,6	1
5	2,5	2
6	3,6	3
7	4,9	4
8	6,4	6
...	...	...
25	62,5	62

Tabela 2.2

"SANDWICH"	CALORIAS
Hot-dog simples	100
Hot-dog incrementado	140
Hamburger sofisticado	190
Churrasco	210
Calabresa	240

Tabela 2.3

Além do preço de cada "sandwich" está indicada a quantidade de calorias que se "ganha" ao comer um.

Supondo que você só pode tomar decisões do tipo SIM ou NÃO para a compra de cada um desses "sandwiches" elabore a tabela completa com a correspondente quantidade em calorias que você adquirirá com a sua escolha.

Note bem que você pode não comprar nada e ganhar nesse caso 0 calorias (não gastando nada e provavelmente emagrecendo) ou até ser um tremendo glutão e querer experimentar um de cada tipo.

Aceite a imposição que é proibido "experimentar" mais de um do mesmo tipo com exceção de calabresa do qual pode-se comer até 2 e engordar bastante.

Solução:

```

10 LPRINT "      "; "HOT DOGS    HOT DOG INC    HAMBUR SOFI
   CHURRASCO    CALABRESA    CALORIAS
20 K=1
30 FOR H1=0 TO 1
40 FOR H2=0 TO 1
50 FOR H3=0 TO 1
60 FOR C1=0 TO 1
70 FOR C2=0 TO 2
80 LPRINT "#";K;": ";
90 LPRINT " ";H1;": ";H2;": ";H3;": ";C1;
100 LPRINT " ";C2;100*H1+140*H2+190*H3+210*C1+240*C2
110 K=K+1
120 NEXT C2

```



```

130 NEXT C1
140 NEXT H3
150 NEXT H2
160 NEXT H1
170 END

```

Ter-se-á a saída:

	HOT DOGS	HOT DOG INC	HAMBUR	SOFI	CHURRASCO	CALABRESA	CALORIAS
#1 :	0	0	0	0	0	1	240
#2 :	0	0	0	0	0	2	480
#3 :	0	0	0	1	0	0	210
#4 :	0	0	0	1	1	1	450
#5 :	0	0	0	1	2	2	690
#6 :	0	0	1	0	0	0	190
#7 :	0	0	1	0	0	1	430
#8 :	0	0	1	0	0	2	670
#9 :	0	0	1	1	0	0	400
#10 :	0	0	1	1	1	1	640
#11 :	0	0	1	1	1	2	880
#12 :	0	1	0	0	0	0	140
#13 :	0	1	0	0	0	1	380
#14 :	0	1	0	0	1	2	620
#15 :	0	1	0	1	0	0	350
#16 :	0	1	0	1	1	1	590
#17 :	0	1	0	1	1	2	830
#18 :	0	1	1	0	0	0	330
#19 :	0	1	1	0	0	1	570
#20 :	0	1	1	0	0	2	810
#21 :	0	1	1	1	0	0	540
#22 :	0	1	1	1	1	1	780
#23 :	0	1	1	1	1	2	1020
#24 :	1	0	1	0	0	0	100
#25 :	1	0	0	0	0	1	340
#26 :	1	0	0	0	0	2	580
#27 :	1	0	0	1	0	0	310
#28 :	1	0	0	1	1	1	550
#29 :	1	0	0	1	1	2	790
#30 :	1	0	1	0	0	0	290
#31 :	1	0	1	0	0	1	530
#32 :	1	0	1	0	0	2	770
#33 :	1	0	1	1	0	0	500
#34 :	1	0	1	1	1	1	740
#35 :	1	0	1	1	1	2	980
#36 :	1	1	0	0	0	0	240
#37 :	1	1	0	0	0	1	480
#38 :	1	1	0	0	0	2	720
#39 :	1	1	0	1	0	0	450
#40 :	1	1	0	1	1	1	690
#41 :	1	1	0	1	1	2	930
#42 :	1	1	1	0	0	0	430
#43 :	1	1	1	0	1	1	670
#44 :	1	1	1	0	1	2	910
#45 :	1	1	1	1	0	0	640
#46 :	1	1	1	1	1	1	880
#47 :	1	1	1	1	1	2	1120
#48 :	1	1	1	1	1	2	

"OUÇA-HE MUNDO!
O MEU PROGRAMA DEU
CERTO E FUNCIONA
MARAVILHOSAMENTE"



\$2.12 - AS FUNÇÕES INT E SGN

O que vem a ser um inteiro?

O que é uma função INTEGER?

Será que é um velho pesadelo oriundo da Álgebra?

Oh não, é apenas uma função extremamente útil e com `INT(X)` em BASIC podemos "arredondar" qualquer número seja *pequeno* ou *grande*, *positivo* ou *negativo* para um inteiro.

Cuidado, não estamos falando sobre o arredondamento comum ou ordinário.

No arredondamento comum pensa-se em obter o número inteiro mais próximo daquele que temos, seja ele grande ou pequeno.

Desta forma `INT(X)` nos dá o *maior número inteiro* que seja menor ou igual ao número `X`.

Em se tratando de um número `X` positivo, esta função simplesmente "trunca" as suas decimais e no caso de um número negativo ela toma o número negativo imediatamente menor que `X`.

Exemplo 2.25

```

10 LPRINT "X","INT(X)","X/3","INT(100*X/3)"
20 FOR X=-2 TO 2 STEP 0.5
30 LPRINT X,INT(X),X/3,INT(100*X/3)
40 NEXT X
50 END
RUN

```

Deve-se ter a seguinte saída:

X	INT(X)	X/3	INT(100*X/3)
-2	-2	-.666667	-67
-1.5	-2	-.5	-50
-1	-1	-.333333	-34
-.5	-1	-.166667	-17
0	0	0	0
.5	0	.166667	16
1	1	.333333	33
1.5	1	.5	50
2	2	.666667	66



Figura 2.35 - "Eu penso que vou chamá-lo de Inteiro"

Vejamos agora mais um exemplo envolvendo a função INT.

Exemplo 2.26

```
30 LET X=3.14159
40 LET Y=INT(X)
50 LET Z=X-Y
60 LPRINT "X=";X
70 LPRINT "Y=";Y
80 LPRINT "Z=";Z
90 END
RUN
```

Este é o nosso
"manjado" $\pi(\pi)$



Aha, Aaha, aha, você não sabe ainda o que vai descobrir com este programa mas deve ser otimista...

Na saída ter-se-á:

```
X = 3.14159
Y = 3
Z = .14159
```

E com este programa "cortamos" ou separamos o número X em sua parte inteira que chamamos de Y e na sua parte decimal que chamamos de Z.



Figura 2.36 - "Aplicando o machado marca INT".

Em algumas situações é muito importante saber se uma variável é positiva ou negativa.

A função SGN tem apenas três valores possíveis

$$\begin{cases} \text{SGN}(X)=1 & \text{se } X>0 \\ \text{SGN}(X)=0 & \text{se } X=0 \\ \text{SGN}(X)=-1 & \text{se } X<0 \end{cases}$$

Exemplo 2.27

Este programa acha o SGN (signum) dos números de 4,8 até -4,8 decrementados em cada passagem de 1,2.

```
10 FOR N=4.8 TO -4.8 STEP -1.2
20 A=SGN(N)
30 LPRINT "SGN(";N;")=";A
40 NEXT N
50 END
RUN
```

Deve-se ter a seguinte saída:

```
SGN( 4.8 )= 1
SGN( 3.6 )= 1
SGN( 2.4 )= 1
SGN( 1.2 )= 1
SGN( 0 )= 0
SGN(-1.2 )=-1
SGN(-2.4 )=-1
SGN(-3.6 )=-1
SGN(-4.8 )=-1
```


Bom, muito bom mesmo caro leitor ou leitora e agora você deve parar um pouco antes de prosseguir na leitura e escutar um pouco de boa música



§2.13 - MAIS INSTRUÇÕES DE DESVIO GOSUB-RETURN, ON-GOSUB, ON-GOTO

Esperando que tenha aproveitado o intervalo concedido com a música de acordo com a nossa última recomendação e que esteja bastante tranquilo (a) para entender bem depressa o que vem a ser as instruções:

- 1) GOSUB - RETURN
- 2) ON - GOSUB
- 3) ON - GOTO

Começemos com a pergunta:

O que é uma subrotina?

É muito gozado, foi você quem perguntou ou fiz a mim mesmo a pergunta.

Não importa, vamos a resposta.

Uma subrotina é um programa, em geral curto porém é um programa bem especializado (ou rotina) que você irá embutir em um programa muito maior para uma finalidade especial.

O BASIC armazena muitas subrotinas em um lugar especial na memória e elas podem ser chamadas por um simples conjunto de letras

Instrução GOSUB #####

Aqui #####
representa o número
de uma linha

Esta instrução determina que o programa deve pular para a subrotina que começa na linha de número ##### (também a linha não precisa ter tantos números não é?).

Uma subrotina tem que ter obviamente um fim pois caso contrário seria impossível separar a subrotina do resto do programa.

As subrotinas em BASIC, terminam obrigatoriamente com a instrução RETURN. Não confunda entretanto com a tecla RETURN do MAXXI.

Isto quer dizer que após a execução da subrotina o programa retornará (RETURN) ao ponto onde estava o programa ou evitando a confusão, o programa retorna para a linha seguinte àquela em que estava a instrução GOSUB.

Exemplo 2.28

Como uma aplicação de subrotina vejamos como o Alexandre, o Sérgio e o Victor Junior podem calcular os seus rendimentos nas respectivas cadernetas de poupança depois de um ano de 365 dias de capitalização se a mesma fosse feita diariamente.

Esta é obviamente uma situação utópica...

```

10 LPRINT "ENTRE COM A TAXA DE JUROS POR DIA"
20 INPUT J
30 LPRINT "ENTRE COM O VALOR DA CADERNETA DO ALEXANDRE"
40 INPUT B
50 GOSUB 1000
55 LPRINT
60 LPRINT "ENTRE COM O VALOR DA CADERNETA DO SÉRGIO"
70 INPUT B
80 GOSUB 1000
85 LPRINT
90 LPRINT "ENTRE COM O VALOR DA CADERNETA DO VICTOR JR."
100 INPUT B
110 GOSUB 1000
120 STOP
1000 REM SUB-ROTINA PARA OBTER OS JUROS PAGOS
1010 REM E O VALOR DA APLICAÇÃO APÓS 1 ANO.
1020 LET N=B
1030 FOR D=1 TO 365
1040 LET N=N+(J/365)*N

```



```

1050 NEXT D
1060 LET I=N-B
1070 LPRINT "VALOR DOS JUROS PAGOS AO ANO=";I;"VALOR ATUAL
    NA CADERNETA =";N
1080 RETURN

```

Dependendo dos particulares valores com os quais você entrar via instrução INPUT ter-se-á um resultado.

Uma possibilidade está listada abaixo:

```

ENTRE COM A TAXA DE JUROS POR DIA ? 532
ENTRE COM O VALOR DA CADERNETA DO ALEXANDRE ? 180000
VALOR DOS JUROS PAGOS AO ANO= 126301 VALOR ATUAL NA CADERNETA = 306301
ENTRE COM O VALOR DA CADERNETA DO SEFGIO ? 220000
VALOR DOS JUROS PAGOS AO ANO= 154368 VALOR ATUAL NA CADERNETA = 374368
ENTRE COM O VALOR DA CADERNETA DO VICTOR JR. ? 400000
VALOR DOS JUROS PAGOS AO ANO= 280670 VALOR ATUAL NA CADERNETA = 680670

```

Que inflação hein! Será que chegaremos lá ou já estamos na própria?

Observação importante:

Não é bem assim que se calculam os juros na nossa caderneta, porém de uma outra forma, parecida com a que acabamos de explicar.

Peça auxílio ao seu pai para mudar a subrotina para que o cálculo de juros seja real.

Ele não sabe isto?

Hum, hum, hum...



Figura 2.37

Passemos agora ao ON-GOSUB.

O que é então ON < expressão aritmética > GOSUB [< número de linha >, < número de linha > ...] ?

Resposta

É a forma geral da instrução de desvio incondicional múltiplo para a subrotina.

A instrução ON-GOSUB não é disponível em alguns micros ou seja os pequenos.

O valor da expressão aritmética, depois de calculado, é truncado, isto é, toma-se o seu valor inteiro.

Se este valor é menor do que 1 ou maior do que o número de linhas da lista à direita, dependendo do sistema BASIC poderá ocorrer uma mensagem de erro ou simplesmente a transferência da execução para a instrução que venha imediatamente após.

Comumente a execução continuará na linha para a qual foi feito o desvio.

O mesmo número de linha poderá aparecer diversas vezes na lista à direita.

Exemplo 2.29

Se tivermos uma instrução como

```
70 ON X GOSUB 300, 400, 500
```

o que acontecerá?

Resposta:

Ocorrerá um desvio incondicional para uma das subrotinas que se iniciam por um dos números de linha que compõem a lista, de acordo com o valor do índice, isto é, o valor inteiro da variável numérica X:

se X=1, desvio para a linha 300

se X=2, desvio para a linha 400

se X=3, desvio para a linha 500

Exemplo 2.30

Se tivermos uma instrução como

```
80 ON 4*N-13 GOSUB 450, 770
```

o que ocorrerá?

Resposta:

Acontecerá um desvio incondicional para uma das subrotinas, que se iniciam por um dos números de linha que compõem a lista, de acordo com o valor do índice, isto é, o valor inteiro da expressão:

- se a expressão = 1, desvio para a linha 450
- se a expressão = 2, desvio para a linha 770

Finalmente a pergunta o que é a instrução ON-GOTO?

Resposta:

Esta instrução *não é*, na realidade, uma instrução de subrotina.

Ela apresenta entretanto alguma semelhança com a instrução ON-GOSUB.

A instrução GOTO, como já vimos anteriormente permite ao programa saltar para uma *única e especificada linha*.

A instrução ON-GOTO permite saltar ou se desviar para *uma* de várias linhas, dependendo do valor da variável numérica que se encontra neste instante na instrução.

Assim a forma geral da instrução ON-GOTO é:

ON <variável> GOTO <lista de números de linha>

Exemplo 2.31

O que ocorrerá se tivermos a instrução

```
60 ON Y1 GOTO 110, 130, 150, 170 ?
```

Resposta:

Se Y1=1 o programa salta para a linha 110; se Y1=2 ele se desvia para a linha 130, se Y1=3 ele salta para a linha 150 e finalmente se Y1=4 o programa se desvia para a linha 170.

Deve-se notar e não esquecer que Y1 não pode ser menor que (1)um ou maior que o número de números de linha na lista.

Como se vê o funcionamento é muito semelhante ao já explicado para o GOSUB, com a única e importante diferença que, sendo um GOTO, não há retorno e portanto não existe a instrução RETURN.

Adiante veremos alguns programas usando ON-GOTO e ON-GOSUB porém que tal você quebrar um pouco a "cuca" e fazer os seguintes programas:

TRABALHO PARA VOCÊ!!!

- 1) Para treinar bem a instrução GOSUB que terminará sempre com um RETURN faça o programa para calcular as raízes de uma equação do 2º grau dando condições de imprimir todos os casos inclusive o caso de raízes complexas.

Pede-se também que se faça inicialmente um teste para ver se não é proposta uma expressão do 1º grau.

Observações: a) Se você ainda não aprendeu o que é uma equação do 2º grau, peça ao seu progenitor para recordar um pouco de Álgebra e lhe explicar a fórmula de Baskara.

b) Seria muito bom que o inventivo leitor ou leitora tentasse fazer o programa sem o uso de subrotina e depois com a subrotina.

- 2) Sabendo também que você já foi ou ainda é como é o Sérgio ou seja gosta de frutas frescas, elabore mesmo assim um programa para fazer sugestões para um cardápio de frutas como sobremesa do seu lauto almoço.

Uma vez recebida a informação do seu "gastrônomo" micro envie a mesma para a sua querida progenitora.

Neste ponto é proibido proferir impropérios...

Como sugestões do cardápio inclua, morango, uva, banana, melancia, abacaxi, pêssago, maçã, ameixa e até jaca.

É útil ressaltar que o micro fará para você uma escolha aleatória para evitar a "rotina" e portanto descanse um pouco...

O autor do livro quis dizer protele um pouco a tentativa de resolver este problema até aprender o que é a função RND e principalmente o que é uma variável subscrita.

Enquanto isto dê uma "regada" nos seus já muitos conhecimentos do BASIC...



Figura 2.38

§2.14 - O QUE SÃO NÚMEROS ALEATÓRIOS?

Definição de número aleatório

Número aleatório é aquele cujo valor é imprevisível.

Hum! Hum! Hum!

Realmente esta não é uma boa definição porém vamos melhorá-la dizendo que números aleatórios são aqueles comandados pela lei das probabilidades ou seja são aqueles obtidos através do acaso e uma das formas de entender a aleatoriedade é pensar em um sorteio.



Figura 2.39 - "Um número aleatório é aquele que tem a mesma probabilidade que qualquer outro de ser escolhido".

No BASIC temos um gerador de números aleatórios que é a função RND (do Inglês Random que significa aleatório ou casual).

Definição

RND(n) é uma função que dá um número aleatório.

Caso se tenha a instrução

```
10 LET X = RND(0)
```

será atribuído a variável X um valor aleatório com seis casas decimais tal que $0 \leq X < 1$.

Já com a instrução

```
10 LET X = RND(N)
```

onde N é um inteiro, se atribuirá a X um inteiro aleatório entre 1 e N inclusive.

Geralmente N pode ser um inteiro entre 2 e 32767.

Dentro deste contexto RND(1) não tem muita utilidade ou melhor não faz sentido pois fará voltar sempre o valor 1.

Observação importantíssima:

A definição que demos para a função RND não é geral e vale a pena testar a mesma no seu micro.

Há micros nos quais só se escreve RND sem o argumento para obter o que descrevemos com RND(0) e aí então para se obter um valor inteiro multiplica-se RND pelo valor desejado acrescido de uma unidade e toma-se ou aplica-se ao resultado a função INT.

Em outros obtém-se com RND(1) o que descrevemos com RND(0).

Dentro do contexto da nossa definição pode-se querer a obtenção de um número aleatório entre dois valores que não sejam 1 e N.

Assim se tivermos uma instrução como:

```
10 LET X = RND(B-A+1) + A-1
```

isto significa que teremos números inteiros entre A e B se $B > A$.

De uma forma mais simples a instrução

```
10 LET X = RND(26)+74
```

atribuirá a X um número inteiro entre 75 e 100 incluindo os extremos.

Virtualmente todos os jogos feitos no micro são baseados na função RND(0) ou em RND(N) e logo, logo, você estará jogando e quem sabe programando algum.

Os números aleatórios são fundamentais para se fazerem simulações e precisaremos de situações simuladas não apenas nos jogos como também em muitos problemas técnicos e científicos.

É possível gerar mais do que um número aleatório usando mais que um gerador no programa.

Isto tem um valor especial quando se necessita de números aleatórios em amplitudes diferentes e também não deixa de ser importante quando se precisa de números aleatórios mesmo que seja dentro do mesmo campo ou intervalo.

Exemplo 2.32

Simulemos um trecho do jogo do "craps" no qual se jogam dois dados de seis faces e é importante dentro das regras do jogo a soma dos pontos.

```
40 LET A=RND(6)
50 LET B=RND(6)
70 LET X=A+B
80 PRINT X
RUN
```

Qual é o valor de X que sai?

Não sai nada!!!

Não é possível ou melhor veja o manual do seu micro para as eventuais modificações na função RND.

Se o leitor(a) anotar a série de valores gerados pela função RND perceberá em cada nova seção em que o programa for executado que eles se repetem.

Pelo fato de se repetirem a cada nova execução do programa costuma-se chamar aos mesmos de pseudo-aleatórios.

Pode-se desejar que a repetição não ocorra, e para tanto existe a instrução RANDOMIZE que altera a raiz do gerador de números aleatórios.

A instrução RANDOMIZE deverá ser executada antes que a função RND seja executada pela primeira vez.

Alguns sistemas BASIC usam para isto a palavra RANDOM e outros RAND.

Conclusão:

A forma geral da instrução de alteração "randômica" da raiz da função RND é dada por uma das palavras do conjunto

{RANDOMIZE, RANDOM, RAND}

Como é no seu micro?

Exemplo 2.33

Este programa simula o percurso aleatório feito por um rato no labirinto da Figura 2.39.

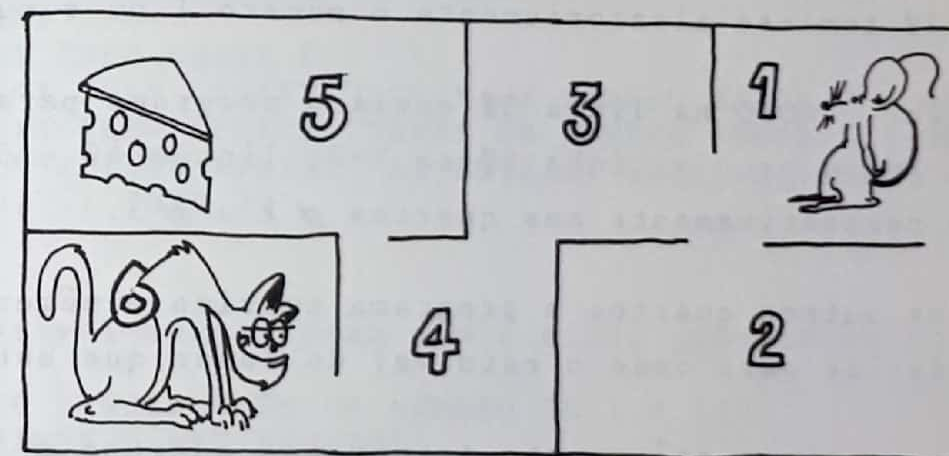


Figura 2.40

Uma vez que o rato entrar no labirinto a porta é fechada.

O rato anda nele até que ache o queijo ou dê de "cara" com o gato.

```
10 LPRINT 1; " ";
20 LET X=RND(2)
30 IF X=1 THEN GOTO 40
35 LPRINT 3; " ";
37 GOTO 80
40 LPRINT 2; " ";
50 LET X=RND(2)
60 IF X=1 THEN GOTO 10
70 LPRINT 3; " ";
80 LET X=RND(3)
90 IF X=2 THEN GOTO 10
100 IF X=3 THEN GOTO 40
105 LPRINT 4; " ";
110 LET X=RND(2)
120 IF X=1 THEN GOTO 140
130 LPRINT 5; "O RATO VAI SE EMPANTURRAR DE QUEIJO"
138 GOTO 199
140 LPRINT 6; "CHEGOU O ALMOÇO DO GATO"
199 END
```


Você "brincando" algum tempo com o seu micro pode ter as seguintes saídas:

```

1 3 4 6 CHEGOU O ALMOÇO DO GATO
1 3 4 5 O RATO VAI SE EMPANTURRAR DE QUEIJO
1 2 1 2 1 3 4 5 O RATO VAI SE EMPANTURRAR DE QUEIJO
1 3 2 1 3 4 5 O RATO VAI SE EMPANTURRAR DE QUEIJO
1 2 3 4 6 CHEGOU O ALMOÇO DO GATO

```

Comentários sobre o programa do Exemplo 2.33.

Na linha 10 o programa imprime 1 para indicar que está no quarto ou cela # 1.

Na linha 20 toma-se aleatoriamente o número 1 ou o número 2.

A instrução GOTO na linha 30 envia o programa para a linha 40 se X=1 e para a linha 50 se X=2; linhas 40 e 50 correspondem respectivamente aos quartos # 2 e # 3.

Uma vez nos outros quartos o programa imprime o número do quarto e decide para onde o rato vai do lugar que está.

A execução continua até o rato entrar nos quartos # 5 ou # 6.

O fim, neste caso o quarto # 6, é evidente!

TRABALHO DE APLICAÇÃO E AO MESMO TEMPO DISTRAÇÃO PARA VOCÊ

1) No estilo do Exemplo 2.33 temos um "labirinto" e um "ladrão" que entrou pela porta #1 podendo escapar através da porta #7 ou encontrar o "super-guarda" João Valentão no quarto #8 e ser preso (Figura 2.41).

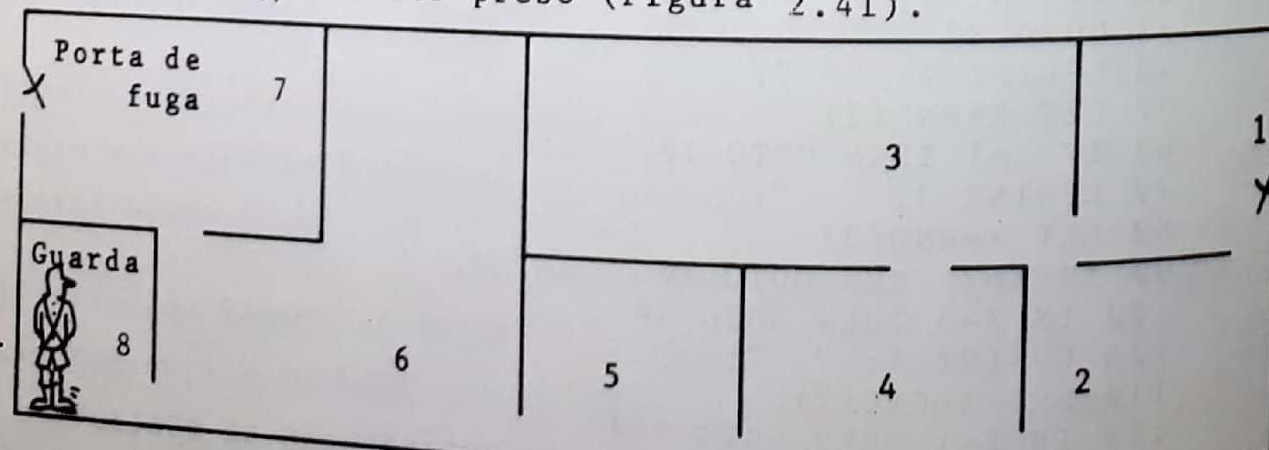


Figura 2.41

Faça um programa para ver se é possível prender o ladrão.

Exemplo 2.34

Uma outra aplicação da função RND pode ser aquela de tentar adivinhar um número inteiro até 133 em até cinco tentativas com a condição que o micro lhe indique se você está perto ou longe ou ainda se está "quente" ou "frio".

```

10 REM *** ADIVINHE O SEU NÚMERO - UM JOGO COM O COMPUTADOR***
20 LET X=INT(133*RND(0))+1
30 LPRINT
40 LPRINT "ESTOU PENSANDO EM UM NÚMERO DE 1 A 133."
50 LPRINT "ADIVINHE O MEU NÚMERO!!"
60 LPRINT
70 INPUT "MEU CHUTE É'";N
80 IF N<X THEN LPRINT "TENTE UM NÚMERO MAIOR." : GOTO 70
90 IF N>X THEN LPRINT "TENTE UM NÚMERO MENOR." : GOTO 70
100 IF N=X THEN LPRINT "AI ESTÁ!!! VOCÊ ADIVINHOU O MEU NÚ-
    MERO." : GOTO 20
RUN

```

Uma possível conversação com o micro poderia ser:

ESTOU PENSANDO EM UM NÚMERO DE 1 A 133.
ADIVINHE O MEU NÚMERO!!

TENTE UM NÚMERO MAIOR.
TENTE UM NÚMERO MENOR.
TENTE UM NÚMERO MENOR.
AI ESTÁ!!! VOCÊ ADIVINHOU O MEU NÚMERO.

Nesta saída nós não temos indicado o valor que entra cada vez que se passa pela instrução INPUT (linha 70) porém ele aparece na tela do vídeo caso o seu micro aceite a instrução como está escrita.

Por outro lado dissemos no enunciado que o "joguinho" não poderia ultrapassar 5 tentativas, mas esta restrição não apareceu no programa apresentado e não há nenhuma mensagem de que você está demorando muito para acertar.

Faça isto como exercício

Será que você consegue?

Neste programa (o do Exemplo 2.34) apareceu pela primeira vez uma instrução múltipla.

Muitos micros hoje já permitem que uma série de instruções entre em uma linha do programa se elas aparecerem separadas umas das outras por dois pontos (:).

Linhas com instruções múltiplas tem um significado todo especial quando forem usadas com o IF-THEN.

Já sabemos que na sua forma mais simples a instrução IF-THEN é dada como:

IF < condição > THEN < número de linha >

No TK-85 por exemplo não basta escrever o número de linha mas sim deve-se escrever GOTO< número de linha>.

Porém já existem micros que permitem um uso mais flexível do IF-THEN ou seja no lugar do número de linha pode aparecer uma outra instrução.

Isto significa que a instrução depois da condição *será executada* somente se a condição *for verdadeira*.

A forma é então:

IF < condição > THEN < instrução >

Exemplo 2.35

É muito comum precisarmos de instruções do tipo

- a) IF X < > 0 THEN PRINT X
- b) IF Z > 13 THEN PRINT "PERDEMOS O NÚMERO DA SORTE"
- c) IF R <= 0 THEN R=X+4.



Note que nem o LET não apareceu. Aliás muitos micros hoje já aceitam a atribuição sem o LET

Quando se tem uma linha com instruções múltiplas as que seguem THEN serão executadas, se a condição seguindo o IF, *for verdadeira*.

Nenhuma das instruções seguindo o THEN será executada quando a *condição é falsa*.

Assim por exemplo no nosso programa de adivinhar o número que o *micro escolheu* (Exemplo 2.34) ele (micro) assim entende a linha 90.

VERDADEIRA
90 IF N>X THEN PRINT "TENTE UM NÚMERO MENOR":GOTO 70

↕ FALSA
(segue o programa)

Ou ainda repisando o mesmo assunto, se tivermos:

- a) N=95 e X=71 então a condição é verdadeira e o micro irá executar o PRINT e o GOTO;
- b) N=70 e X=71 então a condição é falsa e o micro vai para a linha número 100 (veja o Exemplo 2.34).
- c) N=71 e X=71 então a condição da instrução de número de linha 90 é ainda falsa e o micro vai para a linha 100 para a qual a condição é agora verdadeira e então é executado o PRINT e o GOTO.

Acho que você já pegou como funciona uma linha com instruções múltiplas tendo no início o IF-THEN.

Um último adendo para o IF-THEN.

Gostou do adendo. Hum, mas não fique tão obnubilado.

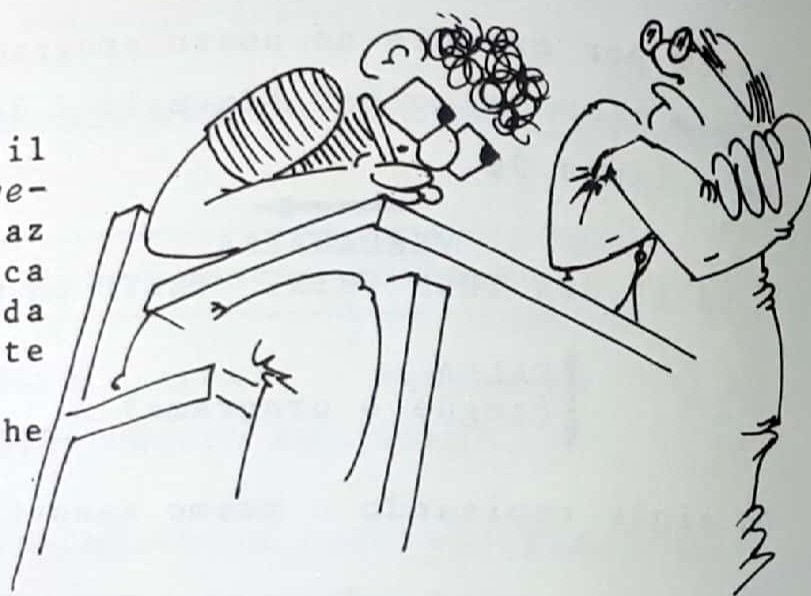
A instrução IF-THEN pode ser ampliada com o uso do modificador ELSE (algo mais) para incluir uma instrução alternativa se a condição do IF-THEN é falsa.

Quando a condição IF é verdadeira, a instrução que segue o THEN é executada, caso contrário a instrução seguindo o ELSE é executada.

A forma geral do IF-THEN-ELSE é então:

```
IF <condição> THEN <instrução> ELSE <instrução>
```

Figura 2.42 - "É muito fácil entender o que é um número aleatório quando se faz uma prova de Matemática da qual não se sabe nada e a mesma é do tipo teste de múltipla escolha. A resposta que se escolhe é um número aleatório".



Exemplo 2.36

Divirta-se com este programa e com os seus amigos para ver quem é o mais rápido na reação.

Quando o "computador escrever "PRONTO!?!?!!" você deve apertar a tecla "BREAK" para pará-lo.

Aquele "jogador" ou seja participante da brincadeira que conseguir isto em menor tempo ganha.

Aquele que demorar mais para parar o programa como castigo deve levar o seu cachorrão para "regar" os postes ou então tirar e limpar toda a mesa para a mãe e o que é pior lavar toda a louça...

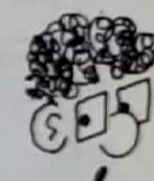
Este programa como você pode ver serve para medir a reação de um "bebum" ou seja quem está com o nível etílico elevado vai quase sempre titubear e será "reprovado" por ser lento neste "joguinho".

Ai, ai, ai!!! o micro pode substituir até o "bafômetro"! Oh! maravilhoso micro da liga anti-alcóolica.

Programa solução

```
10 LPRINT"ESTEJA ATENTO..."
20 FORA=1TO400:NEXTA
30 LPRINT:LPRINT:LPRINT:LPRINT
40 LPRINTTAB(20),"PREPARE-SE..."
50 X=RND(1300)
60 FORN=1TOX:NEXTN
70 CLS
80 LPRINT:LPRINT:LPRINT:LPRINT
90 LPRINTTAB(20),"PRONTO!?!?!!"
100 FORZ=1TO13
110 LPRINTZ
120 NEXTZ
130 LPRINT:LPRINT:LPRINT
140 LPRINT"ALÉM DE SER LERDO NAS REAÇÕES VOCÊ TENDE A PANACA"
150 FORK=1TO1000:NEXTK
```

Se você bater o seu programa assim juntinho o seu micro entende?



Se um dia você estiver "bebum" o seu micro lhe dará a seguinte saída com uma crítica branda.

ESTEJA ATENTO...

PREPARE-SE...

PRONTO!?!?!!

1
2
3
4
5
6
7
8
9
10
11
12
13

ALÉM DE SER LERDO NAS REAÇÕES VOCÊ TENDE A PANACA

E com isto acabou o capítulo 2.

A instrução DATA é uma instrução de especificação isto é, não executável.

Se é possível declarar os valores dados para serem usados em cálculos no programa, nada mais normal do que ter também uma instrução para fazer a "aquisição" desses dados, nos pontos do programa onde se fizer necessário.

A instrução READ não é uma instrução de especificação mas sim uma instrução executável.

Pode-se traduzir essa instrução pela palavra "ler".

Dessa maneira os números que o programa usa (colocados na forma decimal ou científica) são postos na instrução DATA separados entre si por vírgulas e então "adquiridos" na ordem em que aparecem, através da instrução READ.



Figura 3.1 - "É o Sr. Read procurando a sua Srta. Data."

A posição da instrução DATA é totalmente irrelevante isto é, ela pode estar em qualquer posição dentro do programa porém é uma boa idéia colocar os dados ou no topo do programa ou no fim do mesmo e antes da instrução END.

Além disso os dados podem ser declarados em mais de uma instrução READ.

Hi, isto parece agora um costume árabe que permite que um homem tenha mais de uma mulher...

Não, não, não esta analogia é horrível e pode criar problemas familiares com certeza.

Mas mesmo assim não esqueça que para um READ podemos ter várias linhas de DATA.

A ordem de aquisição dos dados obedece a ordem numérica crescente das instruções DATA.

Exemplo 3.1

Digamos que se queira obter uma série de produtos entre os números A e B e imprimir o resultado.

O programa poderia ser o seguinte:

```
10 READ A,B
20 LPRINT A,B,A*B
30 GOTO 10
40 DATA 44.3,32.5,66.4,88.3,13.3,14.4
50 DATA 18.7,14.3,17.3,14.9
60 END
RUN
```

Teríamos uma saída do tipo:

44.3	32.5	1439.75
66.4	88.3	5863.12
13.3	14.4	191.52
18.7	14.3	267.41
17.3	14.9	257.77

Em alguns micros que aceitam o par READ-DATA sai geralmente uma mensagem do tipo:

OUT OF DATA IN 10

Hum em inglês você não entende, pois então pegue o dicionário.

Exemplo 3.2

Vejam os como se pode testar conjuntos de valores inteiros e positivos A, B e C para saber qual é o maior de cada trinca. Deve-se supor que cada conjunto é composto de três valores diferentes e que nunca aparece um número maior que 1000.

Solução:

```

10 DATA 18,17,19,15,14,13,7,29,18,1000,0,0
20 READ A,B,C
30 IF A=1000 THEN 140
40 IF A>B THEN 100
50 IF B>C THEN 80
60 LPRINT "C É O MAIOR"
70 GOTO 20
80 LPRINT "B É O MAIOR"
90 GOTO 20
100 IF A>C THEN 120
110 GOTO 60
120 LPRINT "A É O MAIOR"
130 GOTO 20
140 END
RUN

```

A saída para esses dados será:

```

C É O MAIOR
A É O MAIOR
B É O MAIOR

```

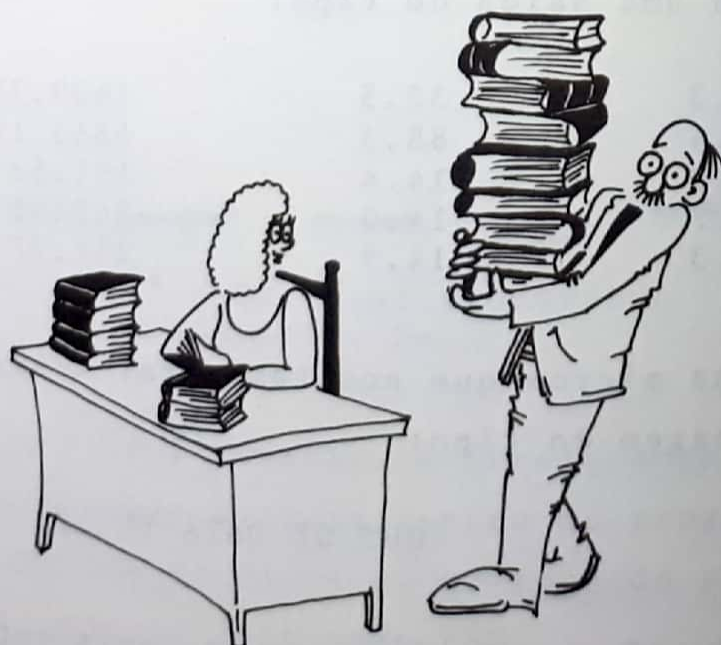


Figura 3.2 - "Comumente a gente vem à biblioteca para ler (READ) ou então para ler e procurar dados (DATA) para algum trabalho".

Observação importante:

Como se vê os valores 1000, 0, 0 que aparecem na instrução DATA são *auxiliares* ou *artificiais* de acordo com o enunciado e são usados para o importante teste que aparece na linha 30 e faz com que o programa pare automaticamente.

TRABALHO PARA VOCE

- 1) Que tal agora brilhante leitor ou leitora escrever um programa que leia quatro valores da instrução DATA e ache o menor.
O programa deve imprimir o menor valor e impõe-se que os quatro valores sejam todos diferentes entre si.
- 2) Refaça o programa do problema 1 deste último "trabalhinho" de tal forma que saia impresso dos quatro valores quem é o menor: primeiro, segundo, terceiro ou quarto.
- 3) Uma conhecida livraria, da qual o autor gosta muito e que está no começo da Rua Maria Antonia em São Paulo calcula o ganho mensal de seus vendedores na seguinte base: o ganho mensal é 18% do total de vendas, mais um "bonus" de 15% sobre qualquer quantidade vendida acima de 6000 u.m. Há 6 vendedores na firma e no mês de maio suas vendas foram as seguintes:

Vendedor	Venda mensal (u.m.=unidades monetárias)
1	8200
2	4800
3	6800
4	7000
5	5560
6	5900

Tabela 3.1

Elabore um programa que permita obter o ganho mensal de cada um.

- 4) Será que você consegue elaborar o desenho indicado na Figura 3.3?
- Utilize obrigatoriamente as instruções, READ-DATA, IF-THEN, GOTO e LPRINT.

```

*****
* * * * *
* * * * *
***   ***   ***   ***
*****   ***   *****
***   ***   ***   ***
*****   ***   *****
***   ***   ***   ***
*****   ***   *****
***   ***   ***   ***
***   ***   ***   ***
***   ***   ***   ***
***   ***   ***   ***
*****   ***   *****
***   ***   ***   ***
*****   ***   *****
***   ***   ***   ***
*****   ***   *****
* * * * *
* * * * *
*****

```

Figura 3.3

Exceções, exceções!!!

Às vezes é necessário ler o mesmo conjunto de dados mais do que uma vez.

É possível fazer isto através da instrução RESTORE (restaurar).

Isto parece até o Richard Burton que "restaurou" mais de uma vez os seus vínculos legais com a bela "diva" Elizabeth Taylor.

Mas isto é livro de computação ou sobre o cinema com astros "coroas"?

Resposta:

É de computação e por isso vejamos o que ocorre com o programa quando o mesmo chegar a uma instrução RESTORE.

Sempre que um programa chegar a uma instrução RESTORE, todas as linhas DATA são restauradas para a sua condição original de "não lidas ainda" tanto aquelas que já foram lidas como aquelas que eventualmente não foram e todas agora são novamente disponíveis para leitura outra vez começando com o 1º elemento da primeira linha DATA.

Será que a analogia do Richard Burton com a Elizabeth Taylor ajuda a entender melhor o READ-DATA junto com o RESTORE. Ajudou é!!!

Você é "entendidinho" de casamento e de divórcio...

Alguns micros não dispõem ainda das instruções READ, DATA e RESTORE o que é uma grande pena pois não se poderá desta forma entender nunca o "bom negócio" que fizeram Burton e Taylor.

Exemplo 3.3

Uma forma melhor de entender o que faz a instrução RESTORE é analisando o resultado do programa abaixo:

```

10 READ A
15 READ B
19 READ C
20 D=A+B*C
30 IF D=10 THEN RESTORE:READ D
40 LPRINT D,
50 IF D=14 THEN END
60 GOTO 10
70 DATA 4,2,3,6,0,2,7
80 END
RUN

```

Ter-se-á seguinte saída:

4

20

14

Você verá os resultados no seu micro porém a ordem das linhas segundo a qual o programa é executado é:

```
<10> <15> <19> <20> <30> <40> <50> <60> <10> <15>
<19> <20> <30> <40> <50> <60> <10> <15> <19> <20>
<30> <40> .
```

Você entendeu isto?

Pense um pouco mais e faça as contas para obter $D=A+B*C$ em cada volta e acho que aí você vai entender o RESTORE.

Note que pela 1ª vez não colocamos uma instrução de atribuição sem o LET o que aliás é possível hoje em dia em muitos micros.

Isto gera uma economia de tempo entre outras coisas.

Iremos alertar sobre este fato em alguns programas e depois deixaremos o cuidado por conta do freguês.

As coisas vão ficar um pouco mais complexas pois iremos nos envolver daqui a pouco com variáveis alfanuméricas, com variáveis subscritas, funções trigonométricas, matrizes, gráficos não tão simples, etc.

Os resultados porém serão compensadores.

Acredite "bicho"!!!



Figura 3.4

§3.2 - VARIÁVEL ALFANUMÉRICA OU STRING

Como já sabemos uma variável é utilizada para armazenar ou guardar uma informação que pode se alterar no decorrer do programa.

Quando a informação é um número, a variável que a representa é do tipo numérico e como já sabemos é comumente indicada por uma única letra ou uma mistura de uma letra e um número começando sempre por letra.

Hoje já temos micros que aceitam variáveis numéricas representadas por duas letras.

Por outro lado se a informação a ser armazenada não é um valor numérico, então a variável que a representa é denominada *alfanumérica* e é diferenciada das numéricas por ser seguida de um cifrão (\$).

Assim A\$ e Y\$ indicam que A e Y são variáveis alfanuméricas ou usando o nome em inglês variáveis *STRING*.

O *STRING* é um conjunto de letras e números que o micro manipulará tal como foi escrito no programa.

Portanto as variáveis alfanuméricas ou *STRING* representam uma fileira de caracteres que podem ser letras, palavras e/ou combinações de letras, números e espaços.

Temos assim três formas para imprimir palavras.

A primeira que já aprendemos há muito tempo é a de embutir ou encaixar palavras ou pequenas frases nas instruções PRINT ou LPRINT.

A isto se chama "imprimir um *STRING*".

A segunda é produzir palavras através da instrução INPUT.

Chama-se a isto "introduzir um *STRING*".

A terceira forma acabamos de aprender e é através do READ-DATA.

É evidente que no READ teremos variáveis alfanuméricas e no DATA aparecerão escritas palavras ou frases separadas por vírgulas.

Desta forma são válidas as seguintes representações para uma variável alfanumérica.

B8\$, PX\$, A9\$, AB\$, C\$

O seu micro aceita duas letras junto?

Veja isto com muito cuidado.

Algumas "dicas" úteis para a boa utilização do seu micro.

1) PRINT ponto de interrogação (?).

É muito comum encontrar versões de BASIC nas quais a palavra PRINT pode ser substituída pelo caractere (?) ou seja pelo ponto de interrogação.

Dessa forma em vez de escrever PRINT usa-se o sinal ? o que diminui o esforço datilográfico, entre outras vantagens.

Note que a instrução PRINT é uma das que você mais usa nos programas.

2) Ao se introduzir várias variáveis em uma única linha INPUT, se "falhamos" em entrar com todas elas, separadas por vírgulas, um sinal especial (??) nos alerta sobre o fato de que mais dados precisam entrar.

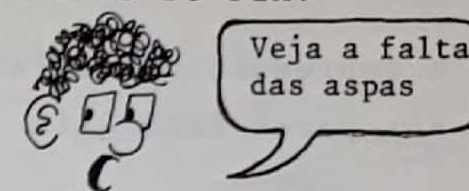
3) O uso das aspas

Tecnicamente, em alguns micros, não é necessário o uso de aspas para fechar, muitas das coisas escritas dentro da instrução PRINT.

Assim a instrução:

10 PRINT "ONDE ESTÃO AS ASPAS DO FIM?"

é executada corretamente em alguns micros.



Por medida de precaução entretanto recomendamos sempre abrir e fechar as aspas.

4) O NEXT opcional

Em alguns "LOOPS" FOR-NEXT nem sempre é necessário especificar qual FOR nos estamos seguindo até o próximo NEXT.

Isto pode ser de grande utilidade se tivermos LOOPS aninhados. Este é o caso do MAXXI?

5) A instrução STOP determina parada no processamento na linha em que for encontrada.

Ela substitui desta forma o END que não existe em alguns micros.

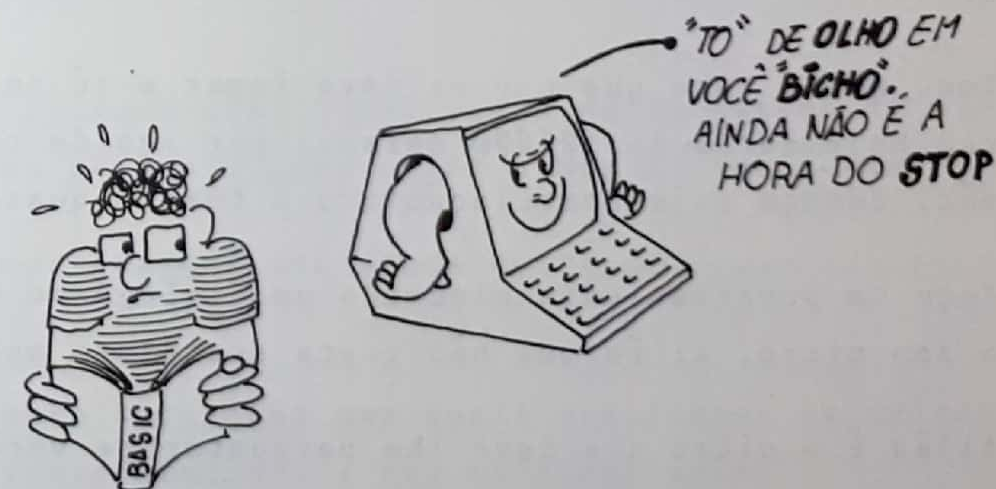


Figura 3.5

6) Nos já falamos do CONT no §2.2 porém vamos repetir que é um comando que manda o programa continuar a ser executado após uma interrupção causada pelo CTRL C (Control C), pela instrução STOP ou em alguns casos, pela tecla BREAK.

Exemplo 3.4

Vejamos a impressão de variáveis alfanuméricas e numéricas através dos pares READ-DATA.

```
10 DATA "A DATA DE HOJE É", 27, " DE ABRIL DE", 1983
20 READ A$, A, B$, B
30 LPRINT A$; A; B$; B
40 END
RUN
```

Teremos na saída:

A DATA DE HOJE É 27 DE ABRIL DE 1983

TRABALHO PARA VOCÊ

Note bem, sempre que você conseguir fazer um programa *sozinho* não é um raio de esperança que surge mas sim forma-se um feixe de certeza que você aprenderá a dominar o seu micro...

Todo mundo sabe que não se deve fumar e só na Europa morrem hoje cerca de 1000000 pessoas por ano de câncer do pulmão, doença cuja correlação com o fumo é quase que "total".

Faça um programa estabelecendo um diálogo do operador com o seu micro, aliás que não gosta que fumem perto dele.

Aliás é o micro que deve lhe perguntar se você *fuma* ou *não*.

A entrada que você dará será uma variável alfanumérica "SIM" ou "NÃO". Em caso de você fumar (o que é uma pena) dirá quanto custa um maço de cigarros, quantos fuma por dia e o micro lhe dará qual é o seu gasto por ano (ele, só não lhe chama de imbecil e o quanto você encurta a sua vida por delicadeza...).

No caso de você não fumar receberá efusivos parabéns pelo seu comportamento normal, pois não há nada de excepcional em ser um fumante.

E abaixo o hábito de fumar!!!

Palavras de um micro e do autor do livro.



Figura 3.6 - "É proibido fumar perto do micro. Aliás deveria ser proibido fumar".

§3.3 - A NECESSIDADE DA VARIÁVEL INDEXADA OU SUBSCRITA

Já sabemos que podemos usar combinações de 26 letras do alfabeto e dígitos de 0 a 9 para criar nomes de variáveis.

Nos programas que fizemos até agora nos precisávamos de poucas variáveis.

Existem entretanto situações nas quais precisamos de muitas variáveis, as vezes *centenas* e até *milhares* delas.

Nesse caso não convém criar tantos nomes diferentes e tenta-se criar um nome para representar a coleção de variáveis.

Dessa forma as variáveis podem ser *simples* ou *coletivas*.

Enquanto uma *variável simples* só pode representar a cada instante um único valor uma *variável coletiva* ou *conjunto* (em inglês ARRAY) pode representar uma coleção de valores TODOS DO MESMO TIPO.

Esse conjunto é ordenado, ou seja, é possível atribuir a cada elemento um número de ordem ou posição relativa dentro do mesmo.

Atribuindo-se à variável conjunto um nome, é possível referenciar cada elemento do conjunto em particular através de um índice ou subscrito que representa o seu número de ordem.

Este subscrito vem entre parênteses após o nome do conjunto.

O nome de um coletivo aplica-se tanto ao conjunto como a cada elemento.

As regras para atribuir nomes a conjuntos (ARRAYS) são as mesmas que para as variáveis simples sejam numéricas ou alfanuméricas.

Assim são nomes de conjuntos numéricos

V, A1, S2

e de conjuntos alfanuméricos (STRINGS)

M1\$, M2\$, M3\$, N\$, B\$

Toda chamada a um elemento do conjunto é uma variável subscrita e é formada pelo nome de conjunto acompanhado do subscrito ou índice.

É por isto então que as variáveis simples são também chamadas de *variáveis não subscritas*.

O índice ou subscrito pode ser qualquer *expressão numérica positiva* de cujo cálculo surja um valor que, truncado, fornece o número de ordem ou posição relativa do elemento dentro do conjunto.

Já existem sistemas BASIC que admitem conjuntos (ARRAYS) com até seis dimensões entretanto é comum que um sistema BASIC aceite conjuntos até duas dimensões.

Um conjunto de uma dimensão é chamado de vetor ou seja é uma matriz linha ou coluna.

Há quem traduza ARRAY como sendo matriz.

Um conjunto de duas dimensões é chamado matriz devido a analogia com a Matemática.

Como exemplo de referências a conjuntos de uma única dimensão temos as variáveis subscritas

A(J+7), Y\$(B3-2.87), A(N) etc.

Para o caso de duas dimensões podemos ter as variáveis subscritas ou indexadas:

E(5,8), A(J**4,I-3), X\$(A**2,Y) etc.

Quando trabalhamos com variáveis indexadas a linguagem BASIC estabelece uma certa quantidade de memórias para armazenar os valores dessas variáveis.

Em outras palavras, o programa precisa saber antes de começarem a aparecer as variáveis indexadas, qual é o tamanho máximo do conjunto (ARRAY) que se terá ou seja quantos elementos existirão nas matrizes ou nos vetores.

Na maior parte das versões de BASIC não é necessário declarar tamanhos de conjuntos com menos de 11 elementos.

Isto significa que, se tivermos uma variável do tipo A(I) e se I na execução não for maior que 10 não será preciso dar uma instrução especial declarando qual o maior valor que se espera que I possa atingir.

A instrução DIM (oriunda do "DIMENSION" ou seja dimensão) é usada da seguinte forma.

Se no início do programa escrevemos

10 DIM A(50)

estamos declarando que o conjunto A(I) poderá ter os elementos A(0), A(1), A(2), ... A(50) ou seja se armazenará um

total de 51 valores se bem que geralmente a variável $A(\emptyset)$ não é usada.

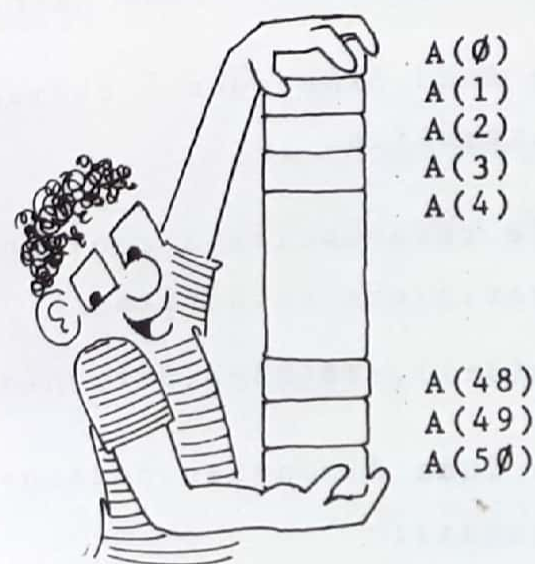


Figura 3.7 - "Quando você DIMensionna uma variável alfanumérica..."

No caso de um índice duplo a instrução DIM poderá ser do tipo

`DIM A(15,18)`

o que significa que a variável A poderá assumir índices desde $A(\emptyset, \emptyset)$ até $A(15,18)$.

Uma maneira de entender a necessidade de uma eventual variável coletiva bidimensional está bem explicada na Figura 3.8.



Figura 3.8

Tanto o tamanho máximo de cada dimensão de vetores e matrizes como o número máximo de caracteres de variáveis alfanuméricas são expressões numéricas positivas quaisquer.

A instrução DIM pode estar colocada em qualquer parte do programa, pois é uma instrução de especificação, isto é, não executável.

Cada variável que for declarada só pode aparecer uma única vez em uma instrução DIM em todo o programa.

Pode existir um número qualquer de instruções DIM no programa, isto é, a declaração de variáveis é completamente arbitrária.

Ao se começar a execução do programa, o sistema BASIC automaticamente inicializa as variáveis e conjuntos, isto é, coloca zero ($\emptyset$) em elementos de conjuntos numéricos e coloca vazio (limpa). Nos elementos dos ARRAYS.

A última imagem para "enxergar" bem a necessidade de uma variável coletiva está nas Figuras 3.9 e 3.10.

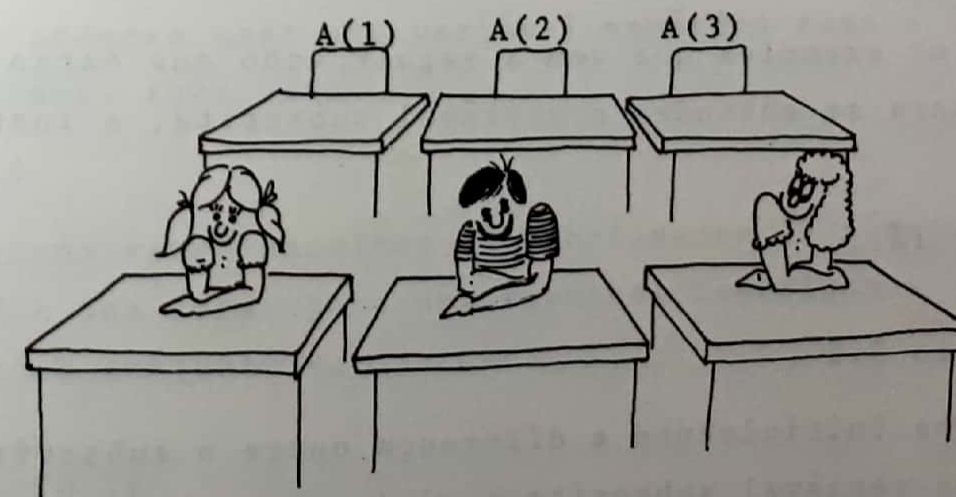


Figura 3.9 - As crianças estão sentadas em linha e a variável coletiva A ou vetor linha contém os elementos $A(1)$, $A(2)$ e $A(3)$ e em cada um está guardada a idade de cada uma delas.

Linha 2

Linha 1

Coluna 1

Coluna 2

Coluna 3

Figura 3.10 - A variável subscrita $A(I,J)$ dada é a forma genérica de representar os elementos $A(1,1)$, $A(1,2)$, $A(1,3)$, $A(2,1)$, $A(2,2)$, $A(2,3)$ da matriz bidimensional A . A cada um desses elementos se atribui a idade de cada criança.

Como se vê da Figura 3.10 o número de linha corresponde ao primeiro subscrito, enquanto o número da coluna corresponde ao segundo subscrito.

Não é difícil de perceber uma matriz tridimensional e basta pensar em uma escola com dois andares.

Teremos alunos definidos por linhas, colunas e andares quanto a posição onde se encontram no prédio.

"Manjou"?

Bem, os exemplos que vem a seguir, acho que darão o golpe final, para se entender a variável subscrita, a instrução DIM, etc.

Vamos lá...

Exemplo 3.5

Veamos inicialmente a diferença entre o subscrito e o valor da variável subscrita ou indexada.

```

10 A(1)=13
20 A(2)=7
30 A(3)=4
40 LPRINT "A(1)=";A(1),"A(2)=";A(2),"A(3)=";A(3)
50 LPRINT "A(1+2)=";A(1+2)
60 LPRINT "A(1)+A(2)=";A(1)+A(2)
70 X=2
80 LPRINT "A(X)=";A(X)
90 END
RUN

```

Diagram illustrating the values of the array A :

- $A(1) = 13$
- $A(2) = 7$
- $A(3) = 4$

Arrows point from the values 13, 7, and 4 to their respective labels $A(1)$, $A(2)$, and $A(3)$.

O seu micro precisa do LET para atribuir

159

Caso a resposta seja afirmativa coloque o LET onde for necessário



Teremos a saída:

```

A(1)= 13      A(2)= 7      A(3)= 4
A(1+2)= 4
A(1)+A(2)= 20
A(X)= 7

```

Como se vê nas linhas 50 e 60 mostra-se claramente a diferença entre adicionar dos subscritos e adicionar o valor de duas variáveis

$A(1+2)$ é a mesma coisa que $A(3)$ e tem o valor 4.

$A(1)+A(2)$ chama os valores 13 e 7 para a soma obtendo-se valor final 20.

Desde que o subscrito X é igual a 2 devido a instrução 70, na linha 80 foi impresso o valor 7.

Como se vê pode-se usar uma variável numérica como o índice de uma variável subscrita.

Exemplo 3.6

Neste programa vamos escolher aleatoriamente 10 números entre 1 e 20 e sem ligar para uma eventual repetição e vamos guardá-los no conjunto R.

```

10 FOR Y=1 TO 10
20 R(Y)=RND(20)
30 NEXT Y
40 LPRINT "DEZ NÚMEROS ALEATÓRIOS FORAM ARMAZENADOS EM R()."
50 LPRINT
60 FOR N=1 TO 10
70 LPRINT "R(";N;") TEM UM";R(N);"ARMAZENADO NELA."

```



```

80 NEXT N
90 END
RUN

```

Uma eventual saída pode ser do tipo:

DEZ NÚMEROS ALEATÓRIOS FORAM ARMAZENADOS EM R().

```

R( 1 ) TEM UM 1 ARMAZENADO NELA.
R( 2 ) TEM UM 6 ARMAZENADO NELA.
R( 3 ) TEM UM 17 ARMAZENADO NELA.
R( 4 ) TEM UM 8 ARMAZENADO NELA.
R( 5 ) TEM UM 6 ARMAZENADO NELA.
R( 6 ) TEM UM 7 ARMAZENADO NELA.
R( 7 ) TEM UM 19 ARMAZENADO NELA.
R( 8 ) TEM UM 10 ARMAZENADO NELA.
R( 9 ) TEM UM 8 ARMAZENADO NELA.
R( 10 ) TEM UM 5 ARMAZENADO NELA.

```

Este programa tem dois LOOPS os quais não estão aninhados, porém um segue o outro.

Cada laço é executado 10 vezes.

No LOOP que começa na linha 20 escolhe-se um número aleatório e armazena-se o mesmo em uma das variáveis subscritas R().

Na primeira volta desta malha (Y=1), um número aleatório é armazenado em R(1), na segunda volta (Y=2) um novo número é guardado em R(2) e assim por diante 10 vezes.

O segundo LOOP (linhas 60 até 80) imprime o conteúdo armazenado nas "caixas" R().

TRABALHO PARA VOCÊ

Repita o Exemplo 3.6 fazendo com que os dois laços FOR-NEXT estejam um aninhado dentro do outro.

Exemplo 3.7

Neste programa será lida aleatoriamente uma lista de 5 nomes dentro de uma variável alfanumérica (STRING) subscrita N\$() sem repetir nenhum dos nomes.

```

10 FOR X=1 TO 5
20 Y=RND(5)
30 IF N$(Y) <> "" THEN 20
40 READ N$(Y)
50 NEXT X
60 FOR Z=1 TO 5
70 PRINT N$(Z)
80 NEXT Z
90 DATA NILZA MARIA,ALEXANDRE,SERGIO,VICTOR JR.,VICTOR
100 END
RUN

```

Dã para escrever sem o LET no seu micro?



Uma eventual saída poderia ser:

```

ALEXANDRE
VICTOR JR.
SERGIO
NILZA MARIA
VICTOR

```

```

VICTOR JR.
NILZA MARIA
VICTOR
SERGIO
ALEXANDRE

```

```

SERGIO
VICTOR
NILZA MARIA
VICTOR JR.
ALEXANDRE

```

```

VICTOR
NILZA MARIA
ALEXANDRE
VICTOR JR.
SERGIO

```

```

SERGIO
NILZA MARIA
VICTOR
VICTOR JR.
ALEXANDRE

```

Este programa lê os nomes na instrução DATA na ordem da ocorrência, porém coloca-os aleatoriamente em N\$(1) até N\$(5).

A "repetição" é evitada através da verificação de cada nova "caixa", descobrindo a medida que é escolhida, se contém um nome.

Se ela está cheia tenta-se uma nova caixa ou gaveta escolhendo-se um novo número.

Na linha 30 as aspas duplas e consecutivas referenciam se uma gaveta ou caixa está vazia.

Informação muito IMPORTANTE

No micro representa-se com ("") quando se tem uma variável alfanumérica subscrita sem atribuição ou seja que está vazia ou limpa.



Figura 3.11 - "Todas as caixas devem estar cheias e com coisas diferentes para a variável coletiva N\$()"

Exemplo 3.8

Uma classe tem 15 carteiras divididas em 5 linhas e 3 colunas.

Faça um programa que atribua aleatoriamente antes que 14 estudantes entrem na sala, o lugar onde eles devem sentar.

Uma carteira deve ficar vazia pois é lá que as vezes descansa a professora Olívia Olho Vivo.

Solução:

```

10 DIM N$(5,3)
20 FOR X=1 TO 14
30 READ A$
40 L=RND(5)
50 C=RND(3)
60 IF N$(L,C) <> "" THEN 40
70 N$(L,C)=A$
80 NEXT X
90 FOR L1=1 TO 5
100 FOR C1=1 TO 3
110 IF N$(L1,C1)="" THEN LPRINT "VAZIA" ELSE LPRINT N$(L1,C1),
120 NEXT C1
130 LPRINT
140 NEXT L1
150 DATA FUSCA,MIURA,POTI,EDMEA,KELLY,""
160 DATA GONZA,ROSINHA,"ORELHA",FRALDA
170 DATA COELHÃO,GUTO,DUMBO,SACHA,CARVÃOZINHO
180 END
RUN

```

Outra vez não colocamos o LET. Isto vale no seu micro? E o RND é também assim?

Uma possível saída poderia ser:

FUSCA	SACHA	EDMEA
COELHÃO	CARVÃOZINHO	GONZA
GUTO	ROSINHA	VAZIA
DUMBO	FRALDA	ORELHA
POTI	MIURA	KELLY

Importante:

Como já dissemos podemos ter mais de duas dimensões, porém quanto mais dimensões tivermos corremos o risco de ao se declarar isto na instrução DIM acabemos com toda a memória do nosso esperto e prestativo micro!!!



Figura 3.12 - "Na vida futura o nosso servo Robotoff, o mestre em manipular variáveis coletivas corretamente dimensionadas através da instrução DIM poderá guardar muita coisa sob o mesmo nome".

Exemplo 3.9

Aqui temos um programa que utiliza um conjunto alfanumérico (STRING ARRAY) unidimensional.

O leitor(ou leitora) facilmente perceberá que ele é controlado por três L00Ps ou laços, um para controlar a repetição através da página, um para escolher as linhas do modelo e um para controlar o número de linhas do tipo de desenho que desejamos fazer.

Solução:

```

10 DIM J$(10)
20 LPRINT"QUANTOS CARACTERES POR LINHA"
27 INPUT C
30 LPRINT"QUANTAS LINHAS POR BLOCO"
37 INPUT L
38 LPRINT"QUANTAS LINHAS DE BLOCOS"
39 INPUT R
40 FOR J=1 TO L
50 LP$PRINT"IMPRIMA ";C;" CARACTERES PARA A LINHA # ";J
60 INPUT J$(J)
70 NEXT J
100 LPRINT" QUANTOS BLOCOS ATRAVÉS DA PÁGINA"
105 INPUT B

```

```

110 LPRINT"VOCÊ PRECISA DE UM TERMINAL COM";B*C;" COLUNA(S)"
120 LPRINT"DESEJA MUDAR O # DE BLOCOS(0=NÃO)"
127 INPUT A
130 IF A>0 THEN 100
200 FOR I=1 TO R
210 FOR J=1 TO L
220 FOR K=1 TO B
230 LPRINT J$(J);
240 NEXT K
250 LPRINT
260 NEXT J
270 NEXT I
280 END
RUN

```

Estes três LOOPS imprimem o modelo ou o tipo de desenho que se quer. R controla o número de linhas de blocos, L o número de linhas por bloco e B o número de blocos por linha

Estes três LOOPS imprimem o modelo ou o tipo de desenho que se quer.
R controla o número de linhas de blocos, L o número de linhas por bloco e B o número de blocos por linha

Se entrarmos com os seguintes caracteres ou valores (cabe a você identificar onde entrou cada um).

CARACTERES

OU

VALORES: 6,5,7,!!!!!!,!.....!,&&&&&,131313,777777,7,0

teremos a seguinte saída:

[illegible]

[illegible]

E agora já está na hora de dar um SUPER-TRABALHO PARA VOCÊ

- 1) Escolha 20 números aleatórios entre 7 e 77 inclusive.
Imprima os ímpares em uma linha e os pares na linha seguinte.

Na saída deve-se ter algo semelhante à Tabela 3.2

INTEIROS ÍMPARES:	7	17	31	33	13	75	47	21	37		
INTEIROS PARES:	48	52	14	28	70	56	66	18	20	34	42

Tabela 3.2

- 2) Juquinha "inventou" o seguinte jogo para ser disputado com os seus amigos:

Paga-se 20 cruzeiros para jogar e tem-se direito de lançar 10 vezes uma placa circular e pesada dentro de um tabuleiro que está marcado com os números 1, 2 e 3 conforme a Figura 3.13.

1	1	1	1	1	1
1	2	2	2	2	1
1	2	3	3	2	1
1	2	3	3	2	1
1	2	2	2	2	1
1	1	1	1	1	1

Figura 3.13

Supondo que se acerta sempre dentro do tabuleiro que está colocado a razoável distância do jogador e que não ocorrerão discussões (o que não é fácil) se a placa está na casela com o número 1 ou 2 ou ainda 2 ou 3, escreva um programa que simule este jogo onde o número de pontos do jogador é simplesmente a soma dos números das caselas em que para a placa, em cada lançamento.

Imprima também o ganho líquido de cada jogador (este ganho pode ser negativo, nulo ou positivo).

- 3) É dada uma matriz A que é (6×6) ou seja tem 6 linhas e 5
colunas.

Faça um programa que dê a soma de todos os elementos da matriz A com exceção daqueles que estão na periferia.

Imprima todos os números que entraram e a matriz "resultante" B sem os elementos da periferia.

$$A = \begin{bmatrix} 28 & 29 & 30 & 31 & 32 & 33 \\ 13 & 13 & 13 & 13 & 13 & 13 \\ 7 & 7 & 7 & 7 & 7 & 7 \\ 640 & 641 & 642 & 643 & 644 & 645 \\ -20 & -21 & -22 & -23 & -24 & -25 \\ 42 & 41 & 42 & 41 & 42 & 41 \end{bmatrix}$$

$$B = \begin{bmatrix} 13 & 13 & 13 & 13 \\ 7 & 7 & 7 & 7 \\ 641 & 642 & 643 & 644 \\ -21 & -22 & -23 & -24 \end{bmatrix}$$

- 4) Considere uma operação de fabricação muito simples na qual se fazem 3 modelos:

Modelo A(1) (padrão)

Modelo A(2) (de luxo)

Modelo A(3) (super)

Na Tabela 3.3 estão dispostas as necessidades de componentes C_1 , C_2 , C_3 , C_4 e C_5 para se obter uma unidade de A(1), A(2) e A(3).

Componente Modelo	C_1	C_2	C_3	C_4	C_5
Padrão A(1)	2	0	1	0	0
De luxo A(2)	2	0	0	1	1
Super A(3)	0	2	0	1	2

Tabela 3.3

Suponha que o fabricante queira produzir 100 modelos de A(1), 150 de A(2) e 75 de A(3).

Faça um programa que imprima para o fabricante quanto de cada componente ele deve comprar para satisfazer a sua meta de produção.

Sugestão: Um trecho de programa, que permita executar a multiplicação matricial (matriz linha por uma quadrada) é:

```
10 FOR J= 1 TO L
20 FOR K=1 TO N
30 LET C(J)=C(J)+A(K)*B(K,J)
40 NEXT K
50 NEXT J
```

- 5) O custo médio por unidade de produto C_p é a razão do custo total pelo número de unidades produzidas e é expresso por:

$$C_p = \frac{T_p}{N}$$

onde T_p é o custo total e N é o número de unidades produzidas.

O custo marginal é o custo de produzir uma unidade a mais do produto e a medida que se vai aprendendo a produzi-lo o custo marginal deve ir diminuindo.

Quantidade produzida	Preço de venda	Custo total
1	241	520
2	230	530
3	215	680
4	190	750
5	152	760
6	128	768
7	111	777
8	99	784

Tabela 3.4

Faça um programa que leia como dados os preços e os custos totais associados com as quantidades produzidas (1, 2, 3, 4... unidades) e que calcule e imprima a tabela de custos médios e marginais. Depois de 8 unidades produzidas já se tem lucro?

- 6) São dadas três listas compostas: uma de números positivos, outra de negativos e outra com negativos e positivos.

Faça um programa para "arrumar" as três listas na ordem crescente.

Lista A	Lista B	Lista C
14	-44	16
6	-1	-3
32	-14	14
1	-3	-8
18	-31	36
4	-16	44
16	-22	-18
22	-23	2
31	-6	0
29	-7	-26
3	-2	19
26	-19	27
8	-15	30
12	-10	-2
17	-8	7

Tabela 3.5

7) Faça um programa para que saia na ordem alfabética a lista de seus compositores de música clássica.

Os dados para este programa são:

BEETHOVEN, GOMES, VERDI, MOZART,
BRAHMS, RESPIGHI, ELGAR, DELIUS,
VIVALDI, PURCELL, POULENC, THCAIKOVSKY,
PROKOFIEV, BORODIN, GLINKA, FAURE, SACHA 7

Realmente o SACHA 7 não é ainda compositor porém é um grande filho.

8) Elabore um programa que tenha as seguintes características:

- permita a introdução de até 7 palavras ou frases com um comprimento máximo de 13 caracteres.
- coloque as frases ou palavras introduzidas em ordem alfabética.

c) Modifique o programa anterior para que o número de palavras introduzidas seja estabelecido no momento em que o programa começa a ser executado.

Observação importante: No CP-200 precisamos escrever DIM M\$(7,13) para dimensionar uma matriz para 7 STRINGS de comprimento máximo 13 caracteres. CUIDADO QUE EM OUTROS MICROS PODE SER DIFERENTE!

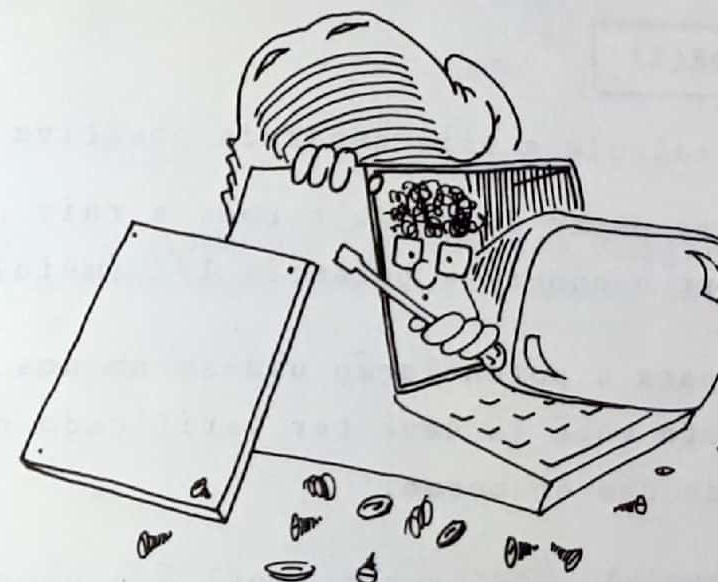


Figura 3.14 - "Se algum dos últimos programas não der certo, não tente consertar o seu micro mas sim o seu programa".

§3.4 - FUNÇÕES MATEMÁTICAS

O BASIC inclui um certo número de funções matemáticas as vezes chamadas "funções de biblioteca".

Todas estas funções matemáticas são muito simples de serem usadas e se os seus conhecimentos matemáticos são insuficientes ou ficaram obscurecidos pelo tempo tentaremos clarear ou refrescar os mesmos...

Nós já estudamos a função INT(N) e a SGN(X).

Comecemos pela função FIX(N).

Esta função, uma vez aplicada a uma expressão numérica ou a

uma variável numérica, faz retornar apenas a parte inteira do número, isto é *trunca* todos os dígitos à direita do ponto decimal.

Assim com a instrução

```
10 LET N=FIX(3.78)
```

atribui-se a N o valor 3.

A função SQR(X)

A função SQR calcula a raiz quadrada positiva de um número.

É claro que uma outra forma de termos a raiz quadrada de um número é elevar o número a potência $1/2$ (meio).

Já vimos que para a potenciação usa-se um dos símbolos $**$ ou $\uparrow$ ou $^$ e isto você já deve ter verificado no seu micro para o adequado uso do mesmo.

Na expressão SQR(X), SQR(square root) é o nome da função e X é chamado de argumento.

Na função SQR, o argumento pode ser qualquer expressão matemática com valor não negativo.

Por exemplo SQR ($4*X+7$) é perfeitamente aceitável desde que a expressão $4*X+7$ não gere um número negativo.

Deve-se notar ainda que o argumento de uma função precisa estar dentro dos parênteses.

Na avaliação de uma função matemática o computador inicialmente calcula o argumento e então a função, usando o valor previamente obtido para o argumento.

Algumas funções tem uma limitação no valor do argumento.



Figura 3.15 - "Realmente brincar com funções matemáticas parece coisa de gente adulta e mostraremos que os "pequenos" também podem entender..."

Exemplo 3.10

Pitágoras descobriu que os lados de um triângulo retângulo obedecem sempre a regra $a^2+b^2=c^2$ onde c é o comprimento do maior lado que se chama hipotenusa.

Os outros dois lados chamam-se catetos.

Uma outra forma de estabelecer isto é dizer que $c = \sqrt{a^2+b^2}$.

Se temos um triângulo retângulo com os lados $a=5$ e $b=12$ escreva um programa para achar o comprimento da hipotenusa c .

```
10 CLS
20 A=5 : B=12
30 C=SQR(A^2+B^2)
40 LPRINT "A RAIZ QUADRADA DE ";A;" ELEVADO AO QUADRADO MAIS";B
50 LPRINT "ELEVADO AO QUADRADO É ";C
60 LPRINT
70 END
RUN
```

Linha de instruções múltiplas e sem o LET. Pode?
Oi é assim a exponenciação no seu micro?

Deve-se ter na saída:

```
A RAIZ QUADRADA DE 5 ELEVADO AO QUADRADO MAIS 12
ELEVADO AO QUADRADO É 13
```


TRABALHO DE EXPANSÃO MENTAL PARA VOCÊ

CUIDADO PARA NÃO SE TORNAR O PRÍNCIPE SUBMARINO E NOTE QUE PENSAR NÃO CANSA.

- 1) Elabore um programa para achar todas as trincas de números inteiros e positivos que seriam pitagóricos até o número 30.

Use para isto três pares de instruções FOR-NEXT uma aninhada dentro da outra.

Não esqueça que deve-se ter $k^2 = i^2 + j^2$ o que ocorre se $i=3$, $j=4$ e $k=5$.

- 2) Elabore um programa no qual se entra com 15 números quaisquer e faça o mesmo colocar os números em ordem crescente antes de imprimi-los, juntamente com suas respectivas raízes. Cuidado com o caso de se ter números negativos!



Figura 3.16 - "Esse STOP não é para parar e sim para um pequeno descanso".

Exemplo 3.11

O programa que iremos analisar a seguir é um jogo que é uma versão em "grid" ou seja em reticulado da batalha naval.

Esta também é uma boa hora para que o pequeno gênio comece a se familiarizar com o sistema de coordenadas cartesianas ou seja o sistema X-Y.

O programa pede para que se ache quatro "espiões" que estão "escondidos" nos pontos de um reticulado 40×40 .

Os pontos ou seja as posições possíveis do reticulado tem a forma (J,K) onde J e K são inteiros entre 0 e 9 inclusive.

O jogador pode dar 10 "chutes" para achar em que posição estão os espiões.

Em cada tentativa diz-se ao jogador ou a jogadora o quanto a sua adivinhação está próxima de cada espião.

No fim do jogo você pode perguntar ao micro onde estão os espiões que você não encontrou.

Se você jogar um outro jogo, os espiões movem-se para novas posições.

Observação importante:

Depois de participar do jogo algumas vezes apenas usando a "cuca", tente fazer o mesmo com a ajuda de um papel quadriculado, um compasso e a "cuca".

Você provavelmente aprenderá a achar todos os jogadores em não mais que 7 movimentos (que número bonito é o 7 não é Sacha!) usando uma "artimanha" que é similar à alguns sistemas de radio navegação.

Quem sabe podem até surgir algumas idéias para melhorar o programa!!!

Solução:

```
10 DIM P(4,2)
20 LPRINT "ESTE É O JOGO DOS ESPIONS EM UMA MALHA RETICULADA."
21 LPRINT "A FINALIDADE É ACHAR 4 ESPIONS ESCONDIDOS EM UM RETICULADO 40x40."
22 LPRINT "A SUA BASE DE BUSCA ESTÁ EM (0,0)."
```

23 LPRINT "QUALQUER ADIVINHAÇÃO QUE VOCÊ FAÇA DEVE CONTER 2 NÚMEROS SEPARADOS POR VÍRGULA."

25 LPRINT " O PRIMEIRO NÚMERO É A DISTÂNCIA A DIREITA DA BASE."

26 LPRINT " O SEGUNDO NÚMERO É A DISTÂNCIA ACIMA DA BASE."


```

30 LPRINT
40 LPRINT "
100 GOSUB 1000
110 LET T=0
120 LET T=T+1
130 LPRINT
140 LPRINT
150 LPRINT "TENTATIVA #";T;"QUAL É O SEU PALPITE?"
160 INPUT M,N
180 FOR I=1 TO 4
190 IF P(I,1)=-1 THEN 400
200 IF P(I,1)<>M THEN 380
220 IF P(I,2)<>N THEN 380
230 LET P(I,1)=-1
300 LPRINT "VOCÊ ACHOU O ESPIÃO";I
310 GOTO 400
380 LET D=SQR((P(I,1)-M)^2+(P(I,2)-N)^2)
390 LPRINT "SUA DISTÂNCIA DO ESPIÃO";I;"É";INT(D*100+0.5)/100
400 NEXT I
410 FOR J=1 TO 4
420 IF P(J,1)<>-1 THEN GOTO 470
430 NEXT J
440 LPRINT
450 LPRINT "VOCÊ ACHOU TODOS OS ESPIÕES EM";T;"TENTATIVAS!!"
460 GOTO 580
470 IF T<10 THEN GOTO 120
480 LPRINT
490 LPRINT "VOCÊ NÃO ACHOU TODOS OS ESPIÕES EM DEZ TENTATIVAS"
500 LPRINT "VOCÊ QUER SABER ONDE ESTÃO OS ESPIÕES QUE NÃO FORAM ENCON-
TRADOS?"
510 INPUT J$
530 IF J$="NÃO" THEN GOTO 580
540 FOR I=1 TO 4
550 IF P(I,1)=-1 THEN GOTO 570
560 LPRINT "ESPIÃO";I;"ESCONDIDO EM (";P(I,1);",";P(I,2);")."
570 NEXT I
580 LPRINT
590 LPRINT "VOCÊ QUER JOGAR DE NOVO?"
600 INPUT V$
610 IF V$="SIM" THEN GOTO 100
620 LPRINT "PASSE MUITO BEM E CUIDADO PARA NÃO CAIR EM UM BUEIRO!!!"
630 GOTO 1300
1000 FOR J=1 TO 2
1010 FOR I=1 TO 4
1020 P(I,J)=10*RND(4)
1030 NEXT I
1040 NEXT J
1050 RETURN
1100 LPRINT
1300 END
RUN

```

COMECE BICHO"

Ficaria mais complicado o jogo se fosse $P(I,J)=RND(40)?$



Uma eventual saída poderia ser do tipo:

ESTE É O JOGO DOS ESPIÕES EM UMA MALHA RETICULADA.
A FINALIDADE É ACHAR 4 ESPIÕES ESCONDIDOS EM UM RETICULADO 10x10.
A SUA BASE DE BUSCA ESTÁ EM (0,0).
QUALQUER ADIVINHAÇÃO QUE VOCÊ FAÇA DEVE CONTER 2 NÚMEROS SEPARADOS POR VÍRGULA.

O PRIMEIRO NÚMERO É A DISTÂNCIA A DIREITA DA BASE,
O SEGUNDO NÚMERO É A DISTÂNCIA ACIMA DA BASE.

COMECE BICHO

TENTATIVA # 1 QUAL É O SEU PALPITE?
SUA DISTÂNCIA DO ESPIÃO 1 É 14.14
SUA DISTÂNCIA DO ESPIÃO 2 É 31.62
SUA DISTÂNCIA DO ESPIÃO 3 É 36.06
SUA DISTÂNCIA DO ESPIÃO 4 É 42.43

TENTATIVA # 2 QUAL É O SEU PALPITE?
SUA DISTÂNCIA DO ESPIÃO 1 É 30.41
SUA DISTÂNCIA DO ESPIÃO 2 É 11.18
SUA DISTÂNCIA DO ESPIÃO 3 É 32.02
SUA DISTÂNCIA DO ESPIÃO 4 É 26.93

TENTATIVA # 3 QUAL É O SEU PALPITE?
SUA DISTÂNCIA DO ESPIÃO 1 É 127.28
SUA DISTÂNCIA DO ESPIÃO 2 É 114.02
SUA DISTÂNCIA DO ESPIÃO 3 É 106.3
SUA DISTÂNCIA DO ESPIÃO 4 É 98.99

TENTATIVA # 4 QUAL É O SEU PALPITE?
SUA DISTÂNCIA DO ESPIÃO 1 É 268.7
SUA DISTÂNCIA DO ESPIÃO 2 É 254.95
SUA DISTÂNCIA DO ESPIÃO 3 É 247.59
SUA DISTÂNCIA DO ESPIÃO 4 É 240.42

TENTATIVA # 5 QUAL É O SEU PALPITE?
VOCÊ ACHOU O ESPIÃO 1
SUA DISTÂNCIA DO ESPIÃO 2 É 20
SUA DISTÂNCIA DO ESPIÃO 3 É 22.36
SUA DISTÂNCIA DO ESPIÃO 4 É 28.28

TENTATIVA # 6 QUAL É O SEU PALPITE?
SUA DISTÂNCIA DO ESPIÃO 2 É 31.62
SUA DISTÂNCIA DO ESPIÃO 3 É 22.36
SUA DISTÂNCIA DO ESPIÃO 4 É 31.62

TENTATIVA # 7 QUAL É O SEU PALPITE?
 SUA DISTÂNCIA DO ESPIÃO 2 É 14.14
 SUA DISTÂNCIA DO ESPIÃO 3 É 30
 SUA DISTÂNCIA DO ESPIÃO 4 É 31.62

TENTATIVA # 8 QUAL É O SEU PALPITE?
 SUA DISTÂNCIA DO ESPIÃO 2 É 26.25
 SUA DISTÂNCIA DO ESPIÃO 3 É 10.44
 SUA DISTÂNCIA DO ESPIÃO 4 É 20.22

TENTATIVA # 9 QUAL É O SEU PALPITE?
 SUA DISTÂNCIA DO ESPIÃO 2 É 17.72
 SUA DISTÂNCIA DO ESPIÃO 3 É 16.55
 SUA DISTÂNCIA DO ESPIÃO 4 É 22.67

TENTATIVA # 10 QUAL É O SEU PALPITE?
 SUA DISTÂNCIA DO ESPIÃO 2 É 45.28
 SUA DISTÂNCIA DO ESPIÃO 3 É 23.02
 SUA DISTÂNCIA DO ESPIÃO 4 É 29.83

VOCÊ NÃO ACHOU TODOS OS ESPIÕES EM DEZ TENTATIVAS
 VOCÊ QUER SABER ONDE ESTÃO OS ESPIÕES QUE NÃO FORAM ENCONTRADOS?
 ESPIÃO 2 ESCONDIDO EM (30, 10).
 ESPIÃO 3 ESCONDIDO EM (20, 30).
 ESPIÃO 4 ESCONDIDO EM (30, 30).

VOCÊ QUER JOGAR DE NOVO?
 PASSE MUITO BEM E CUIDADO PARA NÃO CAIR EM UM BUEIRO!!!!

Note que as entradas via instrução INPUT não aparecem impressas porém aparecem na tela do vídeo.



Figura 3.17 - "Estás começando a achar mesmo que os micros são espertos?"

Função ABS(N)

A função ABS tem muito a ver com sinais ou sem eles.

O valor absoluto de qualquer número é o número sem um sinal. Se você esqueceu o programa abaixo rapidamente lhe refrescará a memória.

Exemplo 3.12

```
10 INPUT "ENTRE COM QUALQUER NÚMERO";N
20 A=ABS(N)
30 PRINT A
40 PRINT
50 GOTO 10
RUN
```

Sem o LET
E precisa?



Experimente neste programa números pequenos, grandes, positivos, negativos e o zero.

Todos eles voltarão como entraram, com exceção do sinal.

Resumindo a função ABS produz valores de acordo com a regra.

$$\begin{cases} \text{ABS}(X) = X & \text{se } X \geq 0 \\ \text{ABS}(X) = -1 * X & \text{se } X < 0 \end{cases}$$

TRABALHO PARA VOCÊ:

1) O que acontece quando se tem o seguinte programa:

```
10 FOR X=0 TO 25
20 LPRINT "VM: ";TAB(X+4);"13"
30 LPRINT "AM: ";TAB(ABS(3*X-36)+4);"7"
40 NEXT X
50 END
```

2) Será que você é capaz de fazer um programa que permite a saída indicada na Figura 3.18 para a função

$$y = 0,3 x^2 + 2 ?$$

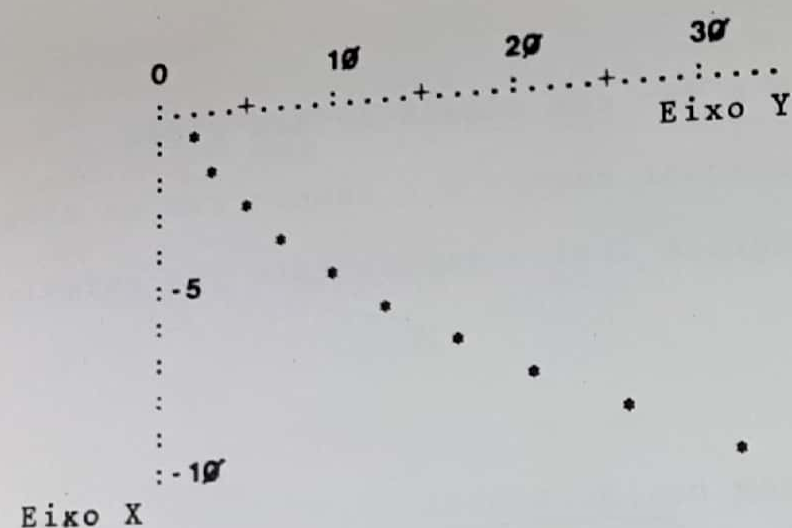


Figura 3.18

Se conseguir continue o problema para as seguintes funções:

- a) $y = 0,2x^2 + x + 3$
- b) $y = 38 - 0,3x^2$
- c) $y = x + 13$
- d) $y = 27 - 2,5x$
- e) $y = |0,28x^2 - 33|$ (isto quer dizer valor absoluto)
- f) $y = |2,8 - 33x|$

Função LOG(X)

A função LOG pode ser usada para encontrar logaritmos naturais ou neperianos, isto é, logaritmos de base $e=2,718282$.

Não se deve confundir este logaritmo com o que se estudou no curso inicial de Álgebra que era de base 10.

O logaritmo comum ou seja aquele de base 10 pode ser encontrado usando a fórmula

$$\log_{10} x = \frac{\log_e x}{\log_e 10}$$

Como se vê não é tão difícil fazer a conversão de uma base para outra!!!

Como se a vida já não fosse suficientemente complicada para você leitor ou leitora agora você deve enfrentar também o sistema de logaritmos o qual como se vê está centrado nos logaritmos naturais.

Importante. O argumento da função LOG precisa ser sempre positivo.

As regras mágicas de conversão são:

- I) Para converter o logaritmo natural para um logaritmo comum multiplique o logaritmo natural por 0,4342945.
- II) Para converter o logaritmo comum para o logaritmo natural multiplique o logaritmo comum por 2,302585.

Exemplo 3.13

Valores do logaritmo natural e comum de um número N.

```
1 REM ** DOMINIO DO LOG **
10 CLS
12 LPRINT
20 INPUT "ENTRE COM UM NÚMERO POSITIVO";N
30 LPRINT
40 LPRINT "O NÚMERO", "LOG NATURAL", "LOG COMUM"
50 LPRINT N, LOG(N), LOG(N)*.4342945
60 LPRINT
70 GOTO 20
RUN
```

Entrando no seu super-micro deve sair algo do tipo:

O NÚMERO	LOG NATURAL	LOG COMUM
14	2.63906	1.14613
O NÚMERO	LOG NATURAL	LOG COMUM
10	2.30259	1

O NÚMERO	LOG NATURAL	LOG COMUM
3	1.09861	.477121
O NÚMERO	LOG NATURAL	LOG COMUM
436	6.07764	2.63949

Função EXP(X)

A função EXP(X) é usada para achar os valores da função e^x onde como já sabemos $e = 2,71828...$

Exemplo 3.14

```
10 INPUT "ENTRE COM UM NÚMERO";N
20 A=EXP(N)
30 LPRINT "2.718282 ELEVADO A ";N;" DA COMO VALOR ";A
40 LPRINT
50 GOTO 10
RUN
```

A saída terá o aspecto:

```
2.718282 ELEVADO A 5 DA COMO VALOR 148.413
2.718282 ELEVADO A 1 DA COMO VALOR 2.71828
2.718282 ELEVADO A 9 DA COMO VALOR 8103.08
```

Cuidado para não entrar com valores muito grandes de N que o seu micro é capaz de "extravasar" e ser incapaz de calculá-los.

Até com os micros pode acontecer isto, viu!!!

DEF

Como se vê várias funções tais como SQR, ABS, LOG já foram introduzidas no computador. Estas são funções predefinidas da linguagem BASIC.

O programador pode entretanto definir outras funções usando a instrução DEF.

Isto é muito útil quando uma mesma fórmula ou expressão é utilizada diversas vezes no mesmo programa.

A forma geral de uma instrução DEF é

`DEF FN < nome da função > (< nome da variável >) =< expressão >`

As palavras DEF FN representam DEFINE FUNCTION.

O nome da função pode ser qualquer nome de variável numérica aceitável como FNA, FNF, FNG7 etc...

O nome da variável ou seja do argumento seguindo o nome da função precisa sempre aparecer dentro dos parênteses e pode ser qualquer variável numérica apropriada.

A declaração de função é uma instrução de especificação e, portanto *não executável*, isto é, nenhum valor resulta da sua interpretação.

Pode portanto, em princípio, ser colocada em qualquer ponto do programa, porém é conveniente colocá-la logo no início do mesmo.

Em relação a função DEF ou seja função *declaração* deve-se dizer que os micros pequenos não possuem a mesma ainda.

Por outro lado já existem sistemas BASIC que permitem o uso de *não apenas um*, mas vários argumentos, embora não seja uma "facilidade" comumente disponível ainda.

Deve-se também citar que algumas implementações de linguagem BASIC permitem a definição de funções declaração não apenas numéricas mas também *alfanuméricas* (STRINGS).

Neste caso, o valor que a função faz voltar é *alfanumérico* e não numérico.

O nome da função deve incluir, no final, o caractere cifrão (\$) e a expressão à direita do sinal de atribuição (=) deve

ser obrigatoriamente uma *expressão alfanumérica*.

É de se esperar que o sistema BASIC que admitir a *função declaração numérica* com mais de um parâmetro, deverá, também admitir a *função declaração alfanumérica* com mais de um parâmetro.

Exemplo 3.15

```
10 DEF FNP(X)=X^2-2*X-1
20 LPRINT "X","FNP(X)"
30 FOR B=1 TO 6
40 LPRINT B,FNP(B)
50 NEXT B
60 INPUT X
70 LPRINT "O RESULTADO DE FNP(";X;") É";FNP(X)
80 END
RUN
```

Se o seu micro aceitar a *função declaração* ter-se-á a saída:

X	FNP(X)
1	-2
2	-1
3	2
4	7
5	14
6	23

? -15

O RESULTADO DE FNP(-10) É 254

Quando a função é calculada na linha 40, a variável dentro dos parênteses é B.

Quando a função é calculada na linha 70, a variável é X.

O nome da variável dentro dos parênteses pode ser o mesmo, bem como diferente do nome da variável usada na instrução DEF.

Deve-se também notar que se a instrução DEF não fosse usada, a fórmula na linha 10 teria que aparecer duas vezes (linhas

40 e 70).

Com isto os resultados da economia de termos usado a *função declaração* são bastante óbvios.

A outra vantagem do uso da instrução DEF é que mudando apenas a primeira instrução podemos usar o resto do programa geralmente com a mesma finalidade.

Aliás, faça isto agora para as funções:

$$\begin{cases} \text{a) } f(x) = x^3 + 2x^2 + 3x + 4 \\ \text{b) } f(x) = 18x^5 - 13x^3 + 441x^2 - 338 \\ \text{c) } f(x) = 14e^x + 338\sqrt{x} + 425 \ln x - 7 \quad \text{para } x > 0 \\ \text{d) } f(x) = 13 e^{-x^2+4x+3} + |x| + \sqrt{x} \end{cases}$$

Exemplo 3.16

```
10 DEF FNA$(A$)="ALÔ QUERIDA ";A$;" UM ABRAÇO!!!"
20 INPUT "NOME=";A$
30 PRINT FNA$(A$)
40 END
RUN
```

Uma saída poderá ser (junto com a conveniente entrada via INPUT)

NOME = ? NILZA MARIA ← Nome teclado
 ↳ Interrogação ou outro sinal devido ao INPUT

A saída será:

ALÔ QUERIDA NILZA MARIA UM ABRAÇO

Como se vê a função declaração alfanumérica FNA\$, fornecido um parâmetro alfanumérico A\$, produz a *concatenação* da constante alfanumérica ALÔ QUERIDA com o valor da variável alfanumérica A\$.

A instrução na linha 20 admite um nome, que será lido para a variável A\$.

A instrução na linha 30 imprime o valor da função FNA\$, com base no valor de A\$.

A concatenação tem uma função muito importante como veremos em alguns programas que virão a seguir.

Exemplo 3.17

```
10 DEF FNG$(A$)=A$+"E AMARELO"
20 B$="O SOL"
30 LPRINT FNG$(B$);"E";FNG$("MELÃO GRANDE");"."
40 END
RUN
```

Precisa do LET?

A saída será:

O SOL É AMARELO E MELÃO GRANDE É AMARELO

Deve-se observar com cuidado a saída produzida pela linha 30.

FNG\$() é calculada duas vezes, primeiro com o "SOL" e depois com "MELÃO GRANDE".

PEQUENO TRABALHO PARA VOCÊ ECLÉTICO LEITOR OU LEITORA

1) Entre com dois números X e Y.

Imprima a quantidade $F(Y) - F(X)$ dado que

$$F(Z) = 9Z^3 - 15Z^2 + 13Z + 4Z - 14$$

2) Qual é a saída exata do seguinte programa

```
10 DEF FN F(X) = 3*X - 6
20 FOR Y = -4 TO 6 STEP 2
30 IF FNF(Y) > 0 THEN PRINT "FNF(;"Y;" ) E POSITIVA "
40 IF FNF(Y) = 0 THEN PRINT "FNF(;"Y;" ) E ZERO "
50 IF FNF(Y) < 0 THEN PRINT "FNF(;"Y;" ) E NEGATIVA "
60 NEXT Y
70 END
RUN
```



Figura 3.19

§3.5 - AS FUNÇÕES TRIGONOMÉTRICAS

Em princípio Trigonometria lida com triângulos, seus ângulos e as relações entre os comprimentos dos seus lados.

No triângulo da Figura 3.20 o seno (abreviatura usada SIN) do ângulo A é definido como a razão (que é o que se obtém após a divisão) do comprimento do lado a para o comprimento do lado c.

$$\text{SIN } A = \frac{a}{c}$$

$$\text{COS } A = \frac{b}{c}$$

$$\text{TAN } A = \frac{a}{b}$$

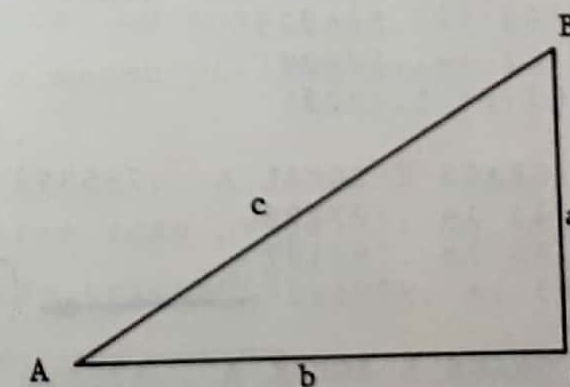


Figura 3.20

Dessas relações podemos achar qualquer razão se conhecermos o ângulo correspondente.

O micro está apto a achar os valores de várias funções trigonométricas.

As funções SIN(X), COS(X) ou TAN(X) nos darão assim o valor do seno, do cosseno ou da tangente do ângulo X, onde o X é medido em radianos.

Para converter um ângulo de graus para radianos basta multiplicar por 3,14159265 e então dividir o resultado por 180 ($180^\circ = \pi$ radianos e $\pi = 3,14159265$).

Exemplo 3.18

Este programa testa a habilidade trigonométrica do seu micro.

```
10 INPUT "VALOR";A
20 X=A*3.14159/180
30 LPRINT A;" GRAUS É IGUAL A ";X;"RADIANOS"
40 LPRINT "SEN(";A;")=";SIN(X)
50 LPRINT "COS(";A;")=";COS(X)
60 LPRINT "TG(";A;")=";TAN(X)
70 LPRINT
80 GOTO 10
RUN
```

Uma possível saída seria:

```
60 GRAUS É IGUAL A 1.0472 RADIANOS
SEN( 60 )= .866025
COS( 60 )= .500001
TG( 60 )= 1.73205
```

```
45 GRAUS É IGUAL A .785398 RADIANOS
SEN( 45 )= .707106
COS( 45 )= .707107
TG( 45 )= .999999
```

```
30 GRAUS É IGUAL A .523598 RADIANOS
SEN( 30 )= .5
COS( 30 )= .866026
TG( 30 )= .57735
```



Será que o seu micro não sabe que $\text{tg } 45^\circ$ é 1,00000?

VALOR? → Onde está ele, você já cansou!?

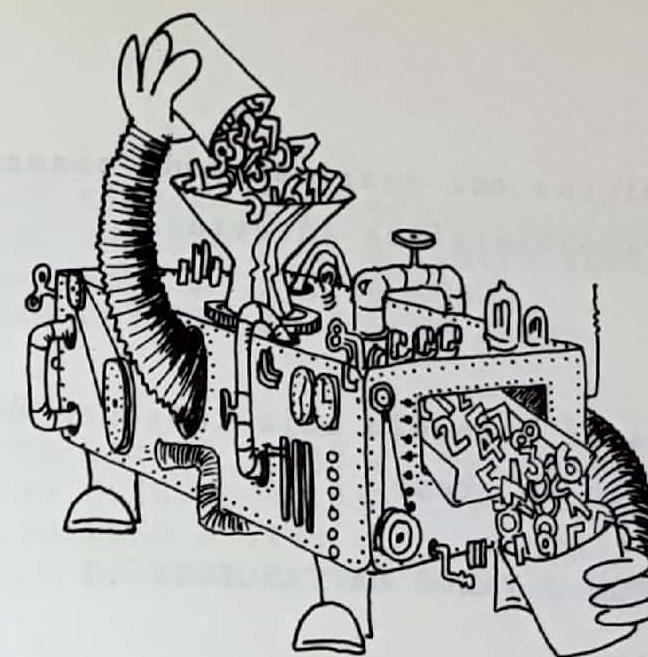


Figura 3.21 - "O mundo movido a base de valores vindo das funções SIN, COS, LOG, EXP, TAN, ATN, SGN."

Aqueles que lembram "de cabeça" alguns valores trigonométricos sabem que $\text{tg } 45^\circ$ é igual a 1, infelizmente alguns micros calculam as funções trigonométricas com alguma aproximação.

O micro faz o melhor que ele pode, pois ele é HONESTO!!!

Há micros que possuem apenas uma função trigonométrica inversa que é a arcotangente (ATN) no trecho de $-\frac{\pi}{2}$ a $+\frac{\pi}{2}$ que é por isto chamado valor principal.

Não esqueça que para converter um ângulo de radianos para graus deve-se multiplicar o mesmo por 180 e então dividir o resultado por 3,14159265.

Para achar por exemplo o arco seno principal de um número é necessário usar a identidade trigonométrica

$$\text{arc sen}(x) = \text{arctg} \frac{x}{\sqrt{1-x^2}} \quad \text{onde } -1 < x < 1$$

Já para calcular o arco cosseno basta usar a identidade trigonométrica

$$\text{arc cos}(x) = \text{arctg} \sqrt{\frac{1-x^2}{x}} \quad \begin{array}{l} \text{para } -1 \leq x \leq 1 \\ \text{para } x \neq 0 \end{array}$$

Porém já temos micros nos quais se pode chamar diretamente estas funções trigonométricas inversas.

Exemplo 3.19

Este programa nos dá o ângulo cuja tangente é conhecida e imprime o resultado em graus.

```
10 INPUT "ENTRE COM O VALOR DA TANGENTE";T
20 R=ATN(T)
30 D=R*180/3.14159
40 PRINT "O ÂNGULO CUJA TANGENTE É";T;"E";D;"GRAUS"
50 PRINT : GOTO 10
RUN
```

Uma eventual conversação com o micro nos dará:

```
ENTRE COM O VALOR DA TANGENTE ? 1.2
O ÂNGULO CUJA TANGENTE É 1,2 E 50.194429 GRAUS
```

```
ENTRE COM O VALOR DA TANGENTE ? .57735
O ÂNGULO CUJA TANGENTE É .57735 E 29.99998 GRAUS
```

```
ENTRE COM UM VALOR ?
BREAK NA LINHA 10
```

Exemplo 3.20

Neste programa obtemos o valor do arcosseno e do arcocosseno de X tanto em radianos como em graus.

```
10 INPUT "VALOR"; X
20 S = ATN(X/SQR(1-X**2))
30 S1 = S*180/3.14159
40 C = ATN(SQR(1-X**2))/X
50 C1 = C*180/3.14159
60 PRINT "O ÂNGULO CUJO SENOS É";X;"E";S;"RADIANOS".
70 PRINT "O ÂNGULO CUJO COSSENOS É";X;"E";C;"RADIANOS".
80 PRINT "O ÂNGULO CUJO SENOS É";X;"E";S1;"GRAUS".
90 PRINT "O ÂNGULO CUJO COSSENOS É";X;"E";C1;"GRAUS".
100 PRINT : GOTO 10
RUN
```

Precisa do LET o seu micro?
Não, precisa

Uma eventual saída pode ser:

```
VALOR ? .5
O ÂNGULO CUJO SENOS É .5 E .523599 RADIANOS
O ÂNGULO CUJO COSSENOS É .5 E 1.0472 RADIANOS
O ÂNGULO CUJO SENOS É .5 E 30 GRAUS
O ÂNGULO CUJO COSSENOS É .5 E 60 GRAUS
```

```
VALOR ? .933
O ÂNGULO CUJO SENOS E .933 E 1.20660 RADIANOS
O ÂNGULO CUJO COSSENOS E .933 E 3.6813 RADIANOS
O ÂNGULO CUJO SENOS E .933 E 68.90 GRAUS
O ÂNGULO CUJO COSSENOS E .933 E 21.09 GRAUS
```

VALOR ?

.....

PEDE PARA CONTINUAR E AQUI É "HORA" DO

BREAK

NA LINHA 10

Exemplo 3.21

O programa que aqui apresentamos calcula a área aproximada da curva $y = \sin x$ de $x=0,5$ até $x=2,5$.

Para se obter isto, a área total é dividida em 100 faixas e a área aproximada de cada faixa é calculada.

Aqueles que fizerem (ou já fizeram) Engenharia vão lembrar que este é o método dos retângulos.

```
10 LET T=0
20 LET R=2.5-.5
30 LET W=R/100
40 LET M=W/2
50 LET X=.5+M
60 LET C=1
70 IF C>100 THEN 120
80 LET A=W*SIN(X)
90 LET T=T+A
100 LET C=C+1
110 LET X=X+W
120 GOTO 70
130 LPRINT "A ÁREA É IGUAL A";T
140 END
RUN
```

A saída é:

A ÁREA É IGUAL A 1.67875

192

Como se vê neste programa a largura de cada faixa é W (veja a Figura 3.22).

A altura de cada faixa é dada pelo $\text{SIN}(X)$ e a mesma é medi-
da no ponto médio de cada intervalo (veja os números de li-
nha 40 e 50 onde M representa a metade de cada intervalo e
X representa o valor da abscissa na qual se calcula a orde-
nada ou seja a altura).

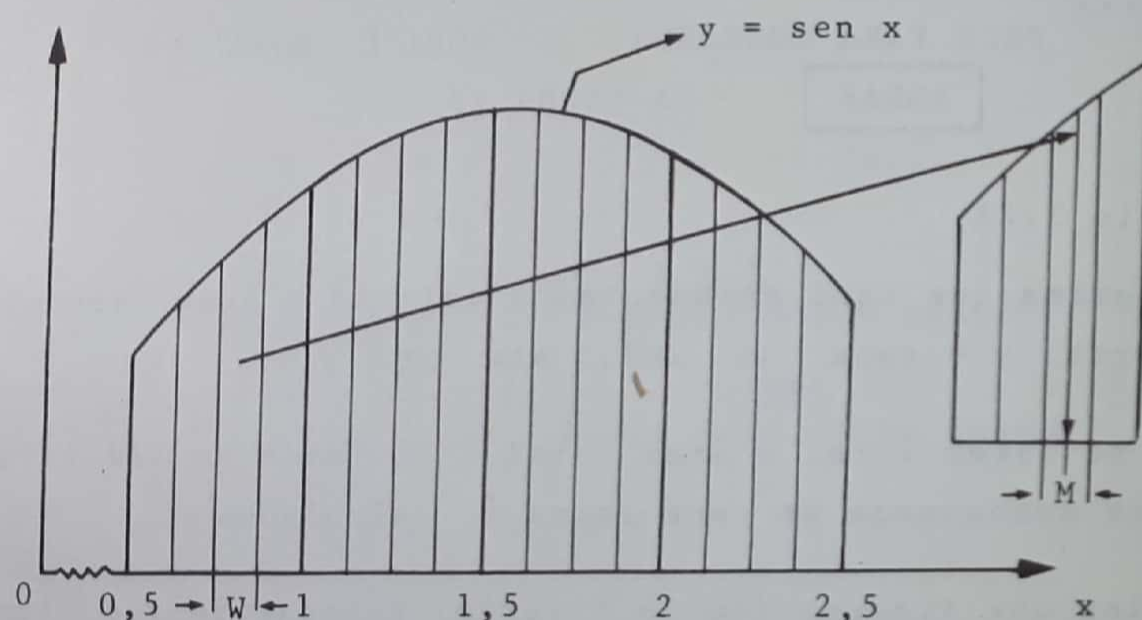


Figura 3.22 - Obtenção da área limitada pela função $y = \text{sen} x$ de 0,5 a 2,5 pelo método da integração numérica - método dos retângulos.

Esta área *não é a exata* e como pequena *distração* verificado
ra pede-se ao expedito leitor ou leitora que aumente a pre-
cisão (já que na realidade um lado do nosso retângulo não é
bem uma reta) tomando agora 200 "faixas" e depois 400.

Qual é a resposta em cada caso?

No caso de termos 200 "faixas" a área é 1,6787|33 e no caso de termos 400 tiras ou "retângulos aproximados" a área é 1,6787|27.

Como se vê a melhora é na 5ª casa decimal e em alguns casos isto não serve para nada.

Exemplo 3.22

Utilizar a função seno para imprimir SACHA KILOBYTE.

Solução:

```
10 FOR X=0 TO 6.2 STEP .2
20 LET Y=SIN(X)
30 LPRINT TAB(20*Y+20); "SACHA KILOBYTE"
40 NEXT X
50 END
RUN
```

Na saída teremos:

[illegible]

Observação fundamental:

Na linha 30 faça as seguintes modificações (uma de cada vez)

```

1) LET Y = COS(X)
2) LET Y = SIN(1.5*X) + COS(X)
3) LET Y = COS(2.1*X) + SIN(0.9*X)

```

Que maravilhosos desenhos você obteve!!!

Agora a Trigonometria está domada?

Não, ainda não, vem aí mais explicações e aplicações e aí quem sabe...

Deve-se tomar muito cuidado com os gráficos.

Se tivermos uma relação entre x e y como dado na Tabela 3.6 e

x	y
-3	9
-2	4
-1	1
0	0
1	1
2	4
3	9

Tabela 3.6

se não indicarmos corretamente qual é a equação que "une" y com x podemos ter um dos seguintes gráficos (veja a Figura 3.23).

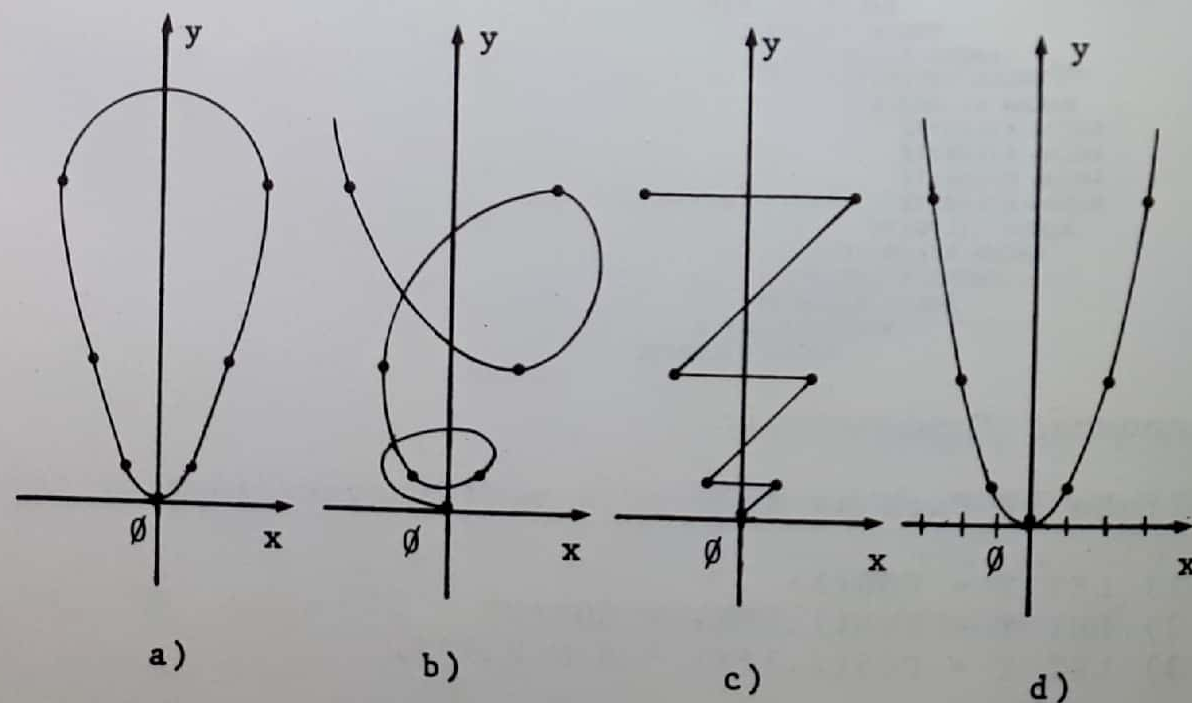


Figura 3.23 - "Dúvidas gráficas"

Se dissermos entretanto que a dependência entre y e x é dada por $y=x^2$ resulta correta apenas a representação gráfica indicada na Figura 3.23 d).

Muito cuidado então com as expressões matemáticas e principalmente com as escalas para não "distorcer" as representações gráficas.

Exemplo 3.23

Aqui vai um superprograma para você repetir e implementar.

```

10 DIM A$(61), X(99), Y(99)
20 K=0
30 J=36
40 FOR A=0 TO 6.2 STEP .1
50 K=K+1
60 X(K)=COS(A)-.5*COS(2*A)-.5
70 Y(K)=SIN(A)-.5*SIN(2*A)
80 REM ACHAR MÍNIMO E MÁXIMO
90 IF K<>1 THEN 150
100 X7=X(1)
110 X1=X(1)
120 Y7=Y(1)
130 Y1=Y(1)
140 GOTO 230
150 IF X7 >= X(K) THEN 170
160 X7=X(K)
170 IF X1 < X(K) THEN 190
180 X1=X(K)
190 IF Y7 >= Y(K) THEN 210
200 Y7=Y(K)
210 IF Y1 < Y(K) THEN 230
220 Y1=Y(K)
230 NEXT A
240 FOR I=1 TO K
250 Y(I)=Y(I)-Y1
260 X(I)=X(I)-X1
270 NEXT I
280 S1=59/(X7-X1)
290 S2=36/(Y7-Y1)
300 A1=36-INT(.5+Y7*S2)
310 A2=INT(1.5-X1*S1)
320 A$(61)="I"
330 FOR I=1 TO 6
340 V=X1+I*10/S1
350 LPRINT TAB(I*10+6);
360 GOSUB 1000
370 NEXT I

```



```

380 LPRINT
390 FOR I=1 TO 6
400 LPRINT TAB(I*10+10);"#";
410 NEXT I
420 LPRINT
440 FOR I=1 TO 60
460 NEXT I
470 B$=B$+"I"
480 LPRINT B$
490 FOR I=1 TO 60
500 IF I<>A2 THEN 530
510 A$(I)="I"
520 GOTO 540
530 A$(I)=" "
540 NEXT I
550 IF J<>A1 THEN 600
560 FOR I=1 TO 60
570 A$(I)="-"
580 NEXT I
590 A$(A2)="+ "
600 FOR I=1 TO K
610 IF Y(I)<=(J-.5)/S2 THEN 650
620 M=INT(X(I)*S1+1.5)
630 A$(M)="*"
640 Y(I)=-Y(I)
650 NEXT I
660 V=Y1+J/S2
670 GOSUB 1000
680 LPRINT TAB(10);"I";
690 FOR I=1 TO 61
700 LPRINT A$(I);
710 NEXT I
720 LPRINT
730 J=J-1
740 IF J>=0 THEN 490
750 LPRINT B$
760 GOTO 9999
1000 REM SUBROTINA-A JÓIA DOS GRÁFICOS-BELEZA DAS ESCALAS
1010 IF V<>0 THEN 1040
1020 LPRINT V;
1030 GOTO 1100
1040 E=INT(LOG(ABS(V))/2.303+.5)
1050 IF E>2 THEN 1090
1060 IF E<-2 THEN 1090
1070 LPRINT INT(V*10↑(2-E)+.5)/10*(2-E);
1080 GOTO 1100
1090 LPRINT INT(V*10↑(1-E)+.5)/10;"E";E;
1100 RETURN
9999 END
RUN

```

Antes de obter a saída uma *OBSERVAÇÃO IMPORTANTE*.

Ao executar este *programa* você obterá uma linda curva que se chama "cardiôide" e que você deve guardar para na próxima oportunidade enviar ao seu namorado ou sua namorada.

Isto pode ser no dia 12 de junho, no aniversário dele (de-la) ou para si mesmo em sinal de intensa auto-satisfação computacional...

E lá vai a linda "saída" que você vai obter:

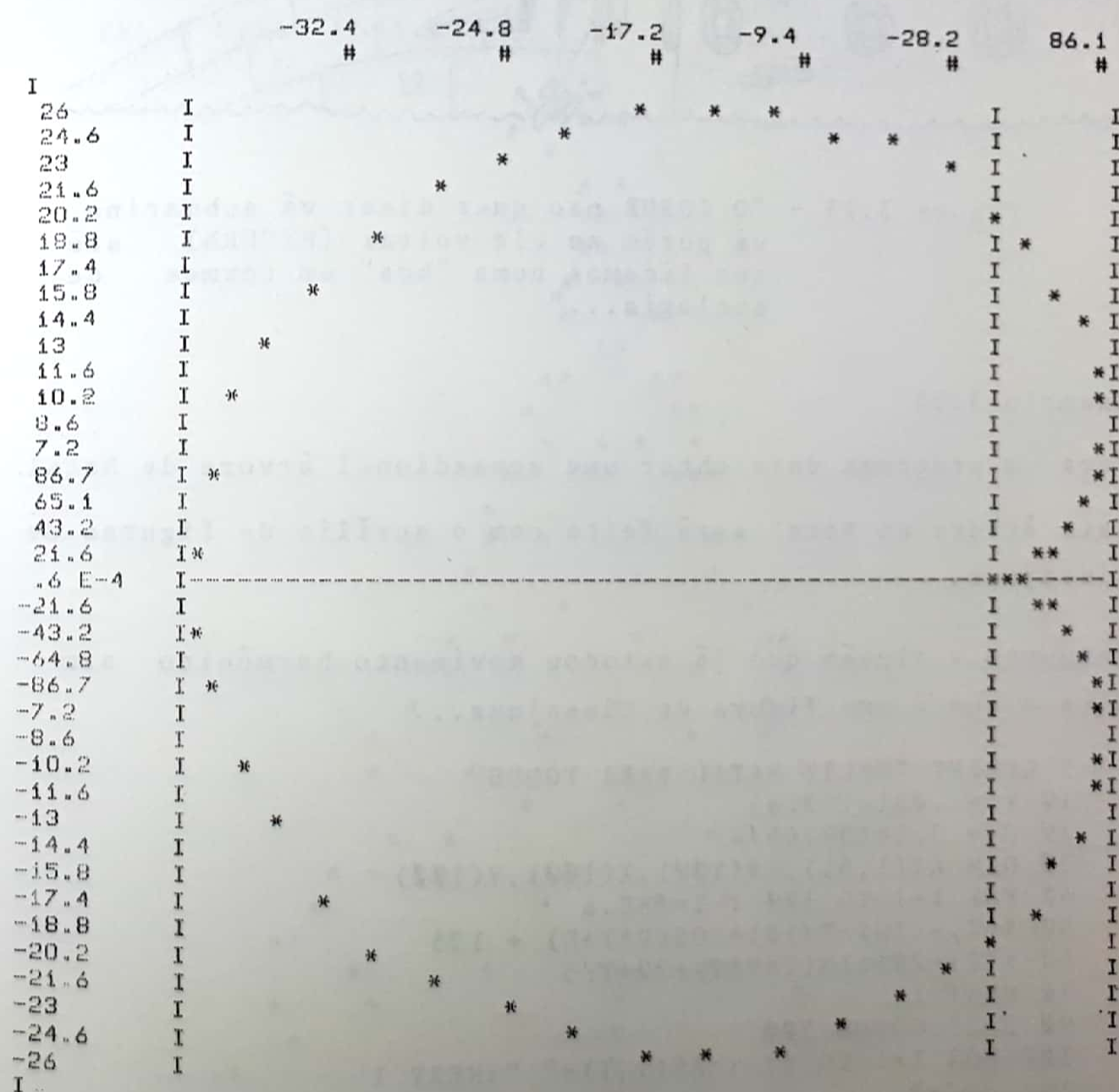


Figura 3.24 - "E viva a Trigonometria, o micro, os lindos gráficos, as coordenadas polares ... Realmente é uma satisfação dominar o BASIC".

LEMBRANÇA - VOCÊ AINDA LEMBRA O QUE É A INSTRUÇÃO GOSUB JUNTO COM O RETURN?

A participação das mesmas no Exemplo 3.23 é fundamental.

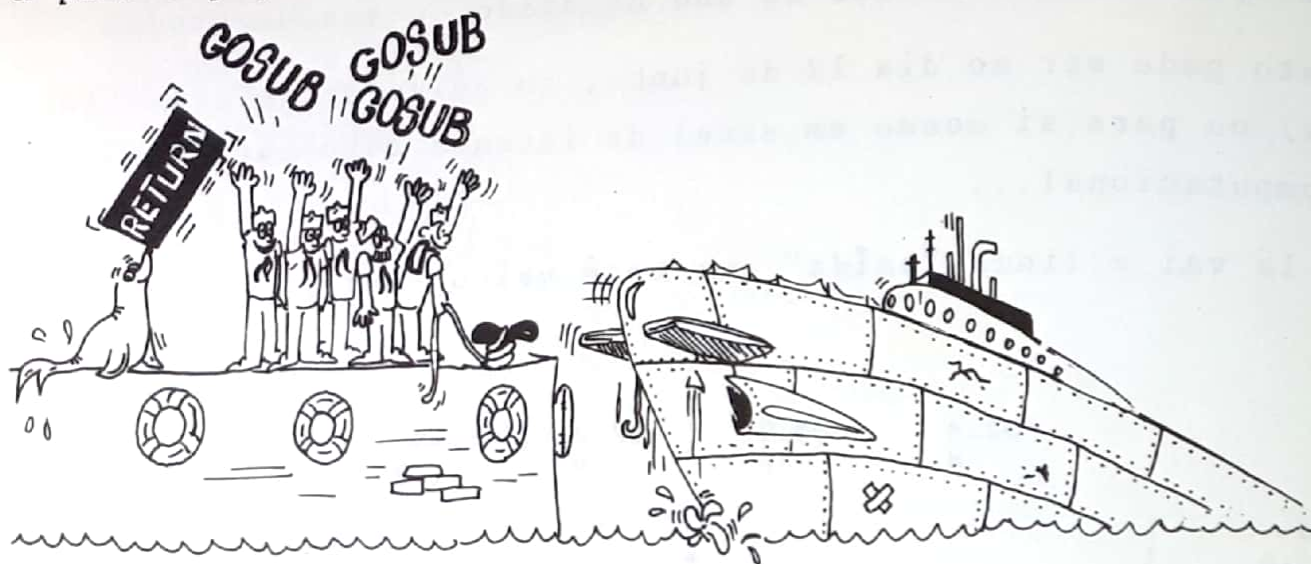


Figura 3.25 - "O GOSUB não quer dizer vá submarino, vá porém se ele voltar (RETURN) até que ficamos numa "boa" em termos de analogia..."

Exemplo 3.24

Faça um programa para obter uma sensacional árvore de Natal.

Esta árvore de Natal será feita com o auxílio de figuras de Lissajous.

Pergunte a alguém que já estudou movimento harmônico simples o que é uma figura de Lissajous...

```

5 LPRINT "FELIZ NATAL PARA TODOS"
10 V = .0314159265
20 D = 3.14159265/4
30 DIM A$(1,61),Z$(100),X(100),Y(100)
40 FOR I=1 TO 100 : T=I*8.6
50 X(I)=(100-T/10)*COS(V*T+D) + 125
60 Y(I)=20*SIN(2*V*T)+20+T/5
70 NEXT I
90 J=29:GOSUB 300
100 FOR I=1 TO 61 : A$(1,I)=" ":NEXT I
107 J=J-1
110 FOR I=1 TO 100
120 IF Y(I)>(J+.5)*256/36 THEN 200
130 NEXT I

```

```

137 IF J<6 THEN A$(1,30)="V"
140 FOR I=1 TO 61:LPRINT A$(1,I);:NEXT I: LPRINT
150 IF J<1 THEN 1000
160 GOTO 100
200 K=INT(X(I)/256*60+.5): A$(1,K)="*":Y(I)=Y(I)-300
220 GOTO 130
300 LPRINT TAB(28) "*"
301 LPRINT TAB(27) "* *"
302 LPRINT TAB(25) "*"      "*"
303 LPRINT TAB(27) "* *"
304 LPRINT TAB(28) "*"
305 RETURN
1000 END
RUN

```

Na saída ter-se-á:

FELIZ NATAL PARA TODOS

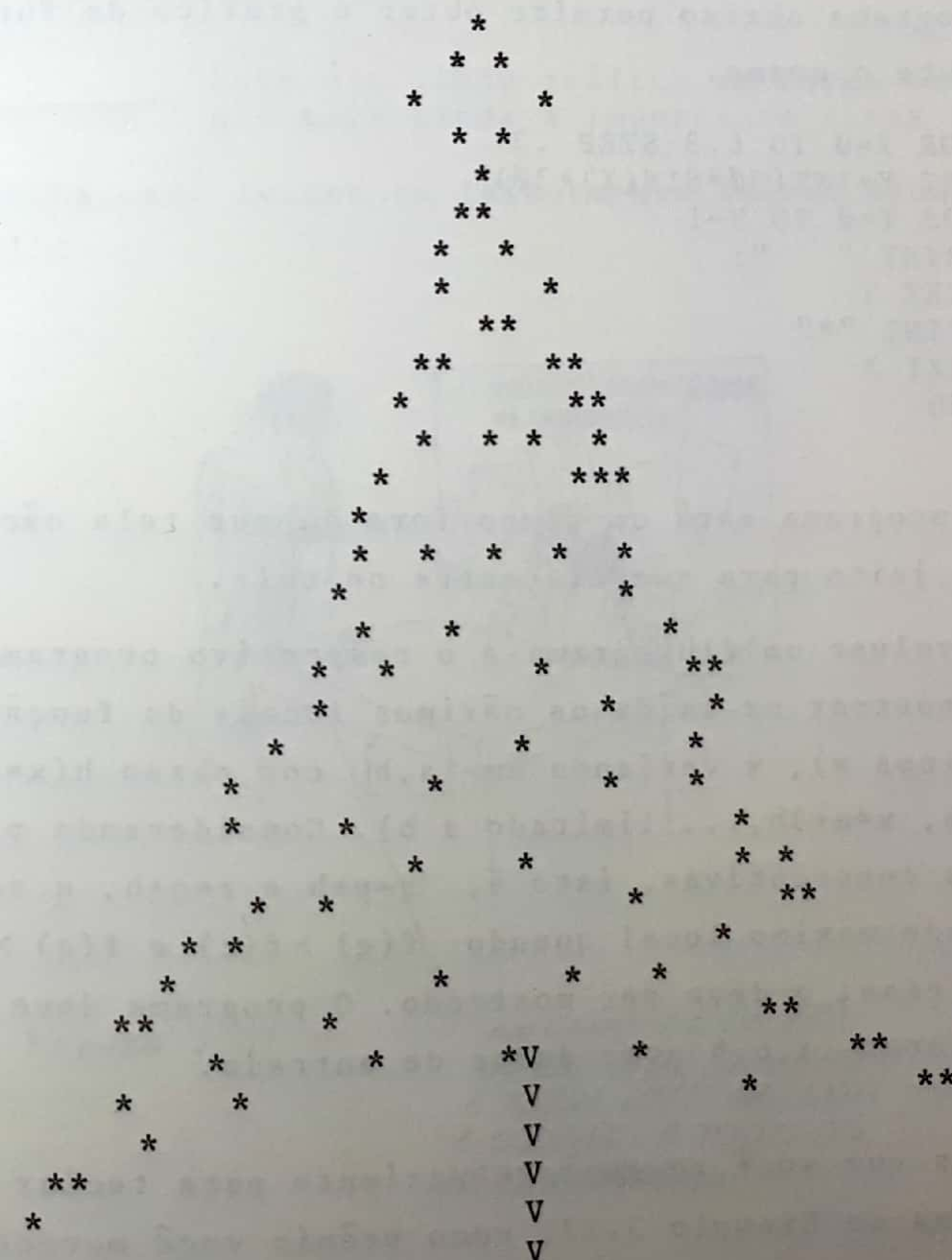


Figura 3.26

TRABALHO PARA VOCÊ

- 1) Escreva um programa que determine qual é o maior valor $\sin(2,6)$, $\cos(1,75)$ ou $\ln(8,4)$ e imprima a resposta.

Como uma "dica" você pode supor que os três valores são diferentes entre si.

- 2) Escreva um programa que calcule a área aproximada abaixo da curva $y=\sin x$ desde $x=0,4$ até $x=2,4$.

Utilize obrigatoriamente o laço FOR-NEXT.

- 3) O programa abaixo permite obter o gráfico da função seno. Execute o mesmo.

```
10 FOR X=0 TO 6.8 STEP .2
20 LET Y=INT(30*SIN(X)+30)
30 FOR T=0 TO Y-1
40 PRINT " ";
50 NEXT T
60 PRINT "*"
70 NEXT X
80 END
RUN
```

Este programa está um pouco fora da sua tela não é?
Dê um jeito para que ele entre na tela.

- 4) Desenvolver um fluxograma e o respectivo programa BASIC para mostrar na saída os máximos locais da função $f(x) = x(1+\cos x)$, x variando em $[a,b]$ com passo $h(x=a, x=a+h, x=a+2h, x=a+3h, \dots$ limitado a b). Considerando p, q, r abscissas consecutivas, isto é, $q=p+h$ e $r=q+h$, q será considerado máximo local quando $f(q) > f(r)$ e $f(q) > f(p)$; nesse caso, q deve ser mostrado. O programa deve aceitar os valores a, b, h como dados de entrada.

- 5) Uma vez que você se manteve paciente para teclar todo programa do Exemplo 3.23, como prêmio você merece outras "figurinhas" e para tanto basta que você substitua as instruções nas linhas de número 40, 60 e 70 pelos seguintes conjuntos de novas instruções:

- a)

```
40 FOR A=0 TO 6.2 STEP .1
60 LET X(K)=SIN(2*A)+COS(A)
70 LET Y(K)=COS(2*A)+SIN(A)
```
- b)

```
40 FOR A=-2.9 TO 2.9 STEP .075
60 LET X(K)=A
70 LET Y(K)=A*A*(A*A-8)+10
```
- c)

```
40 FOR A=0 TO 6.2 STEP .1
60 LET X(K)=COS(A)**3
70 LET Y(K)=SIN(A)**3
```
- d)

```
35 LET PI=3.141592
40 FOR A=-4.9 TO 5 STEP .1
60 LET X(K)=A
70 LET Y(K)=1/(2*PI)*EXP(-A**2/2)
```

Este é o lindo gráfico em forma de sino da não mais linda e importante curva de Gauss

Paciência caro leitor ou leitora que sairão lindas curvinhas!!!!



Figura 3.27 -

"... CHAME-ME DE 'TRIG'
OU ENTÃO DE 'SENINHO'
E ASSIM VOCÊ PASSARÁ
A CONHECER TODOS OS
MEUS ÂNGULOS..."

O basic avançado

§4.1 - O PRINT USING

De todas as formas que nos permitem imprimir (PRINT) a mais poderosa (mas também a mais confusa) chama-se *PRINT USING*.

O nome PRINT USING, ele próprio implica que queremos imprimir (PRINT) algo usando (USING) alguma coisa.

Beleza de explicação!!!

E a implicação está certa.

Como foi originalmente desenvolvido para uso em grandes computadores, PRINT USING é constituído de duas partes - *PRINT* e *USING*.

Imprime-se (PRINT) usando (USING) o formato (chamado "imagem") que se acha na linha.

Desta forma a instrução PRINT USING permite que uma linha completa de saída seja formatada em zonas de comprimento variável.

Essas zonas podem conter uma série de números, STRINGs ou uma combinação de ambos.

A forma geral do PRINT USING é:

PRINT USING "< formato >"; < variáveis ou expressões >

O formato pode ser uma variável STRING ou um STRING dentro das aspas.

A sua finalidade é informar o micro sobre o formato que se deseja empregar na impressão de variáveis ou expressões.

As variáveis ou expressões precisam ser precedidas por ponto e vírgula e separadas uma da outra por vírgulas.

Lá vai um PRINT USING com números.

Exemplo 4.1

```
10 A = 177.457333
30 B$ = "###.##"
40 PRINT USING B$;A
50 PRINT : LIST
RUN
```

A resposta impressa será

177.46

que como se vê foi arredondada e impressa com precisão de 2 casas decimais.

Exemplo 4.2

Se acrescentarmos no Exemplo 4.1 as instruções

```
15 B=1.7
45 PRINT USING B$;B
```

e teclarmos RUN para o programa todo, no vídeo surgirá:

177.46
1.70

Fatos importantes para serem lembrados:

- 1) B\$ "foi chamado" duas vezes;
- 2) os números aparecem com os seus pontos decimais alinhados;

3) no caso da variável B foi acrescentado um zero

Exemplo 4.3

Se adicionarmos ao Exemplo 4.1 completo (isto é com o que foi adicionado no Exemplo 4.2) as instruções

```
20 C = 7788.4476
48 PRINT USING B$;C      o que ocorre?
```

Dando o comando RUN teremos a saída

177.46
1.70
%7788.45

O que aconteceu?

Bem, o símbolo % (pode ser que não seja assim no seu micro) indica que excedemos a capacidade dos símbolos de imagem para imprimir dígitos à esquerda do ponto decimal, mas de qualquer forma o número foi impresso.

É bem melhor perder o alinhamento do ponto decimal (o que realmente ocorreu) do que perder números importantes à esquerda do ponto decimal.

Porém o nosso servil micro chamou a atenção para esta "falha".

Aliás a mesma pode ser facilmente corrigida colocando mais um sinal # para se ter uma posição a mais para o dígito extra.

Com isto estamos adicionando um novo elemento para o campo na linha da imagem.

Faça então:

```
30 B$ = "####.##"
```

e repita o programa do Exemplo 4.3.

Não saiu tudo "bonitinho" e alinhado?

Ótimo.

Exemplo 4.4

Veja agora o que ocorre se fizermos na linha 30 do Exemplo 4.3.

```
30 B$ = "$###.##"
```

Na saída ter-se-á:

```
$ 177.46
```

```
$ 1.70
```

```
$7788.45
```

Lindo não! Como se vê o sinal de "dolar" ou seja o cifrão (\$) sai alinhado ou seja na mesma coluna.

Exemplo 4.5

Será que vão ocorrer problemas se modificarmos a instrução 30 para a seguinte forma:

```
30 B$ = "$$###.##"
```

Teremos a seguinte saída:

```
$ 177.46
```

```
$1.70
```

```
$ 7788.45
```

Como se vê este formato é extremamente útil para escrever "certas quantias" em cheques.

CONCLUSÕES:

- 1) PRINT USING com # (cancela) imprime o ponto decimal (na notação matemática no Brasil corresponde a vírgula) ou ponto fracionário no mesmo lugar para qualquer tamanho de número impresso.

- 2) O PRINT USING arredonda para cima todos os centavos ou seja os números à direita do ponto decimal mas não mexe *nunca* nos cruzeiros ou nos dolares (isto já seria demais) que são os números escritos à esquerda do ponto decimal.

Com o PRINT USING recebemos a mensagem de erro com a "bandeira" % isto quando o tamanho do número é maior do que o campo na linha imagem e com isto o ponto decimal é deslocado para a direita.

- 3) Se apenas um \$ (cifrão) é acrescentado a esquerda o mesmo será impresso e alinhado em coluna como já ocorreu com o ponto decimal.

O cifrão (\$) simples não expande o campo.

- 4) Se dois cifrões (\$\$) forem colocados a esquerda, um cifrão \$ ocupará a posição imediatamente anterior ao dígito de mais alta ordem e será impresso em cada linha.

Um desses cifrões (\$) pode substituir um # no campo, sem entretanto expandi-lo.

Quando se utiliza uma impressora para imprimir cheques é conveniente tomar precauções extras contra eventuais "alterações".

Isto pode ser obtido facilmente através da mudança na linha 30 do Exemplo 4.5.

Exemplo 4.6

```
10 A=177.457333
15 B=1.7
20 C=7788.4476
30 B$ = "***###.##"
40 LPRINT USING B$;A
50 LPRINT USING B$;B
60 LPRINT USING B$;C
70 LPRINT:LLIST
RUN
```

Teremos na saída:


```

**177.46
***1.70
*7788.45

```

Conclusão importante: Proteção ao cheque

Um ou mais asteriscos (*) começando na primeira posição ou segunda posição farão com que sejam substituídos todos os zeros de mais alta ordem ou ainda em outras palavras são preenchidos todos os espaços não usados para números porém perde-se o cifrão (\$).

Porém os micros possuem recursos de edição para não perder este sinal tão importante (\$).

Exemplo 4.7

O que ocorre no Exemplo 4.6 se agora na linha 30 tivermos:

```
30 B$="**$###.##" ?
```

Resposta:

Na saída deve-se ter:

```

*$177.46
***$1.70
$7788.46

```

Exemplo 4.8

Agora se você quiser realmente impressionar alguém com o tamanho dos seus números coloque muitos sinais (#) na linha imagem.

Mude a linha 30 para

```
30 B$ = "**$#####.##".
```

no Exemplo 4.7 e veja o que sai.

Resposta:

Ter-se-á:

```

*****$177.46
*****$1.70
*****$7788.46

```

Como poderíamos separar os números à esquerda do ponto decimal?

A resposta é: usando a vírgula numérica (na notação matemática clássica no Brasil é o ponto numérico).

Uma vírgula aparecerá na posição fixa onde estiver o símbolo, entre os dígitos, ou um branco se os zeros à esquerda forem suprimidos.

Exemplo 4.9

O que ocorre no Exemplo 4.7 se na linha 30 tivermos

```
30 B$ = "**$,###.##" ?
```

Resposta:

Ter-se-á na saída:

```

***$177.46
*****$1.70
*$7,788.46

```

Note que apenas um dos três números tem mais de três dígitos à esquerda do ponto decimal e é neste que apareceu a vírgula para separar os dois 7 (sete) para uma leitura mais simples.

No campo imagem, a vírgula pode ser colocada em qualquer lugar entre o cifrão (\$) e o ponto decimal e somente uma vírgula é necessária para automaticamente inserir vírgulas à esquerda de cada 3 dígitos à esquerda do ponto decimal.

Vamos agora brincar um pouco com STRINGS dentro do PRINT USING.

Exemplo 4.10

Analisemos o programa

```
1 CLS:LPRINT
10 A$=" HOJE ESTÁ"
20 B$="CRUZ"
30 C$="CREDO"
40 D$=" FRIO PRA CACHORRO"
50 U$="%      %"
60 LPRINT USING U$;A$
70 LPRINT:LLIST
RUN
```

A única coisa estranha até agora é a variável alfanumérica U\$="% %" com sete espaços em branco entre %.

Já não basta o uso de % para porcentagem!

Ele apareceu de novo. Por quê?

Resposta:

Quando se usa %< n-2· espaços >% na saída teremos a reserva para os n primeiros caracteres de um STRING. É óbvio neste caso, que se tivermos %, reservamos espaço para os 2 primeiros caracteres de uma variável alfanumérica (STRING). Portanto o % é um símbolo que é para STRINGS é a mesma coisa que o # (cancela) para os números.

No nosso caso será exibido na tela do vídeo

HOJE ESTÁ

Mude, como o uso de um pouco de sua massa cinzenta, a linha 50, para que saia na tela.

HOJE ESTÁ CRUZ CREDO FRIO PRA CACHORRO



Figura 4.1 - "O uso do PRINT USING impede que "malandros" alterem os nossos cheques ou ainda que nos passem cantadas bancárias".

Mais trabalho para você usando obrigatoriamente o PRINT USING.

- 1) Elabore um programa que tenha um cabeçalho e que emita uma nota do seguinte tipo

Artigo	Valor	Desconto	Total
Microcomputador B. Anana	183000	0,15	155550
Adaptador	3490	0,10	3141
Disquete	48350	0,18	39647

Tabela 4.1

- 2) Elabore um programa que tenha um cabeçalho que emita um relatório dos artigos vendidos e além do mais dê como conclusão a venda total por dia.

O seu programa deve ter uma saída do tipo

CÓDIGO	DESCRIÇÃO	CUSTO	QUANT.	TOTAL
:	%	%	###	***\$,###.##

Suponha que como artigos você está vendendo no seu Micro Banana Shop disquetes, cabos para UCP, jogos, micros, impressoras, vídeos gravadores, etc.

Epa, epa, mas você ainda não sabe o que significa (!) dentro do PRINT USING então descanse um pouco do "trabalho" e aprenda o que vem a seguir!!!



Figura 4.2 - "Fique até altas horas da madrugada no seu micro mas domine o PRINT USING"

Vamos partir para o 2º "round" com o PRINT USING.

Nos já aprendemos alguma coisa para fazer o PRINT USING "trabalhar".

Mas existem mais alguns "truques".

Em primeiro lugar vejamos o que ocorre se tivermos dentro do PRINT USING um ponto de exclamação (!)

A melhor forma de perceber isto é analisando o programa do Exemplo 4.11.

Exemplo 4.11

```

1Ø CLS:LPRINT
2Ø X$="ALEXANDRE"
3Ø Y$="SERGIO"
4Ø Z$="VICTOR"
5Ø A$="!!%      %"
6Ø LPRINT USING A$;X$;Y$;Z$
7Ø LPRINT:LLIST
RUN

```

Na saída ter-se-á:

ASVICTOR

Como se vê o ponto de exclamação (!) serve para reservar um elemento no campo para a PRIMEIRA LETRA da variável STRING atribuída a ela.

Isto é muito "jeitoso" principalmente se você quiser colocar as iniciais e o último nome de uma lista de convidados para a festa do seu aniversário.

O que deveríamos fazer no Exemplo 4.11 para se ter na saída:

ALE~~SE~~SE~~VI~~VI?

Resposta:

Basta na linha 50 termos:

$50 \text{ A\$} = \% \overbrace{\% \% \% \% \% \% \% \% }^{2 \text{ espaços em branco}} \% \%$
 $\underbrace{\hspace{1cm}}_{1 \text{ espaço em branco}}$

Podemos ainda ter na especificação do formato ou imagem qua
tro setas (↑↑↑↑) para cima.

Neste caso teremos na saída um número na notação científica.

Deve-se não esquecer que $\uparrow$ é impresso como $(\uparrow)$ na tela do vídeo ou então como $(\bar{O})$.

Então um novo aspecto que pode aparecer no formato de conversão de um dado numérico é:

A\$ = "# #. # # # # ↑ ↑ ↑ ↑"

Observação importante:

O símbolo # que alguns chamam de "cancela" pode também ser chamado de símbolo de *libra* (pound).

É óbvio que também dentro da especificação do formato que pode ser usada na instrução PRINT USING podem aparecer os sinais (+) e (-) e dessa forma podemos indicar ganho ou perda.

O sinal (+) pode ser colocado *antes* ou *depois* do campo #.

Como um destaque final do PRINT USING vejamos a possibilidade de incorporar a linha imagem dentro da linha do PRINT USING.

Isto requer um certo cuidado e tem um valor fundamental quando somente poucas variáveis devem ser impressas ou melhor devem ser impressas uma vez apenas pois geralmente nas aplicações práticas, a linha imagem é referenciada muitas vezes durante a execução e comumente por diferentes linhas de PRINT USING.

Exemplo 4.12

Esta é uma análise especial do PRINT USING

```

1 CLS:LPRINT
10 X$="ALEXANDRE"
20 Y$="SERGIO"
30 Z$="VICTOR"
50 N$="SÃO MEUS FILHOS"
60 LPRINT USING "Z$;Y$;X$;N$";
100 LPRINT:LLIST
RUN

```

Diagrama de formatação para a linha 60:

- Z\$: 7 espaços em branco
- Y\$: 4 espaços em branco
- X\$: 13 espaços em branco
- N\$: 1 espaço em branco

Na saída ter-se-á:

ALEXANDRE SERGIO VICTOR SÃO MEUS FILHOS

Como se vê em relação ao Exemplo 4.11 nos "despachamos" a variável alfanumérica A\$ e incorporamos seus elementos em uma linha imagem embutida no PRINT USING separada das variáveis por um ponto e vírgula.

Isto economiza espaço e para aplicações não muito complexas do PRINT USING tem muito valor.

Recomendamos entretanto para aplicações mais complicadas que a imagem seja mantida em linha separada da linha do PRINT USING.

Observação importante:

E agora é realmente o FIM do PRINT USING.

Tem micro que "entende" o PRINT USING da seguinte forma

Exemplo 4.13

```

10 LET X=3.145692
20 PRINT USING 30,X
30 :#.##
40 END
RUN

```

Hi, apareceu de novo o LET E precisa?



Na saída ter-se-á:

3.15

Explicação: O micro faz exibir na tela do vídeo o valor de X como foi "formatado" pelas linhas 20 e 30.

O sinal de libra (#) ou você gasta mais do "cancela" indica como anteriormente o número de casas decimais.

Quanto ao arredondamento segue tudo como antes no quartel de Abrantes...

Exemplo 4.14

```

10 LET X=35.47
20 LET Y=25.34
30 LET Z=X*Y
40 PRINT USING 50, X,Y,Z
50: ### ## ####
60 END
RUN

```

(será que aqui vai ↑ ou ^)?



Teremos na saída:

```

35.47      25.34      1.8771 E 39

```

Podemos ainda neste tipo de PRINT USING usar um apóstrofo (') seguido de L, R ou E.

Exemplo 4.15

```

10 LET A$= "BATA TODOS OS DADOS"
20 PRINT USING 30, A$
30: 'LLLLLLL
40 END
RUN

```

Teremos na saída

```
BATA TOD
```

Como se vê um apóstrofo (') seguido de letras ou caracteres L (*left*) faz com que a cadeia de caracteres seja alinhada à esquerda (sem levar em conta o espaço em branco).

Se a cadeia é maior que o campo, será alinhada à esquerda e truncada à direita.

Se tivermos por outro lado um apóstrofo (') seguido de uma cadeia de símbolos R (*right*) a cadeia de caracteres é alinhada a direita.

Caso a cadeia seja maior que o campo será alinhada à direita e truncada a esquerda.

Já com um apóstrofo (') seguido de caracteres ou letras C (*center*) a cadeia de caracteres alfanuméricos é centraliza-

da no campo.

Caso não puder ser exatamente centralizada, será deslocada uma posição para a esquerda.

Se a cadeia é maior que o campo, será alinhada à esquerda e truncada a direita.

Com o E (*extended*) após o apóstrofo (') consegue-se estender a cadeia de caracteres.

Exemplo 4.16

```

10 LET A$="A MELHOR COISA É JOGAR NO CENTRO"
20 PRINT USING 30, A$
30: 'CCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCCC
40 END
RUN

```

```
A MELHOR COISA É JOGAR NO CENTRO
```

TRABALHO FINAL PARA VOCÊ SOBRE O PRINT USING.

1) Veja o que sai no seu micro para os seguintes programinhas

```

a) 10 LET B$="GOSTO DE COMER SORVETE"
20 LET C$="NÃO VOU FAZER ISTO"
30 PRINT USING 40, B$, C$
40: 'LLLLLLLLL 'RRRRRR
50 END
RUN

```

```

2) 10 LET X=39.47
20 LET Y= .0000000003587
30 LET Z=X+Y
40 PRINT USING 50, X,Y,Z
50: ##.##↑↑↑↑ ##.##↑↑↑↑ ##.##↑↑↑↑
60 END
RUN

```




Figura 4.3

"EU PENSO QUE DESCOBRI PORQUE SOMOS TÃO POPULARES"... O SENHOR ANTONIO DISTRIBUIR, NOSSO ESPECIALISTA EM COMPUTAÇÃO, JAMAIS CONSEGUIU DOMINAR A INSTRUÇÃO **PRINT USING**..."

§4.2 - O CONJUNTO ASCII

O intuito deste parágrafo é ensinar e você aprender o uso do código ASC e da função CHR \$.

ASCII são as iniciais de American Standard Code for Information Interchange.

Desde que um micro armazena e processa somente números, não letras ou pontuação, é importante que exista alguma forma uniforme para especificar que números representam que letras e símbolos.

Para saber exatamente esta relação recomendamos ao caro leitor ou leitora que execute o seguinte programa.

Exemplo 4.17

```
1 REM C$=CARACTERE QUE ENTRA
2 REM A=VALOR DE C$ NO CÓDIGO ASCII
3 REM K$="STRING" NOVO
4 REM S$=INDICADOR
10 INPUT "QUALQUER CARACTERE";C$
20 A=ASC(C$)
30 PRINT "O VALOR ASCII DE ";C$;" E";A
40 K$=STRING$(13,A)
50 PRINT"";CHR$(A);" REPETIDO 13 VEZES VAI APARECER COMO:";
60 PRINT K$
70 INPUT "VOCÊ GOSTARIA DE CONTINUAR O PROGRAMA?";S$
80 IF ASC(S$)=83 THEN 10
90 END
RUN
```

A saída seria:

```
QUALQUER CARACTERE? M
O VALOR ASCII DE M E 77
M REPETIDO 13 VEZES VAI APARECER COMO: MMMMMMMMMMMMM
VOCÊ GOSTARIA DE CONTINUAR O PROGRAMA? N
```

Explicações sobre o Exemplo 4.17

- 1) X=ASC(P\$) → Converte apenas o primeiro caractere da variável alfanumérica P\$ para o seu número de código ASCII.
- 2) P\$=CHR\$(X) → Atribui a P\$ um caractere com o número de código de X; P\$ terá assim apenas um caractere.
- 3) P\$=STRING\$(X,Y) → Gera um STRING, P\$ de comprimento X composto de caracteres todos tendo como número de código ASCII o Y.

Um pequeno problema para você

- 1) Entre com um caractere simples através do teclado e teste o seu valor ASCII para determinar se o mesmo é número.

Caso não seja volte o controle do programa à instrução INPUT.

Sugestão: Utilize duas instruções IF e ASC e o seguinte trecho do código ASCII.

Código decimal	48	49	50	51	52	53	54	55	56	57
Caractere	0	1	2	3	4	5	6	7	8	9

Tabela 4.2

§4.3 - OUTRAS FUNÇÕES AVANÇADAS

1 - A medição de STRING ou seja de variáveis alfanuméricas

Uma das coisas que mais se quer de uma informação produzida em uma variável alfanumérica é o seu *comprimento*.

Parece inacreditável que comprimento é tão importante. Mas é.

Em alguns casos a falta de comprimento também é importante...

Bem felizmente temos no BASIC a função LEN.

Definição:

`L = LEN(A$)`

Atribui a L um valor igual ao número de caracteres na variável STRING A\$ incluindo os espaços em branco.

Para sentir o grande efeito do LEN passe o seguinte programa:

Exemplo 4.18

```
1 CLS : PRINT
10 INPUT "ENTRE COM UMA SEQUÊNCIA DE CARACTERES";A$
20 L=LEN(A$)
30 PRINT A$; "TEM";L;"CARACTERES"
100 PRINT:LIST
```

```
110 END
RUN
```

Execute várias vezes entrando com o seu nome com o do melhor jogador de basquete do momento, com o sobrenome do autor do livro (para prestigiá-lo e para treinar sua memória), com o nome da sua "namoradinha", com combinações de letras e números etc.

Vão acontecer estranhos "cucumbers" (veja no seu dicionário Inglês-Português o que é isso) caso as linhas 10 e 20 do Exemplo 4.18 fossem mudadas para

```
10 INPUT "ENTRA UM NÚMERO";A
20 L=LEN(A)
```

O que aconteceu?

Bem o seu micro indicou erro e é óbvio pois você tentou entrar com um número dentro do LEN o qual requer a entrada de um alfanumérico.

Bem mas você ainda pode tentar consertar isto deixando

```
10 INPUT "ENTRA UM NÚMERO";A
```

e fazendo

```
20 L=LEN("A")
```

Se fizer o resto do programa e teclar o RUN você verá na saída que temos apenas um caractere.

A resposta é que o LEN avalia o comprimento do que esta atualmente entre parênteses ou aspas.

Antes (no Exemplo 4.18) nos trouxemos um STRING de fora e medimos o seu comprimento.

Tudo funcionou lindamente.

Aliás o nome da sua namoradinha é comprido demais e você deve rapidamente arrumar algo mais curto e mais afetivo ou então troque de namorada.

Esta pode ser uma vantagem especial do LEN !!!

Agora nós estamos medindo o comprimento daquilo que está entre aspas e este comprimento não muda com o seu valor ou seja da variável A!!!

Note bem que ao se aplicar a função LEN se incluem também a vírgula (se aparecer) e os espaços em branco.

2 - Concatenação

Concatenação?

Concatenação???

O que será que significa?

Nos já falamos um pouco sobre concatenação no Exemplo 3.16. Tem muito dicionário que não tem o significado de concatenação.

Podemos entender como seu sinônimo encadeamento, seriação ou entrosamento.

Porém no caso da linguagem BASIC vai significar somar ou adicionar variáveis alfanuméricas ou STRING.

Exemplo 4.19

```
1 CLS PRINT
10 FOR N=1 TO 7
15 PRINT A$
20 READ A$
25 B$=B$+A$
30 PRINT B$
40 NEXT N
90 PRINT:LIST
100 DATA NILZA,MARIA,ALEXANDRE,SERGIO,VICTOR JR,VICTOR,
    MIRSHAWKA
110 END
RUN
```



O seu micro precisa saber quem é B\$? Faça então B\$="" que no caso afirmativo significa nada.

Deve-se ter na saída

NILZA
NILZA MARIA

.....
NILZA MARIA MIRSHAWKA

Como se vê passando pelos vários ciclos vai se adicionando e guardando tudo em B\$ que cresce comumente até a capacidade de 50 bytes.

Acima disto vai ocorrer um "estouro"....

Ressaltando mais uma vez na linha-chave 25 termos a nova variável B\$ que é igual a antiga variável B\$ (que começa com nada) mais aquilo que está em A\$ e foi lido (READ).

Como na saída B\$ é cada vez maior isto pode trazer sérios problemas a ponto de não termos espaço na memória e neste caso convém usar o comando CLEAR para aumentar o espaço para a variável alfanumérica B\$.

Se você vai agora homenagear toda a sua família que sairá impressa do avo até o neto (que é você) é bom antes (digamos na linha 5) escrever

5 CLEAR 150

o que leva B\$ a ter espaço agora para 150 caracteres.



Figura 4.4

TRABALHO PARA VOCE

- 1) Se o seu micro aceita a concatenação use-a obrigatoriamente para obter o histograma da Figura 4.5.

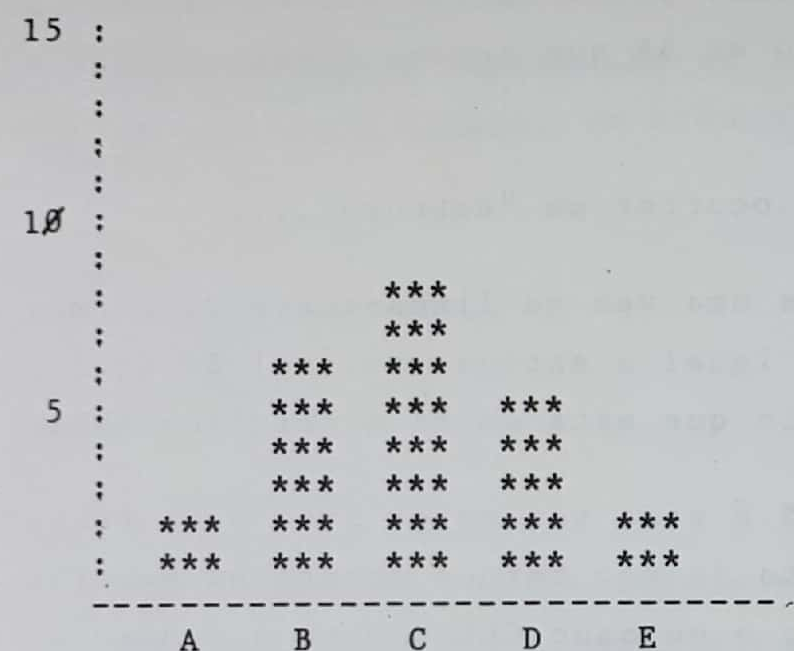


Figura 4.5

3 - Funções VAL(\$) e STR\$(N)

Definição:

V=VAL(A\$)

Converte o primeiro conjunto de caracteres numéricos que estão na variável alfanumérica (há quem chame também de alfabética apenas) A\$ em um valor numérico que é atribuído a V.

Os caracteres não numéricos são ignorados.

Dessa forma a função VAL converte a variável STRING retendo o número como um número, se o número estiver no começo da variável STRING.

O melhor para perceber isto é analisar o seguinte exemplo:

Exemplo 4.20

```

10 LET A$="765.4"
20 PRINT VAL(A$)
30 LET B$=" 765.4 "
40 PRINT VAL(B$)
50 LET C$=" 76 5.4 "
60 PRINT VAL(C$)
70 END
RUN

```

→ Espaços em branco

Após a execução ter-se-á:

```

765.4
765.4
765.4

```

Deve-se notar que os espaços em branco foram ignorados ao se aplicar a função VAL.

Exemplo 4.21

```

100 LET V$="13"
120 PRINT V$
140 PRINT VAL(V$)
150 END
RUN

```

E o resultado será:

```

13
13

```

Observe o deslocamento para a direita de VAL(V\$) com o espaço reservado para a colocação de sinal algébrico.

Exemplo 4.22

```

20 PRINT VAL("6400")
30 PRINT VAL("-2897")
40 END
RUN

```

O resultado será:

6400
-2897

e como você percebe os números saem alinhados.



Figura 4.6 - "É para não esquecer do VAL e do STR..."

Definição: $S\$ = \text{STR}\(X)

Converte X em STRING de caracteres numéricos e o atribui a S\$.

Em outras palavras converte um número, fornecido como seu argumento em uma fileira (STRING) de caracteres alfanuméricos.

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

A soma de duas STRINGS (fileiras) é uma fileira que se obtém por junção das duas (você não esqueceu ainda o que é CONCATENAÇÃO).

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

É claro que se quisermos voltar a poder operar com a fileira (STRING) como número usa-se a função VAL.

Exemplo 4.23

```
10 LET A=713
20 LET B=317
30 PRINT STR$(A)+STR$(B)
40 LET C=VAL(STR$(A)+STR$(B))
50 PRINT C
60 END
RUN
```

O seu micro precisa do LET?

Na saída ter-se-á:

713 317
713317

→ Espaço em branco

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

Definição: $M\$ = \text{MID}\$(A\$,N1,N2)$

Atribui a M\$ uma subfileira (sub STRING) de uma variável STRING A\$, começando com o caracter N1 e tendo um comprimento a partir daí de N2 caracteres.

Definição: $M\$ = \text{MID}\$(A\$,N)$

Atribui a M\$ uma sub STRING da variável STRING A\$ a partir do N-ésimo caractere até a posição mais a direita da STRING.

Definição: $R\$ = \text{RIGHT}\$(A\$,N)$

Atribui a R\$ uma subfileira (subSTRING) da variável A\$ consistindo dos N caracteres mais a direita de A\$.

Definição: $L\$ = \text{LEFT}\$(A\$,N)$

Atribui a L\$ uma subfileira (subSTRING) da variável alfanumérica A\$ a partir da esquerda até incluir o N-ésimo caractere.

Exemplo 4.24

```

10 CLS:PRINT
20 S$="PITUCA ESPEROU VOCÊ"
30 PRINT LEFT$(S$,6),
40 PRINT MID$(S$,8,7),
50 PRINT RIGHT$(S$,4),
60 PRINT:LIST
70 END
RUN

```

Espaço em branco

Na tela do vídeo ter-se-á:

PITUCA ESPEROU VOCÊ

Que mancada hein, bicho!!! Não se deve fazer o sexo fraco esperar.

Isto se aplica se você for homem...

Nas linhas 30 e 40 do Exemplo 4.24 colocamos as vírgulas com o intuito de que tudo fosse impresso na mesma linha.

Veja o que ocorre com o seu programa se você no lugar da linha 40 tiver a linha 25.

```
25 PRINT MID$(S$,8,7),
```

Resposta:

Deve aparecer

ESPEROU VOCÊ PITUCA

Voltando ao comentário que apareceu no Exemplo 4.24 a pergunta agora é

Será que foi mancada sua quanto a espera?

Finalmente para evitar problemas de consciência repita o Exemplo 4.24 fazendo a linha 50 ser a linha 30 e a 30 ser a 50.

O que sairá?

Resposta: VOCÊ ESPEROU PITUCA

Hi,hi,hi!!! O micro (só falta, o ponto de interrogação) está querendo dizer que o culpado foi você.

Definição: $I\$ = \text{INKEY}\$$

Atribui a I\$ um caractere correspondente a um símbolo do teclado de acordo com a combinação de teclas que estiver sendo digitada ou calcada caso contrário atribui nada ou seja o vazio.

Em outras palavras a função INKEY\$ permite ler caracteres do teclado, caso alguma tecla esteja calcada, no exato momento em que a função é executada.

O atento leitor ou leitora perguntará para que existe uma função de ler teclado se já existe a instrução de leitura INPUT.

Resposta:

Existe uma diferença básica entre as duas operações.

Enquanto a instrução INPUT, ao ser executada, mantém a execução do programa suspensa até que um dado tenha sido fornecido pelo teclado a função INKEY\$, quando é executada não espera o fornecimento de dados.

A função INKEY\$ lê a tecla que for apertada, se houver alguma, e, se não houver, gera um caractere vazio.

A função INKEY\$ varre o teclado, independente de haver ou não alguma tecla sendo calcada.

Portanto a função INKEY\$ é muito poderosa e nos capacita a entrar (INPUT) com informações diretamente do teclado sem precisar usar a tecla ENTER.

Exemplo 4.25

```
1 CLS
7 IF INKEY$="V" THEN 13
12 GOTO 7
13 PRINT"VOCÊ TECLOU A LETRA 'V'"
20 GOTO 7
RUN
```

Comentário sobre este programa

O teclado parece morto até que você aperte a letra V, isto é você tecla e não acontece nada.

Passa-se então no teste da linha 7 e a execução se move para a linha 13 e a mensagem é impressa.

Daí para frente o processo recomeça...

Desde que a função INKEY\$ pode somente "fotografar" uma letra ou um número de cada vez, se você quiser fazer o teste para mais de um caractere você precisa escrever um programa para testar cada um na sequência.

Se a função INKEY\$ varrer o teclado e não encontrar nenhuma tecla calcada (que é o caso comum) "recebe" a ordem para ler um "STRING nulo".

Um STRING nulo é representado por duas marcas de aspas não contendo nada entre elas.

O código ASC II para nulo ou nada é 0.

Observação importante:

Não queremos discutir o sexo do STRING e nem deixá-lo em

situação delicada porém quando estiver implícito que estamos falando em variável ou fileira então será feminino e quando nos referirmos a um conjunto alfanumérico ou simplesmente a um nome poderá ser masculino. O importante com a ou o na frente é que você saiba o que é STRING.

Além do mais STRING é estrangeiro...

Exemplo 4.26

Vejamos como se pode "capturar" 13 caracteres diferentes com a função INKEY\$.

```
10 DIM V$(13,1)
20 PRINT"DIGITE 13 CARACTERES"
25 PRINT"PARA INICIAR TECLE A"
30 INPUT A$
40 IF A$<>"A" THEN GOTO 30
50 FOR X=1 TO 13
60 IF INKEY$ <>" " THEN GOTO 60
70 IF INKEY$=" " THEN GOTO 70
80 LET V$(X)=INKEY$
90 NEXT X
100 PRINT"CARACTERES DIGITADOS"
110 FOR Y=1 TO 13
120 PRINT V$(Y)
130 NEXT Y
140 END
RUN
```

Comentários:

O leitor ou leitora já observou neste caso que a linha 60 serve para manter a execução do programa se tiver sido pressionada alguma tecla.

A linha 70 por sua vez retém a execução do programa até que seja pressionada uma tecla.

Esta tecla será "capturada" pela função INKEY\$ da instrução LET, na linha 80 e armazenado na matriz dos caracteres V\$.

Este ciclo se repete 13 vezes até que 13 caracteres tenham sido "acumulados".

O caractere *branco* não pode ser capturado pela função `INKEY$` pois a correspondente tecla (`SPACE` na maior parte dos ^{mi}cros) durante a execução do programa, gera um comando `BREAK`, que fará parar a execução.

Caso se queira que o branco se inclua entre os caracteres exibíveis, é necessário relacioná-lo com outra tecla através da programação.

Definição: T\$ = TIME\$

Atribuí a `T$` um `STRING` representando a data atual e a correspondente época do calendário.

Vamos voltar ao Exemplo 4.26, aquele de entrar com um `STRING` tendo 13 caracteres sem usar a tecla `ENTER`, e fazer com o mesmo algumas manipulações para dar a ilustração final do uso dessas funções mais avançadas.

Exemplo 4.27

```

1 REM I$=ENTRADA DO STRING
2 REM T$=E O STRING ALVO
3 REM ESTA PARTE DO PROGRAMA VISA A FUNÇÃO INKEY$
10 I$=INKEY$
20 IF I$="" THEN 10
30 T$=T$+I$
40 PRINT I$;
50 IF LEN(T$) < 13 THEN 10
51 REM ESTA PARTE DO PROGRAMA COMEÇA A MANIPULAÇÃO
52 PRINT
50 PRINT"";T$;"' DO FIM PARA O COMEÇO E:";
61 FOR V=1 TO 100:NEXT V
70 FOR X=13 TO 1 STEP -1
80 PRINT MID$(T$,X,1)
81 FOR V=1 TO 100:NEXT V
90 NEXT X
91 FOR V=1 TO 100:NEXT V
100 PRINT "O VALOR DE '";T$;"' E: ";VAL(T$)
101 FOR V=1 TO 100:NEXT V
110 PRINT "";T$;"PARTIDO E MODIFICADO E: ";
111 FOR V=1 TO 100: NEXT V
120 PRINT RIGHT$(T$,6);LEFT$(T$,7)
121 FOR V=1 TO 100: NEXT V
130 PRINT"";VAL(T$);"'DO FIM PARA O COMEÇO E:";
131 FOR V=1 TO 100: NEXT V

```

Já esqueceu a CONCATENAÇÃO?



```

140 FOR X=LEN(STR$(VAL(T$))) TO 1 STEP -1
150 PRINT MID$(STR$(VAL(T$)),X,1);
151 FOR V=1 TO 100: NEXT V
160 NEXT X
170 END
RUN

```

Uma possível saída pode ser:

```

13713 NILZAMIR
'13713 NILZAMIR' DO FIM PARA O COMEÇO E: RIMAZLIN31713
O VALOR DE '13713 NILZAMIR' E: 13713
'13713 NILZAMIR' PARTIDO E MODIFICADO E: ZAMIR13713NI
'13713' DO FIM PARA O COMEÇO E: 31713

```

101010101010111011-
1010111010110101101-
11100011101010100-
110111010101101010-

0101011110101-
010110011101-
010101101011-
010101110101-



Figura 4.7 - Conversa em ASCII

PARA TREINAR BEM ESTAS FUNÇÕES AVANÇADAS SÓ TEM UM JEITO -
É PRATICANDO
NÃO VALEM IMPROPÉRIOS - MÃOS A OBRA

- 1) Escreva um programa no qual você entra com o seu nome e então o micro imprime o número do código ASCII para cada letra do seu nome.
- 2) Usando números convenientemente escolhidos do código ASCII (veja a página 212 do nosso primeiro livro -Linguagem BASIC ou consulte de algum outro autor e concorrente "amigo") na linha do `DATA`, imprima a frase "ASCII FAZ ISTO!"

3) Entre com o STRING

TRES! @ # \$ % FUNÇÕES! @ # \$ % STRING

Faça uso das funções LEFT\$, MID\$ e RIGHT\$ para imprimir a frase

"TRÊS FUNÇÕES STRING"

4) Entre com um STRING A\$ e use um laço (loop) para imprimir o número ASCII de cada um dos seus caracteres.

5) Entre com as palavras "BARBARELA", "BELA" e "CABIDELA". Imprima cada uma destas palavras tirando das mesmas o B (no código ASCII o número do B é 66).

6) Suponha que A\$ consiste das treze primeiras letras do alfabeto.

Usando A\$, construa o triângulo retângulo como mostrado na Figura 4.8.

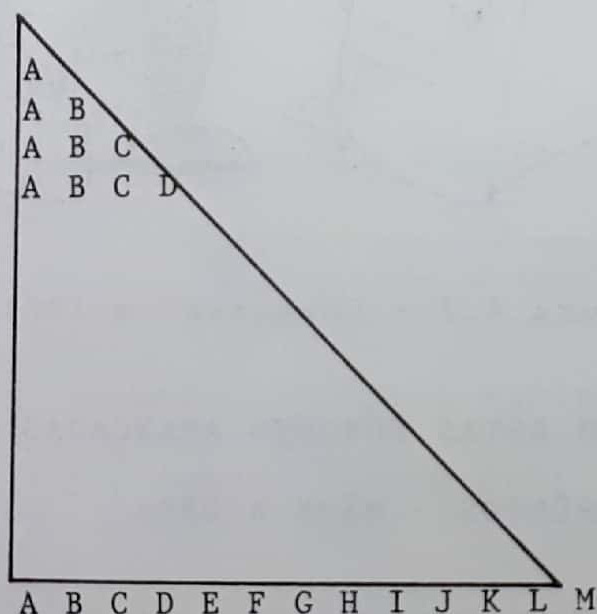


Figura 4.8

7) Escolha aleatoriamente 15 números entre 65 a 90 inclusive, para serem usados como números no código ASCII. Converta esses quinze inteiros nos quinze caracteres que eles representam e imprima os resultados.

8) Entre com o seu endereço, isto é nome da rua e número (Alameda dos Tacaunas 694).

Use a função VAL para extrair o número da sua casa e acrescente mais 7 para obter o número da casa do seu vizinho amigo.

9) Escreva um programa usando a função STR\$ para imprimir 30 artigos que estão no estoque de uma loja e que aparecem com as etiquetas

101 VM, 102 VM, ..., 130 VM.

Parece bem natural que aqui se deve usar o laço FOR-NEXT.

10) Escreva um programa que faça a pergunta "O SEU MICRO BANANA NÃO É ESPERTO"

Entre com uma resposta SIM ou NÃO.

Se o 1º caractere da resposta é S, imprima "AFIRMATIVO"

Se o 1º caractere é um N, imprima "NEGATIVO".

Em caso contrário imprima

"ESTA É UMA PERGUNTA DE SIM OU NÃO"

e mande o controle de volta para a instrução INPUT.

11) Leia (use a instrução READ) os seguintes alfanuméricos, V204 VM, A306 VM e S414 VM.

Utilize a função MID\$ para obter só a parte numérica desses dados alfanuméricos e armazene os mesmos em um STRING (caracteres que estão da posição 2 até a 4).

Relate qual dos STRINGS dados tem dentro de si o maior número.

12) Utilizando números aleatórios e os números do código ASCII de 32 a 126 inclusive faça o micro gerar um STRING de 80 caracteres admitindo repetições.

Tabule quantos caracteres são letras, números e caracteres genéricos.

Imprima os resultados.

13) Você é um espião que irá utilizar o micro para produzir um código secreto.

a) Entre com uma mensagem curta e faça o micro imprimir o que parece uma sequência de palavras sem sentido. Para produzir palavras codificadas, converta cada letra da mensagem original no correspondente número de código ASCII adicione 3 a cada número e converta para caracteres para produzir a mensagem.

Mantenha todos os espaços entre as palavras nos seus lugares originais e "compreenda" que as letras X, Y e Z devem ser convertidas em A, B e C.

b) Escreva um programa que decodifique a mensagem

ANCPQRTKUNNGVVAASSNM

Tem sentido em alguma língua?

Escreva você uma mensagem que tenha sentido em Portugues.

14) Entre com um inteiro positivo N\$ como se ele fosse expresso na forma binária, isto é entre com um STRING de "uns" e "zeros".

Faça o micro imprimir o seu equivalente N\$ no sistema decimal.

Ainda bem que a tarefinha terminou!!!

Não é?



Figura 4.9 - "Será que os humanos conversarão algum dia assim, entre si?"

Exemplo 4.28

A teoria matemática dos jogos é explicada de forma mais simples em termos do que se chama *teoria dos jogos de soma zero* (se quiser se aprofundar no futuro veja o nosso livro de pesquisa Operacional vol. I).

pode-se descrever este jogo em termos de uma matriz quadrada que mostra todos os possíveis "ganhos" (Tabela 4.3).

Temos no nosso caso particular um jogo matricial baseado em uma versão de um jogo chamado "Morra".

Chamemos os jogadores de Victor e "Cheio de byte" (os quais se traduzirão como *Você versus Computador* usando as variáveis V e C).

A "matriz dos ganhos" é:

		Número de dedos	
		2	3
"Cheio de byte"	2	+4	-5
	3	-5	+9

Tabela 4.3

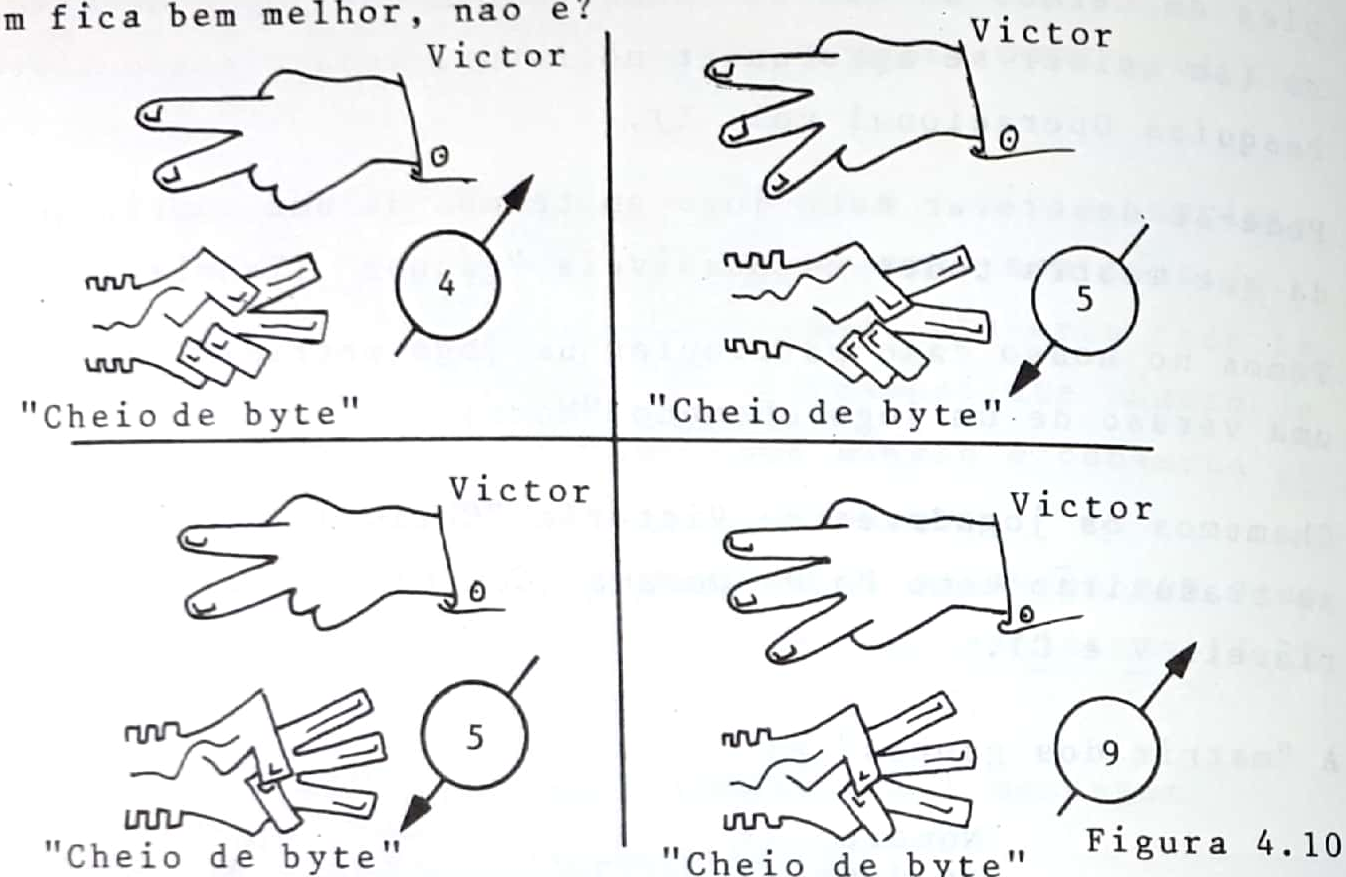
Assim se o número de dedos é par "Cheio de byte" paga a Victor o produto e se o número de dedos é ímpar Victor paga a "Cheio de byte" a soma (veja a Figura 4.10).

O discutível inicialmente talvez seja saber se "Cheio de byte" tem dedos e ver se o nome Victor, que em latim quer dizer o vencedor, não predispõe a um joguinho sempre favorável a algum humano!?!.

Aliás em jogos isto sempre acontece!!!

Porém os dedos de "Cheio de byte" serão simulados através da função RND e os de Victor virarão um chute através da instrução INPUT.

Assim fica bem melhor, não é?



Para tornar a situação mais concreta suponhamos que ambos os jogadores começam com a quantia de 100 cruzeiros cada um.

Você caro leitor ou cara leitora pode muito bem perguntar se este jogo é honesto e a resposta é NÃO; ele é levemente a favor de Victor (já viu algum autor não querer tirar de alguma forma a sua vantagem).

Se ambos os jogadores "se aplicarem" da melhor forma possível, pode-se mostrar (e isto será ensinado a alguns nos cursos de Engenharia ou de Administração) que em média Victor ganhará $\frac{11}{23}$ (aproximadamente 48 centavos) por jogo.

Isto significa que "Cheio de byte" pode esperar perder esta quantia por jogo e dessa forma jogando da melhor forma possível, o jogo, Victor tenta maximizar os ganhos, enquanto que "Cheio de byte" tenta minimizar as perdas.

Naturalmente se qualquer um dos oponentes aplicar uma estratégia ou tática mais pobre, as coisas tornar-se-ão diferen-

tes.

A pergunta que já se esperava do aguçado leitor é "Qual é a melhor estratégia para cada jogador"?

A teoria minimax de Von Neumann mostra que se Victor e "Cheio de byte" jogarem muito tempo de cada 23 partidas em 14 devem mostrar dois dedos.

A razão de Victor estar a favor de dois dedos, é que se coloca três dedos com o intuito de ganhar +9 (se Cheio de byte colocar 3) Cheio de byte (que não é "burro") também sabe isto!

Como "Cheio de byte" também é esperto ou seja foi bem ensinado estará a favor de colocar 2 dedos.

Porém Victor espera que ocorra exatamente isto e dessa forma opta mais pelos dois dedos.

Naturalmente os dois oponentes precisam mesclar as suas jogadas, pois de outra forma indicarão antecipadamente o seu plano.

O computador pode não ser "tão esperto" e jogar mais (3) do que (2) dedos.

Isto porque o computador determina a sua jogada escolhendo inicialmente um número aleatório (sem lei de formação) entre 0 e 1 (veja a linha 420).

Se este número aleatório X cai abaixo de um número B ($X < B$), o computador "jogará" dois dedos (veja as linhas 430 e 450).

Porém, bem antes na linha 56 fizemos $B = 9/23$ se o usuário não quer dizer "SIM" sobre a questão acerca da esperteza do computador.

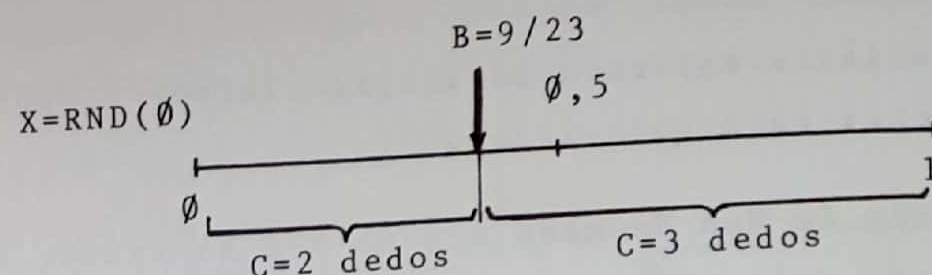


Figura 4.11

Já se for dito que se quer o computador bem "serelepe" dizendo "SIM" para a sua esperteza nas linhas 430 e 450 vemos que ele jogará mais $C=2$ do que $C=3$.

O número de escolhas $C=2$ não será precisamente $\frac{14}{23}$ em um jogo com um número pequeno de lances porém a medida que o número de "partidas" aumentar nos aproximaremos desta razão.

Lembre assim caro leitor que a estratégia ótima de $\frac{14}{23}$ não funciona adequadamente para poucas partidas; tem-se a melhor situação só quando se realizam muitas partidas.

É por esta razão que se fixou uma quantidade necessária que se deve ganhar para ser declarado "vencedor".

Como meta estabeleceu-se que começando com 100 cruzeiros, mais $\frac{11}{23}$ cruzeiros por jogada e mais um termo que "penaliza" alguém que escolheu um pequeno número de partidas (é a parcela $\frac{30}{N}$) para ganhar deve-se obter um valor igual a parte inteira da expressão

$$100 + \frac{11N}{23} + \frac{30}{N}$$

O programa solução é:

```

4 RANDOM
5 LET S1=100 : LET S2=100
10 PRINT "REGRAS: VOCÊ MOSTRA 2 OU 3 DEDOS; A MESMA COISA FAZ"
20 PRINT "O COMPUTADOR, SE O TOTAL É UM NÚMERO PAR O COMPUTADOR"
30 PRINT "PAGA A VOCÊ O PRODUTO DOS NÚMEROS, SE O TOTAL É"
40 PRINT "UM NÚMERO ÍMPAR, VOCÊ PAGARÁ AO COMPUTADOR A SOMA"
50 PRINT "QUANTAS VEZES VOCÊ DESEJA JOGAR"
51 INPUT N
52 LET G=INT (100+N*11/23+30/N)
53 PRINT "VOCÊ COMEÇA COM"; S1; "PONTOS"
54 PRINT "PARA GANHAR VOCÊ PRECISA"; G+1; "PONTOS"
55 PRINT "VOCÊ DESEJA QUE O COMPUTADOR SEJA ESPERTO?"; INPUT A$
56 IF LEFT$(A$,1)="S" THEN LET B=14/23 ELSE LET B=9/23
90 PRINT TAB(25), "VOCÊ"; TAB(30), "MICRO"
100 FOR K=1 TO N
110 PRINT "SUA JOGADA (2 OU 3)"; INPUT V
120 IF INT(V) < > V THEN 110
130 IF ABS(V-2.5) > .6 THEN 110
150 GOTO 420
200 PRINT "O COMPUTADOR TINHA"; C
210 LET S1=S1+P: LET S2=S2-P
220 PRINT "****"; N$; "*****RECEBE="; P; TAB(32); S1; TAB(47); S2
225 IF S1 > G THEN 700
230 NEXT K
235 PRINT "O COMPUTADOR GANHA SEMPRE DOS HUMANOS"
237 PRINT "!# $ % ? * + ! * * + "
240 STOP
410 REM-----ESCOLHA DO COMPUTADOR
420 LET X=RND(0)
430 IF X < B THEN 450
440 LET C=3: GOTO 510
450 LET C=2
500 REM-----CÁLCULO DO GANHO
510 IF (C+V)/2-INT ((C+V)/2) > .1 THEN 540
520 LET P=C*V: LET N$="PAR"
530 GOTO 200
540 LET P=-(C+V): LET N$="ÍMPAR"
550 GOTO 200
700 PRINT "!!!!!!BOA SACHA!!!!!!"
710 PRINT "!!!!!!LEGAL SERIOJA!!!!!!"
720 PRINT "!!!!!!JOIA VOCÊS GANHARAM!!!!!!"
730 PRINT "!!!!!!BOA EU!!!!!!"
800 END
RUN

```

A saída não é dada pois ficou para você brincar não é caro leitor ou leitora?



Figura 4.12 - "O seu micro é maravilhoso.

Nele dá para jogar roleta, "craps", dama, xadrez e uma infinidade de outros jogos que não são matemáticos porém que distraem e acalmam a cada um.

O seu micro seguramente é melhor que o Julio Iglesias que foi definido pelo novelista e crítico espanhol Juan Cueto como a pessoa que projeta uma vulgaridade sentimental esplendida sem muita pretensão cultural com a qual todos nos identificamos por ao menos três minutos ao dia.

Você daqui para frente ficará muito mais do que três minutos por dia com o seu micro e nada o impede de ter também o rádio ou o toca fitas ligado com o som de alguma canção do Julio Iglesias inundando alegremente o ambiente...
Olé, olé, olé

§4.4 - ATÉ QUE ENFIM O MICRO COMO ARTISTA ELE COMO MESTRE DOS GRÁFICOS E DESENHOS

A capacidade de fazer gráficos do micro é frequentemente usada para produzir programas para jogos e quem se habilitar bem neste sentido poderá realmente proporcionar a si mesmo agradáveis horas ou um bom passa tempo.

Assim por exemplo o TRS-80 divide a área gráfica sobre a tela do vídeo em 48 linhas e cada linha é subdividida em 128 áreas retangulares como se fossem pequenos "ladrilhos".

Cada ladrilho vai representar um ponto e pode ser achuriado com preto ou branco.

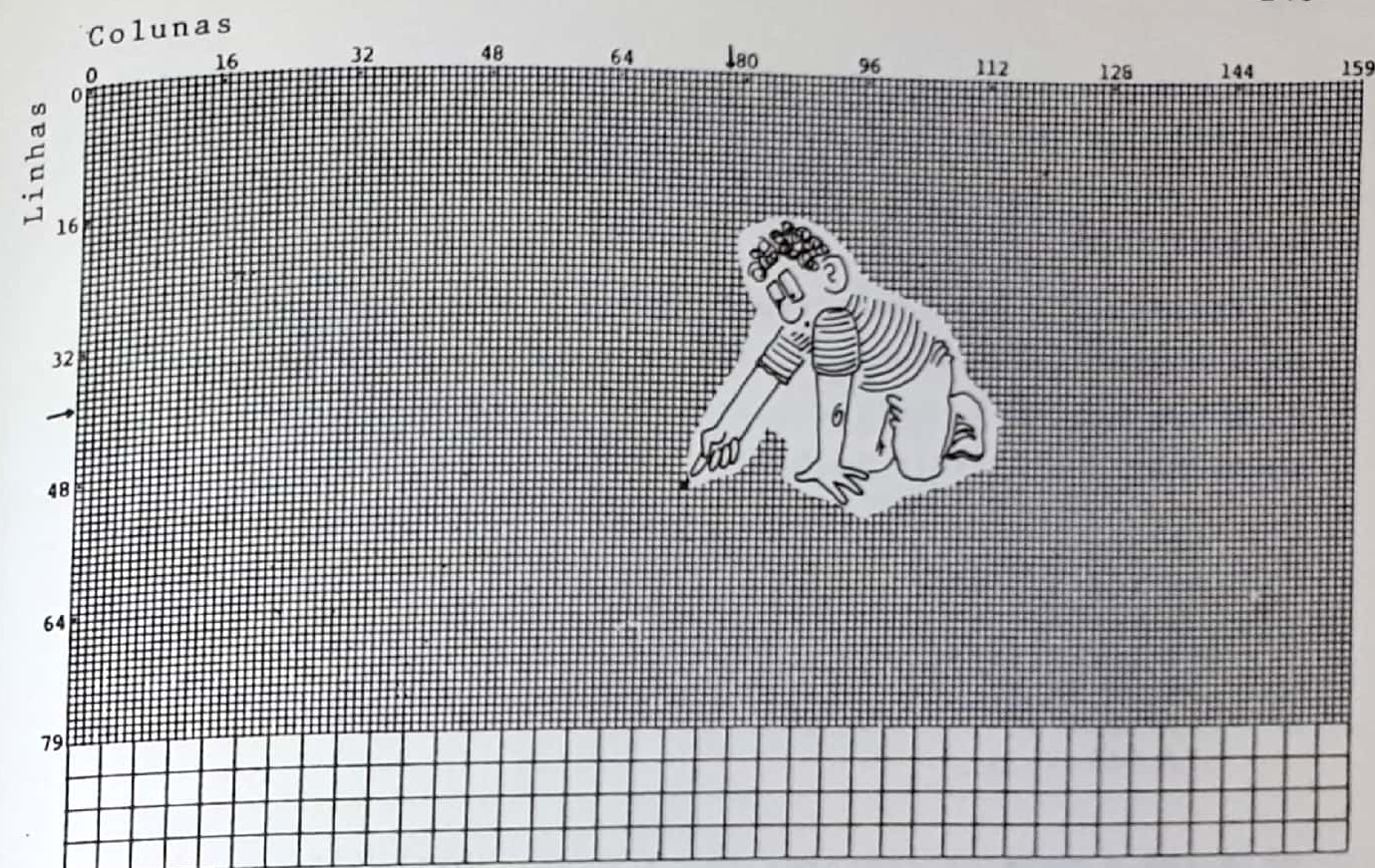


Figura 4.13

De micro para micro este reticulado (Figura 4.13) pode ser outro ou inclusive, através do modo gráfico pode ser até modificado assim como a origem pode não ser no canto alto da esquerda.

Instruções SET-RESET

A instrução SET é utilizada para desenhar um simples retângulo ou ladrilho na tela do vídeo e achuriá-lo com branco (isto é meio estranho, mas na realidade é correto).

A instrução SET(X,Y) colocará um "ladrilho" branco nas coordenadas (X,Y).

O primeiro número X é a abscissa e determina a que distância ele se encontra da coluna mais a esquerda [vamos supor que a nossa origem (0,0) do "grid" de 48 por 128 está localizada no canto baixo à esquerda da tela do vídeo], e o segundo número Y diz o quão elevado ele está em relação a linha mais baixa.

Note bem que a origem que adotamos não confere com o que foi mostrado na Figura 4.13.

Exemplo 4.29

O programa ora analisado representa 13 retângulos na tela do seu vídeo, 12 colocados aleatoriamente e 1 no ponto (7,7) em homenagem ao Sacha que gosta do 7 (será que o número de sorte não é o 13 hein Sacha?).

```
10 CLS
20 FOR N=1 TO 13
30 X=RND(128)-1
40 Y=RND(48)-1
50 SET(X,Y)
60 NEXT N
70 SET(7,7)
```

Dê o RUN e veja se concorda com a nossa explicação.

A instrução RESET funciona da mesma forma que a instrução SET (cuidado que em alguns micros o RESET é para outra coisa) com a diferença que o ladrilho desenhado é achuriado com preto (que é o fundo da tela) no lugar de branco.

Aliás é bom dizer que o fundo da tela pode ser branco ou até colorido com outra cor e nesse caso o ladrilho pode ser preto ou ter até outra cor.

Falaremos também um pouco sobre isso mais adiante...

As instruções SET e RESET serão usadas de forma combinada e o seu uso inteligente nos permitirá ter a idéia que somos um verdadeiro gênio da pintura...

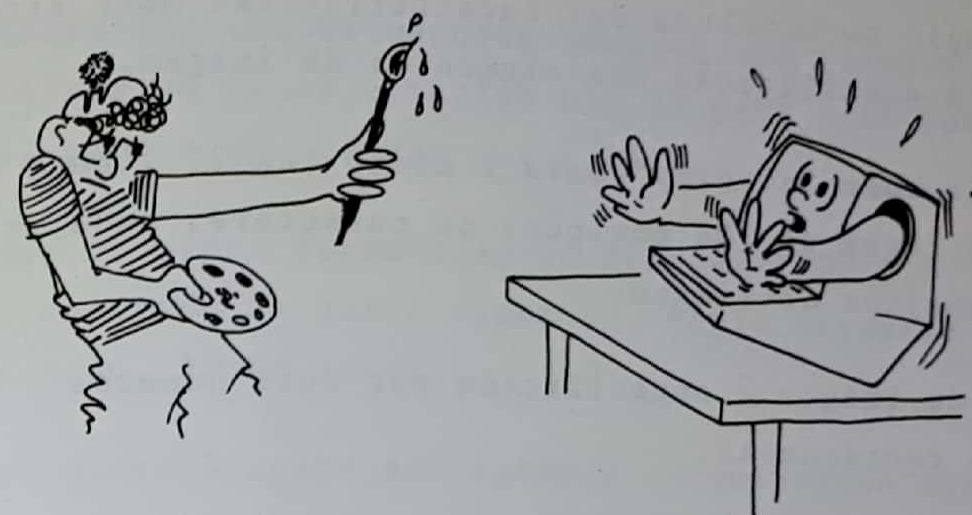


Figura 4.14 - "Pintando o 'sete' com as instruções SET e RESET"

Exemplo 4.30

Brinque um pouco de desenhar com as teclas 5,6,7 e 8 executando o seguinte programa:

```
10000 X=30
20000 Y=20
30000 SET (X,Y)
40000 LET T$=INKEY$
50000 IF T$="5" THEN X=X-1
55000 IF X<0 THEN X=0
60000 IF T$="8" THEN X=X+1
70000 IF X>61 THEN X=61
75000 IF T$="7" THEN Y=Y+1
80000 IF Y>43 THEN Y=43
85000 IF T$="6" THEN Y=Y-1
90000 IF Y<0 THEN Y=0
92000 IF T$="E" THEN CLS
100000 GOTO 30000
```



Este número de linha cabe no seu micro?

Os pequenos micros (aliás todos são pequenos...) não tem o SET e o RESET e como você gostou da brincadeira do Exemplo 4.30 vejamos como pode realizá-la no TK-85 ou no CP-200.

Para isto basta substituir o SET pelo PLOT e o RESET pelo UNPLOT.

Observações importantes:

1) Por exemplo no TK-85 uma das características mais atraentes é também a utilização dos elementos de imagem.

A tela usada como "display" para o micro tem 22 linhas e 32 colunas num total de 704 posições de caractere, cada um contendo 4 elementos de imagem.

O elemento de imagem é especificado por dois números que são as suas coordenadas.

Neste caso teremos os pontos $(0,0)$, $(63,0)$, $(0,43)$ e $(63,43)$ correspondendo, respectivamente, aos cantos, inferior esquerdo, inferior direito, superior esquerdo e superior direito da tela.

A instrução PLOT X,Y faz com que o elemento de imagem, em preto, coincida com este par coordenado.

A instrução UNPLOT X,Y por outro lado apaga o elemento de imagem.



Figura 4.15 - "Seguramente são desenhos porém todos eles parecem com quadros feitos pelo mestre Picasso..."

2) No micro Engenho poderemos colocar cor na tela e podemos escolher uma entre as 16 possíveis [que são numeradas de 0

(preto), 1 (magenta ou vermelho azulado, ...15 (branco)].

Já no micro Engenho os maiores números que podemos utilizar para colocar um ponto colorido são 39 para a 1ª posição (coluna) 47 para a 2ª posição que é linha.

Notar que no micro Engenho ao se digitar PLOT 0,0 coloca-se o ponto colorido na parte superior, canto esquerdo da tela (coluna zero, linha zero).

Assim no micro Engenho por exemplo se quisermos traçar uma linha horizontal de cor azul claro da coluna 7 até a 12 e na linha 14 devemos fazer:

Exemplo 4.31

```

5 COLOR=7
10 PLOT 7,14
20 PLOT 8,14
30 PLOT 9,14
40 PLOT 10,14
50 PLOT 11,14
60 PLOT 12,14
RUN

```

Repetindo, note que o PLOT X,Y do micro Engenho tem para a 1ª posição a coluna e para a 2ª posição a linha.

Em outras palavras a primeira posição faz variar a abscissa e a segunda a ordenada.

A exibição de gráficos na tela muitas vezes requer uma melhor definição da imagem através de um maior detalhamento.

Em vez de usar a divisão da tela em caracteres, é melhor usar uma tela de "pixéis".

Portanto a tela é dividida em píxeis (bem melhor que o nosso "tijolinho" ou "ladrilho") que são $1/4$ de caractere.

Aliás é para isto que se usam as instruções PLOT e UNPLOT.

Como já dissemos a instrução PLOT coloca uma marca (um quadradinho preto) em um pixel, enquanto a instrução UNPLOT retira a marca que tiver sido colocada, isto é, apaga a marca.

Então não esqueça que cada *pixel* ocupa meia ($\frac{1}{2}$) linha e meia ($\frac{1}{2}$) coluna.

Caracteres gráficos

Os caracteres gráficos são usados, como qualquer outro caractere para a formação de cadeias de caracteres, em constantes ou variáveis.

Nos menores micros pessoais (que é o caso do TK-85) o conjunto de caracteres gráficos é constituído por 20 símbolos que estão na Tabela 4.4.

Para obter um caractere gráfico em branco, a tecla SPACE deve ser calcada em *modo normal* de operação do teclado.

Para obter um caractere gráfico totalmente *preto*, deve-se entrar em modo gráfico e então calcar a tecla SPACE, que dará o vídeo inverso de *branco*.

Incluindo-se esses dois símbolos, o conjunto de símbolos gráficos passa a ser de 22 símbolos.

Lembre que para entrar em ou sair do modo gráfico a tecla SHIFT deve ser pressionada simultaneamente com a tecla 9 (GRAPHICS).

Símbolo	Código	Como obter	Símbolo	Código	Como obter
	0	or SPACE		128	SPACE
	1	shift 1		129	shift Q
	2	shift 2		130	shift W
	3	shift 7		131	shift 6
	4	shift 4		132	shift R
	5	shift 5		133	shift 8
	6	shift T		134	shift Y
	7	shift E		135	shift 3
	8	shift A		136	shift H
	9	shift D		137	shift G
	10	shift S		138	shift F

Tabela 4.4 - Símbolos gráficos comuns nos menores micros pessoais como o TK-85



Figura 4.16 - "Não seja egoísta convide e ensine a sua querida mamãe a fazer lindos desenhos no micro".

Agora para conhecer bem o seu micro atuando com a função PLOT você vai passar os seguintes programas.

Exemplo 4.32

```

10 FOR X=2 TO 30 STEP 2
20 FOR Y=2 TO 30 STEP 2
30 SET(X,Y)
40 NEXT Y
50 NEXT X
60 END
RUN

```

O que ocorre?

Acho que você vai encher a tela do vídeo de quadradinhos.

Exemplo 4.33

```

10 FOR X=1 TO 62
20 FOR Y=1 TO 31 STEP 2
30 PLOT X,Y
40 NEXT Y
50 NEXT X
60 FOR X=1 TO 31
70 FOR Y=39 TO 21 STEP -2
80 UNPLOT X,Y
90 NEXT Y
100 NEXT X
110 FOR Y=2 TO 31 STEP 4
120 FOR X=1 TO 31 STEP 3
130 PLOT X,Y
140 NEXT Y
150 NEXT X
160 FOR X=1 TO 31
170 FOR Y=18 TO 2 STEP -2
180 UNPLOT X,Y
190 NEXT Y
200 NEXT X
210 PRINT " MEUS PARABÊNS SERGIO MIRSHAWKA "
RUN

```

Na saída além de ter um desenho parecido com o da bandeira de um país conhecido aparecerá:

MEUS PARABÊNS SERGIO MIRSHAWKA

que foi o autor do programa.

Poderíamos voltar agora ao SET e ao RESET mas é muito importante aprender antes a utilizar bem a instrução PRINT @ ou

em alguns micros a instrução PRINT AT.



Figura 4.17 - "O seu micro ficará mais "forte" e saudável se você usar o PRINT @ ..."

§4.5 - AS INSTRUÇÕES PRINT @ e PRINT AT

Em primeiro lugar a letra A escrita estranhamente como @ se rá lida como a de arroba.

Suponhamos que se tem uma tela de vídeo com 16 linhas de 64 caracteres num total de 1024 caracteres.

A instrução PRINT @ permite começar a saída em qualquer uma dessas 1024 posições de 0 até 1023.

A instrução PRINT@ é muito usada em gráficos.

A forma geral do PRINT @ é:

PRINT @ < posição >, < lista do que deve ser impresso >

onde "posição" refere-se a uma das 1024 numeradas desde 0 até 1023.

Na linha de topo temos as posições de 0 até 63.

A segunda linha vai da posição 64 até a 127 e assim por diante até a última que vai de 960 até 1023.

Por "lista que deve ser impressa" entende-se qualquer expressão válida na instrução PRINT normal.

A instrução 10 PRINT @ 30, "AQUI BREJEIRA!" fará com que a expressão "AQUI BREJEIRA!" saia quase do meio da linha de topo do vídeo enquanto a instrução 10 PRINT @ 409, V\$ imprimirá o conteúdo da variável STRING A\$ no meio da tela.

A "posição" pode ser uma variável.

Por exemplo, se X=863 então a instrução

```
100 PRINT @ X, "PIPOCAS!!!";
```

colocará a palavra "PIPOCAS!!!" a partir da posição 863.

O ponto e vírgula (;) logo após esta expressão faz com que o cursor fique posicionado imediatamente após "PIPOCAS!!!"

Uma instrução PRINT subsequente começará a saída exatamente no ponto onde está o cursor.

Uma instrução como

```
13 PRINT @ 27,"ORA BOLAS"; @441, "ACHO QUE"; @ 860 "ERREI"
```

imprimirá "ORA BOLAS" a partir da posição 27, "ACHO QUE" de 441 em diante e finalmente "ERREI" a partir da posição 860.

Exemplo 4.34

O programa abaixo imprimirá a palavra "SA*C*HA" e "13x7" 14 vezes nas diagonais do vídeo usando a instrução PRINT@ .

```
10 CLS
20 FOR I=0 TO 13
30 N=I*67+1
40 P=(36-3*I)+I*64
50 PRINT @ P," 13*7"; @ N,"SA*C*HA"
60 NEXT I
70 END
RUN
```

Cadê o LET?
Precisa deles?

Isto cabe na sua tela?

Em caso de não caber faça as convenientes modificações.

Exemplo 4.35

Que horas são agora?

Vamos criar um programa para isto usando o PRINT @

```
1000 CLS
2000 PRINT "H=HORAS , M=MINUTO , S=SEGUNDO"
3000 PRINT @ 407 , "H M S"
4000 FOR H=0 TO 23
5000 FOR M=0 TO 59
6000 FOR S=0 TO 59
7000 PRINT @ 470,H;":":M;":":S
8000 FOR N=1 TO 400 : NEXT N
9000 NEXT S
10000 NEXT M
20000 NEXT H
20001 GOTO 10
RUN
```

Execute este lindo programinha no seu super micro.

Sai mesmo o relógio digital?

Em alguns micros usa-se a palavra chave AT na lista de saída de instruções PRINT.

A palavra chave AT, requer dois parâmetros.

O primeiro parâmetro indica o número de linha e o segundo, o número da coluna para onde será movido o cursor, antes da exibição do dado seguinte.

Em uma ou outra implementação, AT tem apenas um parâmetro, isto é, tem a mesma função que TAB, com a diferença apenas de o valor do parâmetro poder ser maior do que o número de caracteres de uma linha.

Neste caso, linhas serão saltadas até que o número de caracteres saltados corresponda a posição definida.

Um exemplo de instrução de saída com posicionamento do cursor é:

Exemplo 4.36

```
10 PRINT X, AT 13,7; Y,Z, AT 5, 18, A; B
```

Explicação:

Com esta instrução dar-se-á a saída ao valor X, numa posição que depende da instrução PRINT anterior.

Depois na coluna 7 e na linha 13, o valor de Y e na mesma linha, no início da próxima zona, o valor de Z.

Essa colocação é provocada pelo AT 13,7, seguido de ; (ponto e vírgula) e pela , (vírgula) entre o Y e o Z.

O próximo posicionamento será na zona que segue imediatamente a coluna 18 porém na linha 5 e portanto haverá um "retrocesso" para a colocação do valor de A, seguido do valor de B, sem intervalo.

Este posicionamento é provocado por AT 5,18 seguido de vírgula (,) e pelo ponto e vírgula (;) entre A e B.

Observação importante:

Entre os micros nacionais destacamos o TK-85 que aceita o PRINT AT.

ESTOU DOMINANDO O BASIC,
MAS UMA INSTRUÇÃO ESTÁ RESISTINDO
AÍNDÁ.
É O TAL DO PRINT "ARROBA"...



Figura 4.18

§4.6 - O INTERESSANTE E DEFINITIVO 2º TEMPO COM A SET E COM A RESET

Exemplo 4.37

Nos podemos desenhar linhas retas (aproximadamente) mudando os "endereço" de X e Y na instrução SET de um laço FOR-NEXT.

Repita este programa para obter uma linha na diagonal.

```
10 INPUT "PONTO INICIAL HORIZONTAL (0 ATÉ 127)";X
20 INPUT "PONTO INICIAL VERTICAL (0 ATÉ 47)";Y
30 INPUT "COMPRIMENTO DA DIAGONAL";K
35 FOR N=1 TO 100:NEXT N
40 CLS
50 FOR L=0 TO K
60 SET(X+L,Y+L)
70 NEXT L
100 GOTO 100
RUN
```

Uma vez que você passou este programa veja se dá para formar um triângulo retângulo fazendo

```
{ 70 SET (X,Y+L)
  80 SET (X+L, Y+K)
  90 NEXT L
```


ou

```

70 SET (X+K, Y+L)
80 SET (X+L, Y)
90 NEXT L

```

Tente ambos os programas - Qual é a diferença entre cada caso?

Resp.- Um é a imagem invertida do outro.

No Exemplo 4.37 poderíamos ter linhas "pontilhadas" ou interrompidas introduzindo um passo (STEP) diferente de "1" nos FOR-NEXT.

Exemplo 4.38

Programa para uma linha horizontal com "vazios"

```

10 INPUT "ENDEREÇO VERTICAL (1 ATÉ 47)";Y
20 INPUT "TAMANHO DO PASSO";H
25 FOR N=1 TO 100: NEXT N
30 CLS
40 FOR X=0 TO 127 STEP H
50 SET(X,Y)
60 NEXT X
70 GOTO 70
RUN

```

Imprima este programa tomando diversos valores de H.

Observe que a medida que você aumenta o H, a linha reta é desenhada bem mais rapidamente (visto que o micro tem menos coisa para fazer).

Realmente com H=10 ou maior não dá quase para perceber como a linha reta está sendo desenhada.

É desta forma que se cria o "quadro" na T.V.

Aliás é ponto por ponto, porém a uma velocidade tão grande que nos não percebemos o "tempo de desenho"!!!!

Mude o programa do Exemplo 4.38 introduzindo as seguintes instruções

```

11 REM** Y PRECISA SER MAIOR QUE 0**
57 RESET (X, Y-1)
70 Y=Y+1
80 IF Y< 48 GOTO 40

```

Neste caso, se H é "pequeno", você poderá ver as linhas serem formadas e "desmanchadas".

Porém se H é relativamente grande (tente 13) a linha vai se movimentar de forma semelhante ao "cinema nos velhos tempos"

Este é a forma com que a *ilusão de movimento* é criada no aparelho de TV e em alguns vídeo jogos bem populares.

Exemplo 4.39

Vamos "pintar" um ponto na tela do vídeo e movê-lo para baixo

```

10 INPUT " PONTO INICIAL HORIZONTAL (0 ATÉ 127)";X
20 INPUT " PONTO INICIAL VERTICAL (1 TO 47)";Y
30 CLS
40 RESET (X, Y-1)
50 SET (X,Y)
60 Y=Y+1
70 IF Y< 48 GOTO 40
100 GOTO 100
RUN

```

Como se vê a função RESET simplesmente apaga o ponto do último SET.

O que ocorreria se você omitisse a instrução com a função RESET? Quando você tentar isto não esqueça de mudar a linha 70 para GOTO 50.

Detalhes, detalhes e momentos especiais...

Um pequeno problema deve ser destacado...

RESET e SET não trabalham com coordenadas negativas.

Perceba assim porque nos começamos com Y de 1 para frente, pois caso contrário corríamos o risco de na linha 40 termos Y=-1.

Exemplo 4.40

Vamos agora mover o ponto para a direita.

```

10 INPUT "PONTO INICIAL HORIZONTAL (1 ATÉ 127)";X
20 INPUT "PONTO INICIAL VERTICAL (1 ATÉ 47)";Y
30 FOR N=1 TO 100: NEXT N
40 CLS
45 RESET (X-1,Y)
50 SET(X,Y)
60 X=X+1
70 IF X<128 GOTO 40
100 GOTO 109
RUN

```

Pequeno exercício para você

Modifique estes dois últimos programinhas (Exemplos 4.39 e 4.40) de tal forma que se mova o ponto para cima e para a esquerda (que tendência hein?!?!?!)

Vamos melhorar as coisas e vejamos como podemos deslocar um ponto até que ele bata em alguma "barreira".

Exemplo 4.41

```

10 INPUT "PONTO INICIAL HORIZONTAL (0 ATÉ 127)";X
20 INPUT "PONTO INICIAL VERTICAL (1 ATÉ 20)";Y
30 INPUT "BARREIRA INFERIOR (30 ATÉ 47)";K
40 CLS
50 FOR N=0 TO 127
60 SET(N,K)
70 NEXT N
80 RESET (X,Y-I)
90 SET(X,Y)
100 Y=Y+1
110 IF Y<K THEN 80
200 GOTO 209
RUN

```

Não parece que o ponto bate na barreira e crava-se na mesma. Agora vamos sofisticar um pouco mais ...

Exemplo 4.42

O ponto vai aparecer no meio e fazer um ricochete na parte de cima e na parte de baixo.

```

10 CLS
20 FOR N=0 TO 127
30 SET(N,0)
40 SET(N,47)
50 NEXT N
60 Y=13
70 H=1
80 RESET(64,Y-H)
90 SET(64,Y)
100 Y=Y+H
110 IF Y=48 THEN 130
120 IF Y<>1 THEN 80
130 Y=Y-2*H
140 H=-H
150 GOTO 90
199 GOTO 199
RUN

```

aí está a formação das "barreiras"



Que beleza, não?



A mudança no sentido de movimento do ponto é pela linha 140 ($H=-H$).

Note que nos precisamos ser bem cuidadosos para não apagar acidentalmente parte da fronteira.

Para efetuar isto não apenas movemos o ponto 2 "passos" para trás com a linha 130 (após termos movido para frente 1 passo na 100) mas também voltamos ao SET na linha 90 antes de voltar ao RESET na linha 80.

Que truque astucioso não é?

Porém você não percebeu que melhorou muito o seu Q.I. depois que começou a mexer com o seu micro e principalmente quando atingiu o atual nível do livro.

Você pode "matar" agora todo o seu dia brincando com esta bola "maluquinha" que bate e volta e que elasticidade e energia tem o seu micro!!!

OBSERVAÇÃO IMPORTANTE:

Não esqueça este programinha, pois vamos usar o mesmo no futuro.

E agora vamos aos quadros ou seja as ilustrações que andam...

Dá para acreditar que você está na iminência de se tornar um artista.

Realmente no micro dá para desenhar a figura que quisermos.

Exemplo 4.43

Vamos começar desenhando um "homem de pau" (quem sabe o próprio Pinocchio estilizado).

Faremos isto em 4 etapas:

1ª PARTE - As suas pernas (as do "homem de pau")

```
10 CLS
20 X=64
30 FOR K=0 TO 7
40 SET (X+K, 40+K)
50 SET (X-K, 40+K)
60 NEXT K
779 GOTO 779
RUN
```

2ª PARTE - O seu corpo e os seus braços

```
70 FOR N=0 TO 5
80 SET (X+N, 34+N)
90 SET (X, 34+N)
100 SET (X-N, 34+N)
110 NEXT N
```

Claro que mais tarde você vai juntar a 1ª com a 2ª parte de forma conveniente e dar o RUN.

3ª PARTE - A cabeça do homem de pau

```
120 SET (X, 32)
130 SET (X+1, 33)
140 SET (X-1, 33)
```

Junte agora a 1ª, 2ª e a 3ª parte e "mande bala". Isto é dê o RUN.

Será que um dia conseguirei fazer desenhos como os que aparecem na Figura 4.19?

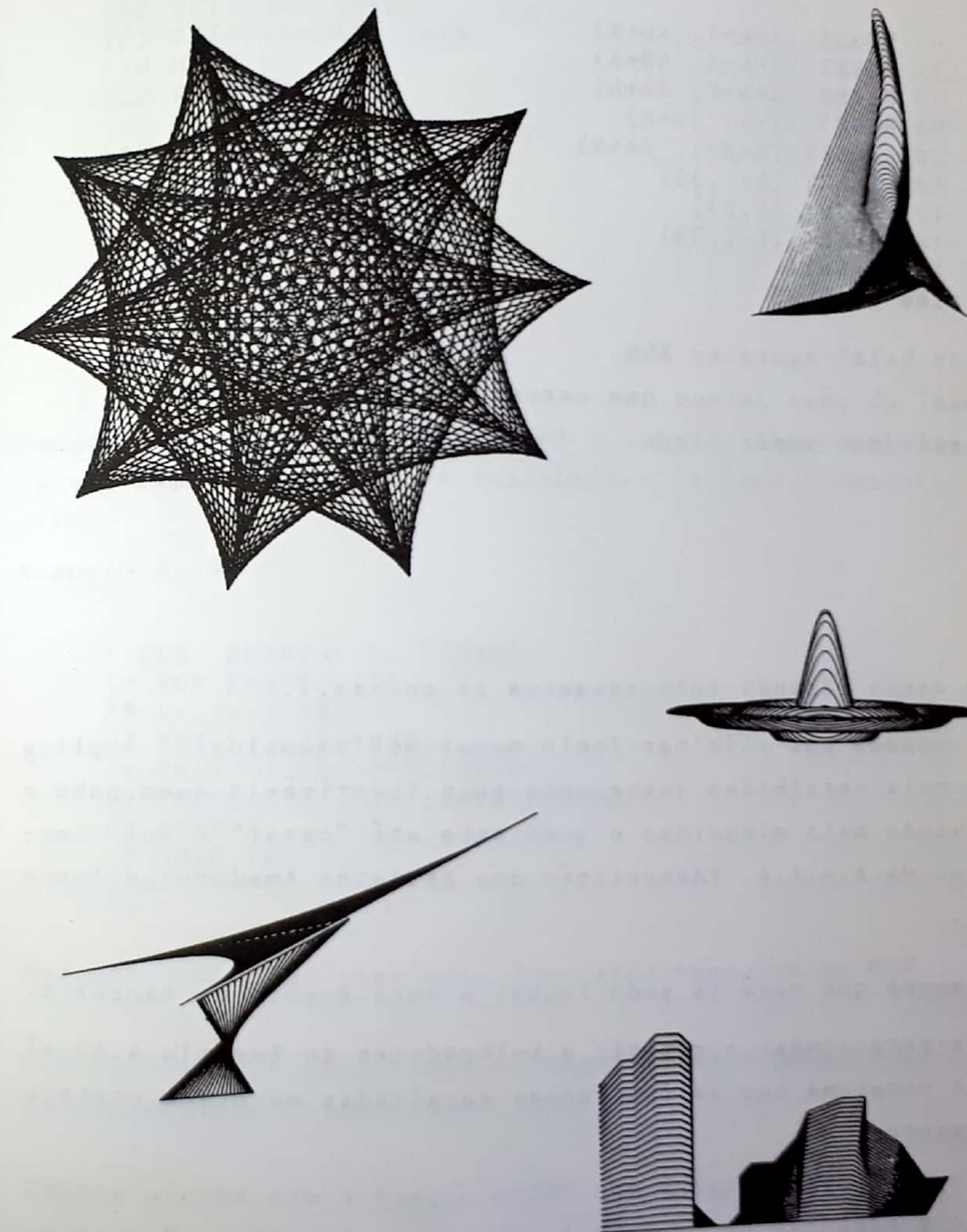


Figura 4.19 - Respondendo a última pergunta. Conseguirás.

4ª PARTE - Finalmente vamos fazer o nosso "boneco" andar para a direita.

Encaixe convenientemente as seguintes instruções

```

45 RESET (X+K-1, 40+K)
55 RESET (X-K-1, 40+K)
85 RESET (X+N-1, 34+N)
95 RESET (X-1, 34+N)
105 RESET (X-N-1, 34+N)
125 RESET (X-1, 32)
135 RESET (X, 33)
145 RESET (X-2, 33)
150 X=X+1
160 GOTO 30

```

"Mande bala" agora no RUN.

Oh mãe! Oh céus parece que estou virando um artista.

Obrigado meu super micro.

Como estão ficando interessantes as coisas...

Nós optamos por eliminar (pelo menos nos exemplos) explicações mais detalhadas justamente para incentivar-lo quem sabe a um estudo mais minucioso e quem sabe até "cavar" a sua inscrição na A.A.A.A. (Associação dos Artistas Amadores e Amorosos).

Bem, agora que você já pode fechar a boca depois de tantos A. Aliás para ajudar a repetir a brincadeira do Exemplo 4.43 aí vai o programa com as instruções encaixadas na ordem certa e crescente.

```

10 CLS
20 X=64
30 FOR K=0 TO 7
40 SET(X+K, 40+K)
50 RESET (X+K-1, 40+K)
55 SET(X-K, 40+K)
57 RESET(X-K-1, 40+K)
60 NEXT K

```

```

70 FOR N=0 TO 5
80 SET(X+N, 34+N)
85 RESET(X+N-1, 34+N)
90 SET(X, 34+N)
95 RESET(X-1, 34+N)
100 SET(X-N, 34+N)
105 RESET(X-N-1, 34+N)
110 NEXT N
120 SET(X, 32)
125 RESET(X-1, 32)
130 SET(X+1, 33)
135 RESET(X, 33)
140 SET(X-1, 33)
145 RESET(X-2, 33)
150 X=X+1
160 GOTO 30
779 GOTO 779
RUN

```

Vamos a mais gráficos, agora com a função trigonométrica seno tão importante para os engenheiros da área industrial e civil.

Exemplo 4.44

```

1 CLS: PRINT @ 1, "SENO"
10 FOR X=0 TO 255
20 Y=SIN(X/40)
30 SET(X/2, 20-Y*20)
40 PRINT @ 50, "X="; INT(X/2)
50 PRINT @ 58, "Y="; INT(20-Y*20)
60 NEXT X
70 GOTO 70
RUN

```

Mas por que é que você está demorando tanto vá ao RUN.

Epa, epa, isto é elogio, não queremos briga e além do mais este programinha é lindo não é?

Certos micros tem a função POINT que é usada para testar se um dado "tijolinho" ou seja um retângulo no vídeo está achuriado de preto ou de branco.

Esta função tem um valor não nulo (geralmente -1) se o "ladrilho" estiver achuriado de branco e em caso contrário vol-



Figura 4.20

ta o zero, significando isto que o ladrilho escolhido está achuriado de preto.

Desta forma a função POINT interroga se uma posição (x,y) está clara (branca) ou escura (preta) e no caso de ser verdade (sim é branca) a resposta é -1 e em caso contrário falsa (está escura) dá o zero.

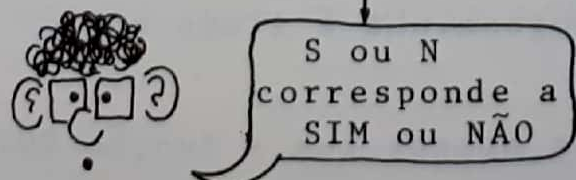
Exemplo 4.45

Veja se com este programa, desde que o seu micro tenha a função POINT isto fica mais claro.

```

10 X=75:Y=20
20 INPUT "VOCÊ DESEJA CLAREAR UM LADRILHO(S/N)";A$
30 FOR F=1 TO 100: NEXT F
40 IF A$="N" GOTO 40
50 SET(X,Y)
60 GOTO 100
80 RESET(X,Y)
100 IF POINT(X,Y)=-1 PRINT@ 200,X;Y,"ESTÁ CLARO"
200 IF POINT(X,Y)=1 PRINT@ 200,X;Y,"ESTÁ ESCURO"
1099 GOTO 1099

```



Exemplo 4.46

Vamos colocar 7 alvos na tela do vídeo em posições aleatórias.

A missão do jogador é destruir todos os sete alvos.

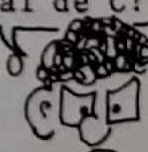
Cada vez que um tiro acerta um alvo o ponto com aquelas coordenadas deve piscar ou cintilar momentaneamente.

```

10 REM A,B=COORDENADAS NAS QUAIS ATIROU O JOGADOR
20 REM C= # DE TIROS CERTEIROS QUE VOCÊ DEU
30 REM CRIAÇÃO DE SETE ALVOS ALEATÓRIOS NA TELA DO VÍDEO
35 FOR F=1 TO 100: NEXT F
40 CLS
50 FOR N=1 TO 7
60 SET(RND(127), RND(41))
70 NEXT N
80 PRINT @ 896,"COORDENADAS DO TIRO (1-127,1-47)";
90 INPUT A,B
100 PRINT @ 896,"
110 IF POINT(A,B)=0 THEN PRINT @ 896,"PERDEU ESSA CHAPA..."
    ELSE PRINT @ 896, "UM ACERTO ,BRAVO!!!":C=C+1
120 REM CRIAÇÃO DO PONTO CINTILANTE
130 FOR M=1 TO 7
140 SET(A,B)
150 FOR A=1 TO 50:NEXT A:REM É ASSIM QUE A GENTE FAZ O
    MICRO SE ATRASAR
160 RESET(A,B)
170 FOR A=1 TO 50: NEXT A: REM VAMOS DAR MAIS UMA ATRASADA
    NO MICRO
180 NEXT M
190 IF C<7 THEN 90
RUN

```

Onde está o valor inicial de C?



Comentários:

Nas linhas de 50 até 70 temos as instruções que permitem colocar os sete alvos na tela do vídeo.

O jogador entra com o seu "chute" na instrução com número de linha 90.

Na linha 110 temos a verificação se o tiro *acerta* ou *não* e informa-se o resultado ao jogador.

Quando a variável C tomar o valor 7 todos os alvos foram destruídos e o jogo terminou.

"HONESTAMENTE 'TIA' ZULMIRA O USO DE PRINT @, NÚMEROS ALEATÓRIOS, SET, RESET ETC. NÃO FAZ COM QUE SE OBTENHA DESENHOS PORNOGRÁFICOS?"



Figura 4.21

UM TRABALHO DURO PORÉM DIVERTIDO E COMPENSADOR DEPOIS DE TERMINADO

- 1) Faça um programa que crie o gráfico de três ondas cossenoidais.
- 2) Faça um programa para obter o gráfico da tangente trigonométrica de um argumento real positivo.
- 3) Usando obrigatoriamente uma instrução SET desenhe uma bola de futebol no meio do vídeo.

Sugestão: Equação de circunferência de círculo que precisará ser usada no programa

$$x = \sqrt{r^2 - y^2}$$

- 4) Faça o seu micro desenhar um retângulo sólido com o vértice superior da esquerda em (13,13) tendo um comprimento de 4 colunas e uma altura de 3 linhas.
- 5) Na tabela abaixo é dada a produção diária de cada empregado do Palácio da Pizza.

É, aqui em São Paulo, metrópole do trabalho e de pouco lazer, a gente só conhece mesmo é Palácio da Pizza.

Elabore um diagrama de hastes ou barras para se ter o gráfico da produção semanal para cada empregado.

Empregado	Produção de pizzas
João Pepipo	28, 22, 17, 14, 26, 30, 24
Antonio Tomate	18, 28, 13, 13, 28, 31, 24
Manuel Batata	27, 25, 14, 13, 29, 32, 24
Carlos Torresmo	31, 26, 13, 13, 25, 30, 24

Tabela 4.5

- 6) Elabore um programa usando a instrução PRINT @ quando for necessário para que apareça a palavra BONG junto ao ponto (sua bola) toda vez que ela fizer uma rebatida com o limite superior e o inferior.

É óbvio que você deve aproveitar o que já está feito no Ex. 4.42 e fazer um programa "joinha" para se distrair.

É claro que você merece um descanso ou seja uma PAUSA agora e nada melhor do que ler a seguinte observação sobre esta instrução.

Observação importante:

No TK-85 por exemplo se você deseja que um programa dure um tempo específico na tela basta aplicar a instrução **PAUSE n**

Essa instrução para durante n quadros da televisão (com 60 quadros por segundo); n pode ser usado até 327 677, o que dá cerca de 9 minutos.

Caso n seja maior, isto significa que deve parar para sempre.

Ao fim de uma pausa, a tela brilhará mais forte.

A pausa pode ser abreviada pelo acionamento de uma tecla, após o início do período de pausa.

A instrução PAUSE deve ser seguida por POKE 16437,255 (verás adiante o que é o POKE e portanto não te amofines).

A instrução PAUSE aparentemente funciona sem isto, mas às vezes poderá abortar o seu programa.

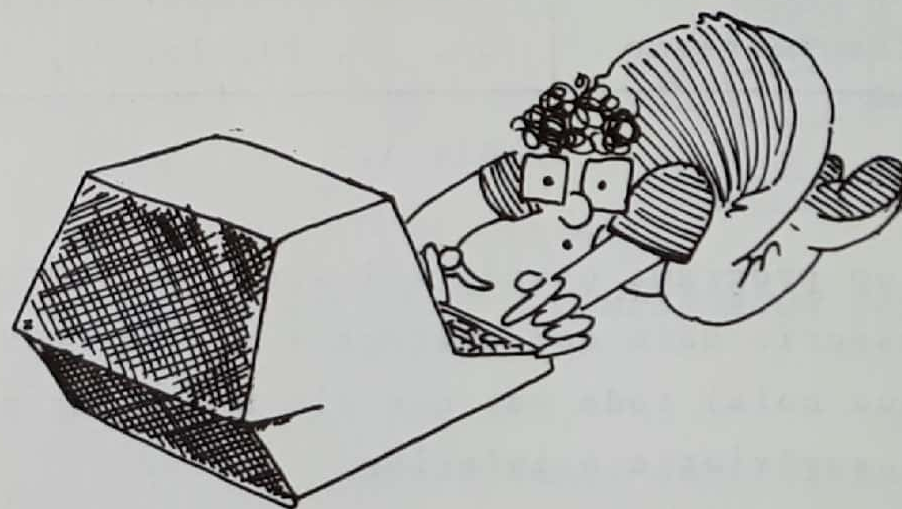


Figura 4.22 - "Caro leitor ou leitora o micro é contagiante mas não se habitue a teclar desta forma pois isto trará consequências maléficas no futuro..."

§4.7 - GRÁFICOS AVANÇADOS

O uso da instrução PRINT @ vai muito além de uma imposição do formato de saída.

Utilizada em conjunção com a função CHR\$, PRINT @ pode aumentar a velocidade de exibição de um gráfico de até seis vezes em algumas aplicações.

É verdade sim, seis vezes mais rápido que o gráfico que se obtém com a instrução SET.

Como pode ser isso?

Bem, precisamos voltar ao nosso conjunto de caracteres do ASCII discutido no §4.2.

Com certeza você se lembra ainda de alguma coisa não é?

Caso a certeza tenha ido para o brejo, volte algumas páginas e refresque ou seja alimente novamente as suas células de memória.

Os códigos de 128 a 191 no ASCII representam caracteres gráficos (isto vale por exemplo para o CP-500).

Mas vamos ver como são estes símbolos e para tanto execute o seguinte programa:

Exemplo 4.47

```
10 CLS
20 FOR I=-128 TO 191
30 IF I=8*INT(I/8)PRINT
40 PRINT I;CHR$(I);" ";
50 NEXT I
60 GOTO 60
RUN
```

Bem, na saída você deve ter os caracteres mostrado na Figura 4.23.

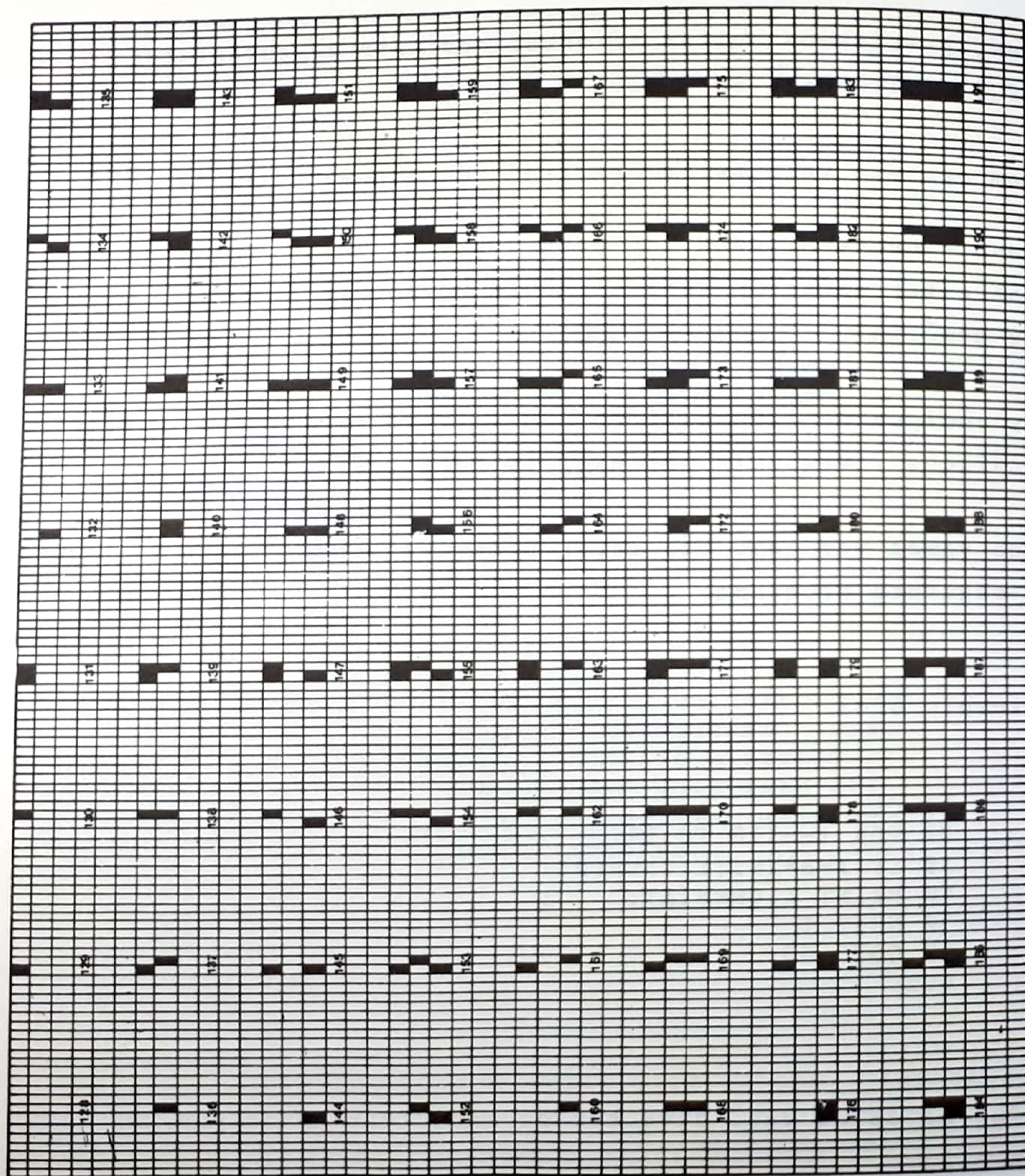


Figura 4.23

Os caracteres são compostos da mesma forma como o foram os retângulos ou os antigos ladrilhos da função SET ou PLOT.

Da Figura 4.23 vê-se que os retângulos são arranjados em grupos de seis: três linhas e duas colunas.

Ao se imprimir os caracteres, pode se perceber claramente que tem-se seis retângulos ou como também diz Ian Sinclair no seu livro sobre o super-popular micro inglês o Dragon-32 obtêm-se seis pixels em uma única instrução.

Comumente isto nos levaria a usar seis instruções SET.

Como se vê o ganho em velocidade é considerável.

Você está cético ainda?

Bem só tem um jeito para sair desta situação: é fazendo um novo programa modelo.

Exemplo 4.48

Faça um programa com o qual saia o desenho da Figura 4.24.

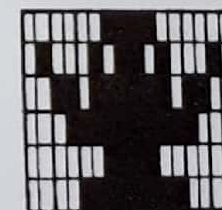


Figura 4.24

Ah, já "afinou"?

Bem vamos lhe ensinar mais alguns truques antes de se meter nessa.

Uma forma para fazer o gráfico da Figura 4.24 é usando o que se indica na Figura 4.23.

Porém procurar por cada caractere leva muito tempo.

Felizmente, existe uma forma simples para calcular o número associado com qualquer caractere gráfico sem usar a Figura 4.23.

Suponhamos que estejamos olhando o número de código do seguinte caractere.



Figura 4.25

Aqui está a forma segunda a qual deveríamos proceder.

Atribui-se os seguintes valores para cada um dos seis retângulos (Figura 4.26).

Memorize este modelo (o da Figura 4.26)!

1	2
4	8
16	32

Figura 4.26

Ele é simples e muito útil.

Para obter o número de código associado com qualquer caractere gráfico deve-se somar os valores de cada pixel em "negro" ao número 128.

Assim no caso da Figura 4.25 nos temos que somar 2 e 4 a 128 e neste caso o número de código ASCII do nosso caractere é

$$2 + 4 + 128 = 134$$

"Treininho"

Qual é o número de código associado ao caractere gráfico mostrado na Figura 4.27?

Bem os "pixels" 16 e 32 estão "negros" logo o número de código é

$$16 + 32 + 128 = 176$$

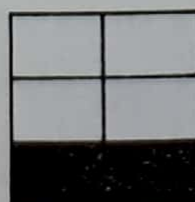


Figura 4.27

Como você percebe não é tão difícil calcular o número de qualquer caractere gráfico.

OBSERVAÇÃO IMPORTANTE:

Como se vê toda a regra do código ASCII está estruturada sobre potências de base 2.

Apesar de não ser tão importante para a linguagem BASIC acreditamos que é oportuno o leitor ou leitora ter boas noções sobre o SISTEMA BINÁRIO.

O sistema numérico com o qual estamos acostumados a trabalhar e pensar é o sistema decimal. Neste sistema os números são formados por algarismos correspondentes às unidades, outros às dezenas, às centenas, etc. É um sistema baseado na analogia com os dedos da mão, ou seja, em potência de 10.

Hoje com o desenvolvimento da ciência da computação trabalhar com o sistema decimal não é conveniente. No sistema binário podemos representar qualquer informação com apenas dois níveis de tensão. Passou-se a utilizar desde então o sistema binário, que é muito simples e utiliza apenas dois símbolos.

A base do sistema binário é o número 2 e os dois símbolos são 0 e 1. Podemos, então, representar qualquer número inteiro (positivo) pela equação:

$$a_n x 2^n + a_{n-1} x 2^{n-1} + \dots + a_2 x 2^2 + a_1 x 2 + a_0$$

onde

$$\begin{cases} a_i = \begin{cases} 0 & \text{se } 0 \leq i < n \\ 1 & \end{cases} \\ a_n = 1 \end{cases}$$

Exemplo 4.49

Qual é o valor de 1001 no sistema decimal?

Resposta:

Por exemplo o número 1001 na base 2 resulta em:

$$\begin{array}{cccc} 1 & 0 & 0 & 1 \\ \downarrow & \downarrow & \downarrow & \downarrow \\ a_3 & a_2 & a_1 & a_0 \end{array} \therefore (1001)_2 = 1 \times 2^3 + 0 \times 2^2 + 0 \times 2^1 + 1 \times 2^0$$

que é igual a 9 no sistema decimal.

Assim, por exemplo $(1011)_{10}$ significa "mil e onze" no sistema decimal, enquanto que $(1011)_2$ significa "onze" no sistema binário.

Ainda não ficou claro porque o sistema binário é conveniente? Vejamos então.

Por exemplo, sabemos que uma chave pode assumir dois estados, fechado ou aberto, ou então, seja uma lâmpada. Esta, ou está acesa ou está apagada, e também apresenta dois estados. Um sinal elétrico pode ser transmitido em forma de pulsos, ou seja instantes com tensão e instantes sem tensão, portanto dois estados.

Daí resulta a grande vantagem do sistema binário, passa-se a atribuir aos estados um dígito que passa a representar a condição elétrica.

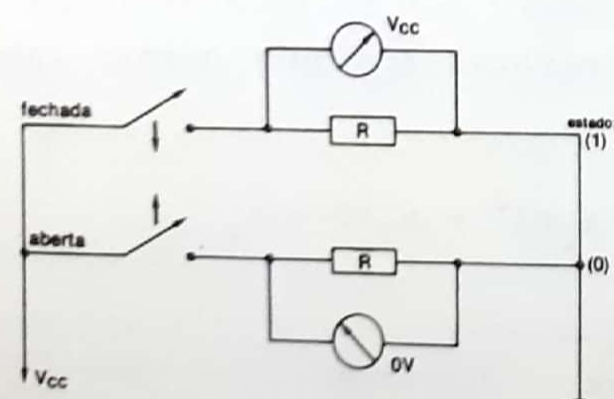


Figura 4.28 Chave

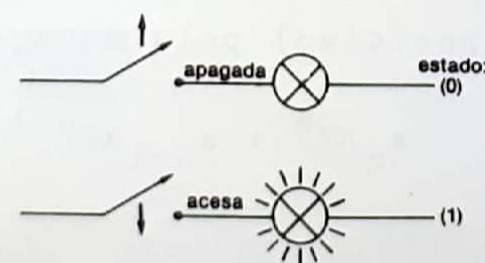


Figura 4.29 Lâmpada

Exemplo 4.50

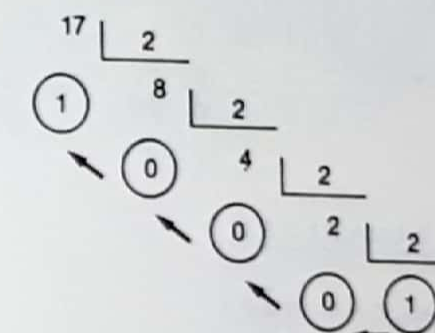
Quanto é $(1011101)_2$ no sistema decimal?

Resposta:

$$(1011101)_2 = 1 \times 2^6 + 0 \times 2^5 + 1 \times 2^4 + 1 \times 2^3 + 1 \times 2^2 + 0 \times 2^1 + 1 \times 2^0 = (93)_{10}$$

Exemplo 4.51

Vamos inverter agora o processo. A partir do número decimal obter o número binário. Seja 17 o número. O processo basicamente é dividir o número por 2 e o resto por 2 sucessivamente até obter resto 1. Os "restos" na sequência inversa formarão o número binário.

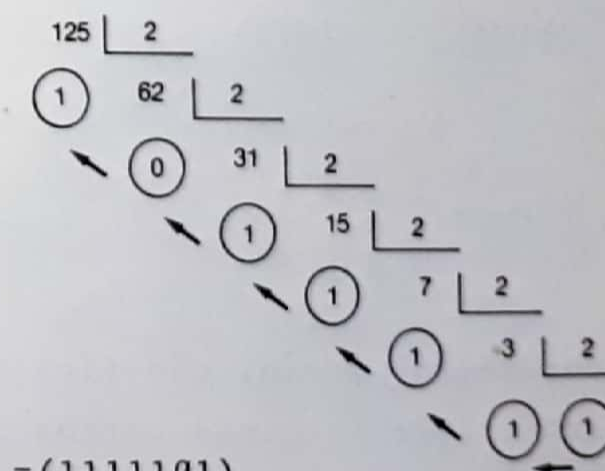


$$\text{Portanto: } (17)_{10} = (10001)_2$$

Exemplo 4.52

Quanto é $(125)_{10}$ no sistema binário?

Resposta:



$$\text{Portanto: } (125)_{10} = (1111101)_2$$

Vejamos agora as operações no sistema binário.

Adição de números binários:

Para a adição de números binários basta seguir as seguintes regras:

$$\begin{cases} 0 + 0 = 0 \\ 0 + 1 = 1 \\ 1 + 1 = 0 \end{cases} \text{ e carrega-se 1 para a pr\u00f3}$$

xima coluna mais significativa (vai um).

Exemplo 4.53

Faça a adi\u00e7\u00e3o $(101011)_2 + (1101)_2$

$$\begin{array}{r} 1111 \\ 11111 \\ + 1111 \\ \hline 11100 \end{array} \quad \begin{array}{l} \text{vai um} \\ \text{vai um} \end{array}$$

Subtra\u00e7\u00e3o de n\u00fameros bin\u00e1rios:

Para a subtra\u00e7\u00e3o as regras s\u00e3o:

$$\begin{cases} 0 - 0 = 0 \\ 1 - 0 = 1 \\ 1 - 1 = 0 \\ 0 - 1 = 1 \end{cases} \text{ e empresta 1 da pr\u00f3xima co}$$

luna mais significativa (vem um).

Exemplo 4.54

Faça a opera\u00e7\u00e3o $(1101)_2 - (1011)_2$

$$\begin{array}{r} 111 \\ - 1111 \\ \hline 0010 \end{array} \quad \begin{array}{l} \text{vem um} \\ \text{vem um} \end{array}$$

O universo da computa\u00e7\u00e3o, por\u00e9m, n\u00e3o fica restrito ao sistema bin\u00e1rio. A partir dele formamos outros sistemas num\u00e9ricos (octal, hexadecimal, etc.) muito importantes e aos quais dedica-se muita aten\u00e7\u00e3o.

Por exemplo a grande sensa\u00e7\u00e3o de 1984 deve ser o micro de 16 bits.

TRADUZINDO DA MELHOR
FORMA POSS\u00cdVEL
QUE EU POSSO, TEM-SE:
(10011100011001100
110011001100111)



Figura 4.30

Realmente no caso de termos a situa\u00e7\u00e3o indicada na Figura 4.30 podemos cair no colapso do poder bin\u00e1rio indicado na Figura 4.31.

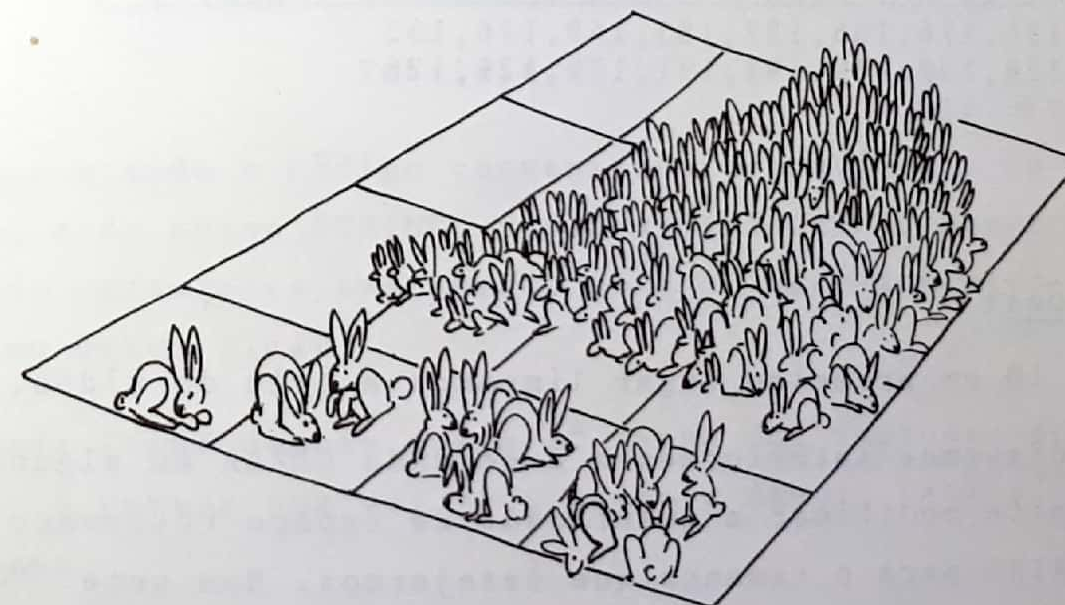


Figura 4.31

Bem voltemos ao Exemplo 4.48 praticando a regra baseada no sistema bin\u00e1rio para os n\u00fameros de c\u00f3digo dos caracteres gr\u00e1ficos.

Depois de "quebrar" um pouco o "c\u00f3co" voc\u00ea deve obter duas linhas com os seguintes n\u00fameros de c\u00f3digo.

136	176	168	187	183	148	176	132
128	130	179	191	191	170	129	128

Vamos agora construir a variável alfanumérica ou STRING.

Ao primeiro ímpeto, poderíamos querer usar o método da força bruta ou seja em alguns casos também chamado de "método lusitano" como por exemplo mandar uma instrução do tipo

$X\$ = \text{CHR}\$(136) + \text{CHR}\$(176) + \text{CHR}\$(168) + \dots$

Porém podemos fazer uma coisa um pouco mais inventiva e colocar estes números na instrução DATA e ler (instrução READ) os mesmos dentro do programa.

Desta forma o programa completo para o Exemplo 4.48 é:

```
10 CLS : CLEAR 3000 : DEFSTR X
30 FOR I=1 TO 8 : READ N : X=X + CHR$(N) : NEXT I
40 X=X + CHR$(26) + STRING$(8,24)
50 FOR J=1 TO 8 : READ M : X=X + CHR$(M) : NEXT J
70 DATA 136,176,168,187,183,148,176,132
80 DATA 128,130,179,191,191,170,129,128
100 PRINT@ 410,X
110 END
RUN
```

Explicações:

Na linha 10 em primeiro lugar limpamos a tela do vídeo.

Como já dissemos anteriormente o comando CLEAR em alguns micros permite modificar a quantidade de espaço reservado para um STRING para o tamanho que desejarmos. Bem este "desejarmos" tem sempre um limite.

Por outro lado podemos eliminar todo o espaço reservado, deixando toda a memória para o uso de variáveis que não sejam alfanuméricas.

Definição: DEFSTR

A instrução DEFSTR permite-nos a priori definir quais são as variáveis que devem ser STRING e com isto não é neces-

sário mais usar o cifrão (\$).

Na linha 30 cada número de código é lido como N e convertido em caracter gráfico via função CHR\$ e acrescido ao fim da variável STRING X.

Lembre que X já foi definido como STRING no fim da linha 10. A linha 40 é necessária para que a posição da 2ª linha esteja diretamente abaixo da primeira.

Devemos agora voltar a tabela ASCII e prestar uma particular atenção aos códigos de 24 a 27.

Decimal	Função
24	Retrocesso sem limpeza
25	Avança o cursor
26	Desce o cursor
27	Sobe o cursor

Tabela 4.6

Acrescentando o código conveniente de movimento do cursor no meio do nosso STRING, podemos deslocar ou mover a 2ª linha de caracteres exatamente para o lugar que eles pertencem ou devem ficar.

E dessa forma depois que a 1ª linha foi impressa precisamos mover o cursor uma vez para baixo e depois para a esquerda 8 vezes.

Acrescentamos para isto

$40 X=X + \text{CHR}\$(26) + \text{STRING}\$(8,24)$

É este o truque!!!

Aliás expliquemos também o STRING\$(P,A) e não confunda com STR\$.

Nos já vimos o STR\$, o qual converte uma variável numérica para uma variável STRING.

Já o STRING\$(P,A) é um modificador "especializado" do PRINT que nos permite imprimir um caractere simples ASCII representado por A, num total de P vezes. Aliás você já recebeu esta informação anteriormente. Veja como a figura sai rapidamente uma vez que o STRING estiver montado na memória.

Veja ainda se você consegue fazer o mesmo programa usando instruções SET.

É dose para mamute, não é?

Não há comparação.

Vimos até agora um pequeno segredo dos gráficos.



Figura 4.32

TRABALHO PARA VOCÊ

Faça os programas para se obter os seguintes desenhos

1) -

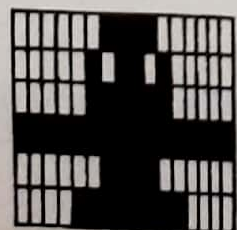


Figura 4.33

2)

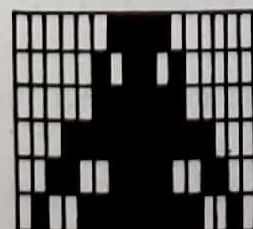


Figura 4.34

3)

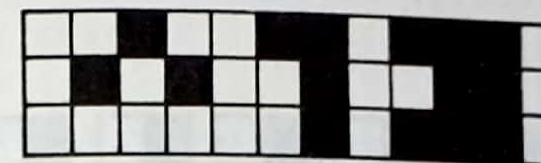


Figura 4.35

Porém nem todos tem ainda micros muito sofisticados e precisamos também saber usar os gráficos de baixa resolução ou seja com poucos detalhes.

Quem sabe depois do exemplo que vamos resolver você consegue fazer no micro a Figura 4.36.

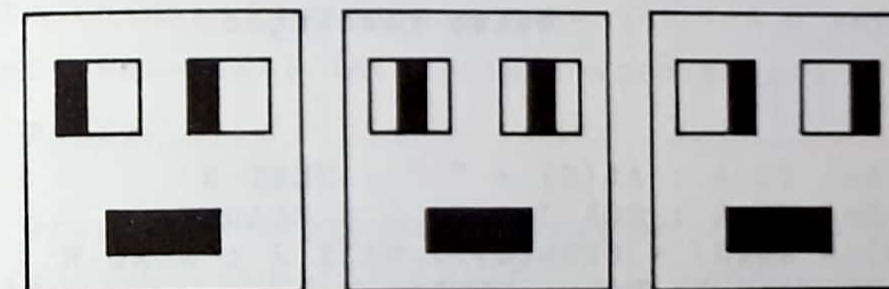
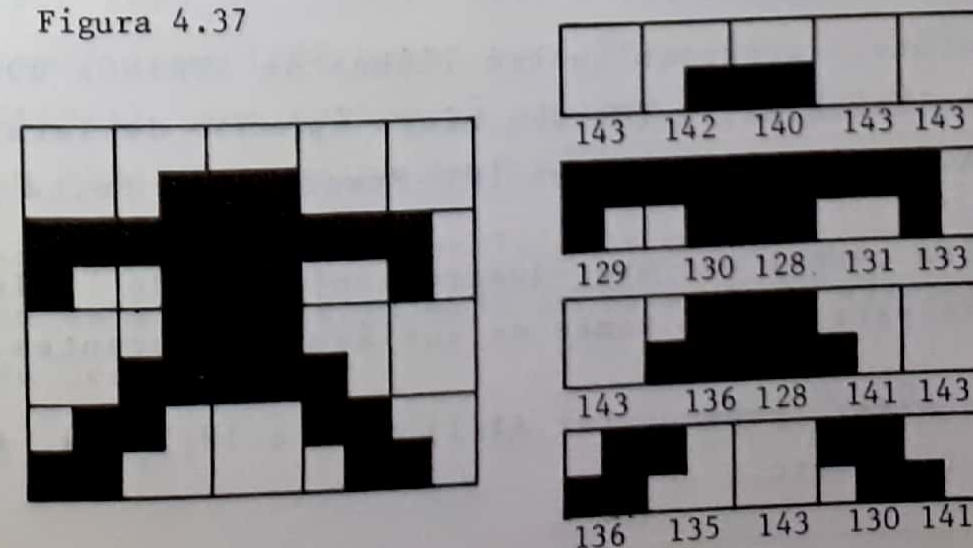


Figura 4.36

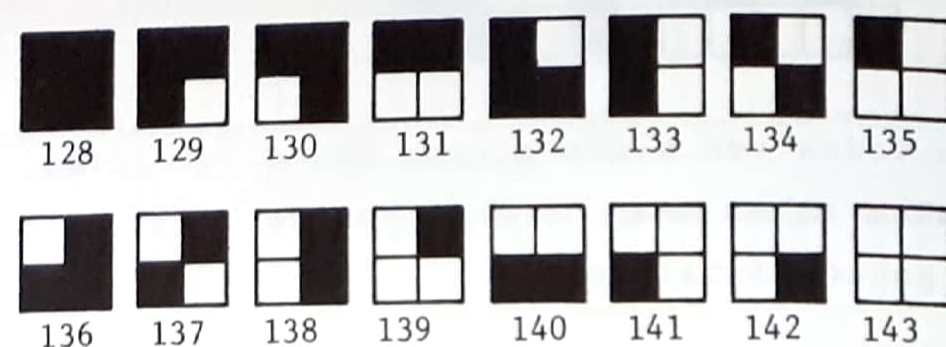
Exemplo 4.55

Ian Sinclair no seu livro "The Dragon-32 e and how to make the most of it" elabora o programa para se obter o seguinte desenho (Figura 4.37).

Figura 4.37



Para o Dragon-32, que é um micro muito comum na Inglaterra o conjunto de caracteres em código decimal é:



Chave da cor: ☐ Verde ☒ Preto

Figura 4.38 - Utilização dos gráficos de baixa resolução.

```

10 CLS
20 FOR N=1 TO 4 : A$(N) = " " : NEXT N
30 FOR N=1 TO 4 : FOR J=1 TO 5 : READ C
40 A$(N) = A$(N) + CHR$(C) : NEXT J : NEXT N
50 REM CRIAÇÃO DA FORMA PEDIDA
60 FOR N=1 TO 4
70 PRINT@ 106 + 32*N, A$(N)
75 NEXT N
80 PRINT
85 PRINT TAB(9) "SER EXTRA TERRESTRE"
90 END
95 DATA 143, 142, 140, 143, 143
96 DATA 129, 130, 128, 131, 133
97 DATA 143, 136, 128, 141, 143
98 DATA 136, 135, 143, 130, 141
RUN

```

Algumas explicações

Criou-se neste programa quatro linhas de STRINGS usando a variável coletiva A\$() (No seu micro é preciso declarar a dimensão de A\$ se ela é menor que 10? Precisa ser feita alguma modificação?

Isto evita termos que usar quatro conjuntos de instruções READ-DATA para quatro nomes de variáveis diferentes.

Usando o ARRAY, podemos ter A\$(1) para a 1ª linha, A\$(2) para a 2ª linha etc.

O desenho que queremos, necessita de cinco posições por linha de forma que para cada variável alfanumérica subscrita nos precisamos colocar cinco números CHR\$.

É por isto que se coloca o laço FOR J=1 TO 5.

É a linha 40 que cria realmente o desenho do E.T. (extra-terrestre) depois de termos lido os números de código dos caracteres.

Note que na linha 70 temos a impressão que começa por A\$(1) na posição $106 + 32 \times 1 = 138$ e que equivale a TAB(10), 5ª linha para o Dragon-32 e que pode não servir para o seu micro.

Fizemos este exemplo justamente para que com o pequeno uso do seu grande potencial intelectual você adapte o programa para o seu micro.

No caso do Dragon-32 a próxima impressão ou seja de A\$(2) vai para $106 + 32 \times 2 = 170$ o que equivale a TAB(10) na 6ª linha.

Isto continua até que se imprimam todas as quatro linhas.

TRABALHO PARA VOCÊ

- 1) Faça o Exemplo 4.55 de tal forma que se possa armazenar os caracteres em uma única variável STRING e faça o E.T. se mover.

Para isto será necessário usar o CLEAR 1000 (o qual como já sabemos toma 1000 bytes para armazenar STRINGS) e do STRING\$(P,A) (que também já sabemos é uma função que fornece um STRING composto de P símbolos ou caracteres).

Não esqueça para este exercício os códigos dos caracteres que estão na Figura 4.38 ou então veja aqueles do manual do seu micro.

Mãos a obra!!!

- 2) Para satisfação pessoal de que você já é um grande teclador repita este programa:

```

10 REM "ESC"
20 LET A=16
30 LET B=5
40 LET P=240
50 LET N=3
60 FOR I=A/A TO 12
70 FOR J=A/A TO N
80 PRINT AT A,B;"█"
90 PRINT TAB B;"█";TAB B-1;"█";TAB B;"█";TAB B;"█";
  TAB B;"█"
100 NEXT J
110 CLS
120 GOTO P
130 FOR K=1 TO N
140 PRINT AT A,B;"█"
150 PRINT TAB B;"█";TAB B;"█";TAB B;"█";TAB B;"█";
  TAB B;" "
160 NEXT K
170 CLS
180 NEXT I
190 LET P=270
200 LET N=1
210 PRINT AT 0,25;"█";TAB 25;"█"
220 IF A=16 THEN STOP
230 GOTO 60
240 LET A=A-1
250 LET B=B+1
260 GOTO 130
270 LET A=A+1
280 LET B=B-1
290 GOTO 130
RUN

```

Observação: Veja se no seu micro não precisa colocar o que vem depois do TAB entre parênteses.

- 3) Faça um programa que permita a saída dos desenhos da Figura 4.39.



Figura 4.39

Dá para fazer o programa que faça o "hominho" da Figura 4.39 fazer ginástica?



Figura 4.40 - "Você já do mina um pouco os gráficos mas vai ficar melhor ainda!!!"

§4.8 - PEEK E POKE

PEEK (espiar) e POKE (meter ou colocar) são palavras da linguagem BASIC que permitem fazer coisas que não são do BASIC.

É possível, se bem que não seja simples, executar programas em linguagem de máquina, isto é, linguagem de microprocessador.

É necessário conhecer a linguagem de máquina ou seja a linguagem assembler.

Não é nosso intuito tornar o leitor "expert" em programação em linguagem de máquina ou mesmo explicar como o seu micro funciona eletronicamente, mas sim aprender a utilizá-lo.

Entretanto com um programa escrito em assembler, é possível transcrevê-lo para um código numérico que será armazenado internamente antes da execução.

Esse armazenamento é possível através da instrução POKE dentro de um programa escrito em linguagem BASIC.

Pode-se, dentro de um programa BASIC, ler o conteúdo de posições de memória através de uma referência à função PEEK.

A função PEEK aparece como `PEEK (endereço)` e a função POKE como `POKE < endereço, valor >`

Nos iremos introduzir e abordar estas instruções de uma forma cuidadosa para não cair no abismo computacional...

Exemplo 4.56

Como um primeiro treino veja o que ocorre no seu micro (se é que ocorre alguma coisa) quando você entrar com o seguinte programa:

```
20 K=0
30 PRINT K, PEEK(K), CHR$(PEEK (K))
40 K=K+1
50 GOTO 30
```

Façamos uma análise do programa antes de executá-lo.

Na linha 20 nos começamos com o endereço zero onde aliás desejamos iniciar as nossas "espiadelas".

A linha 30 faz com que se imprima três coisas:

- a) O endereço - isto é, o número do byte cujo conteúdo nos estamos querendo ver.
- b) O conteúdo deste byte, expresso como um número decimal entre 0 e 255.
- c) O conteúdo daquele endereço convertido no seu caractere ASCII.

Deve-se destacar que muitos caracteres ASCII não podem ser impressos.

OK, agora "estraçalhe" com todo carinho o comando RUN e pare o micro quando ocorrer algo interessante através da tecla SHIFT@ ou pare o mesmo com a tecla BREAK ou então recomence sem precisar passar por K desde 0 até o ponto no qual você parou com a tecla CONT.

Você verá coisas interessantíssimas neste programa, acredite e para tanto execute-o.

Com a função POKE devemos tomar muito cuidado para não estragar tudo que já existe na memória.

Se utilizarmos esta função fora da faixa poderemos armazenar bytes em posições críticas, como, por exemplo, dentro de nosso programa, ou mesmo em lugares piores.

Desde que POKE possa ser usado para armazenar informações, em qualquer lugar da memória, é importante elaborar os gráficos dentro da faixa de posicionamento da tela.

Exemplo 4.57

Programa para sentir o efeito do POKE

```
10 REM *PROGRAMA PARA "SENTIR" O POKE*
20 N=20000
30 READ D
40 POKE N,D
50 N=N+1
60 IF N=20011 THEN END
70 GOTO 30
100 DATA 80,69,69,75,45,65,45,66,79,79,33
```

Antes da execução analisemos um pouco este programa.

Na linha 20 inicializa-se o endereço em 20000.

Na linha 30 lê-se um número da linha DATA.

Na linha 40 coloca-se (POKE) o dado D no endereço "N".

Na linha 50 incrementa-se o número do endereço de uma unidade.

Na linha 60 chega-se ao fim da execução quando colocarmos ou seja introduzirmos todos os valores nas específicas posições de memória dadas pelos endereços "N".

Com a linha 70 volta-se para ler mais dados.

Na linha 100, temos os dados que iremos introduzir (POKE) na memória.

Bem agora que está tudo explicadinho é a hora do RUN.

Bem isto foi muito rápido.

Você deve estar pensando, o que aconteceu?

Como poderemos descobrir?

Deveríamos dar uma espiadela (PEEK) nele?

Sim, mas vamos deixar o velho programa (Exemplo 4.57) no micro e começar um novo na linha 200.

Exemplo 4.58

Este resultado está intrinsecamente ligado ao do Exemplo 4.57

```
200 REM * PEEK PROGRAMA
210 FOR N=20000 TO 20010
220 LPRINT N, PEEK(N)
230 NEXT N
RUN
```

Teremos na saída:

20000	80
20001	69
20002	69
20003	75
20004	45
20005	65
20006	45
20007	66
20008	79
20009	79
20010	33

E dava para ter outro tipo de saída?

Claro que não, pois com o POKE nos colocamos (Exemplo 4.57) e com o PEEK nos espiamos (Exemplo 4.58).



Figura 4.41

§4.9 - OPERADORES LÓGICOS - AND, OR E NOT

Na Matemática Clássica (palavras caprichosas para idéias simples...) existe o que se chama de AND (e) lógico, OR (ou) lógico e o NOT (não) lógico.

Onde já se viu um *não lógico* para a maior parte das situações.

E dessa forma vejamos o que uma "vaquinha" disse para outra...

Na Figura 4.42 se a "porteira" A AND (e) a porteira B AND (e) a porteira C estiverem abertas, a vaca pode-se deslocar do pasto # 1 para o pasto # 2.

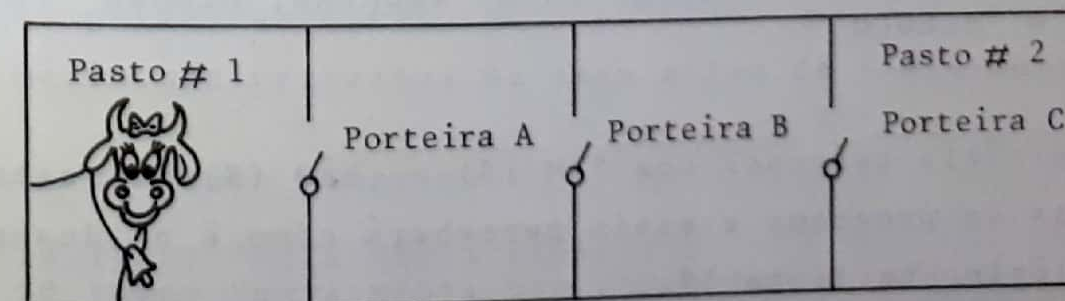


Figura 4.42

A propósito o nome desta "vaquinha" é Mimosa.

Se qualquer uma das portas estiver fechada o trajeto da vaca está bloqueado.

A este princípio chama-se de AND(e) lógico.

Já na Figura 4.43 se a porta X ou (OR) a porta Y ou (OR) a porta Z estiverem abertas então a Mimosa pode-se deslocar do pasto # 3 para o pasto # 4 e lá quem sabe poderá encontrar o touro Franklin.

Estamos agora aplicando o princípio do OR (ou) lógico.

Aliás as idéias expostas até agora parecem ser bem lógicas.



Figura 4.43

Claro que se a vaca pode compreender isto, você também pode não é! Você é o "homo sapiens" já esqueceu disso?

Veja agora como ilustração do AND um programa

Exemplo 4.59

```
10 INPUT "ESTÁ A PORTEIRA 'A' ABERTA";A$
20 INPUT "ESTÁ A PORTEIRA 'B' ABERTA";B$
30 INPUT "ESTÁ A PORTEIRA 'C' ABERTA";C$
40 LPRINT
50 IF A$="S" AND B$="S" AND C$="S" THEN GOTO 80
60 LPRINT "A VELHA MIMOSA NÃO PODE IR AO PASTO # 2."
70 END
80 PRINT "TODAS AS PORTEIRAS ESTÃO ABERTAS, MIMOSA 'PARTE'
   PARA O 'ALMOÇO'"
RUN
```

Como exercício responda com SIM (S) ou NÃO (N) em execuções distintas do programa e assim perceberá como é o desempenho do AND lógico na linha 50.

Mas onde está a LÓGICA em tudo isto?

Você não deve se inquietar, pois provavelmente entendeu todas as linhas do último programa menos (talvez) a linha 50.

Nas linhas 10, 20 e 30 entra-se com as posições das portas como abertas (que definimos como sendo igual a "S") ou fechadas (definidas com "N").

A linha 50 é a chave de tudo!!!

A explicação literal é "Se a porta A está aberta e (AND) a porta B está aberta e (AND) a porta C está aberta, então vá para a linha 80. Se qualquer uma dessas portas estiver fechada, indique este fato indo a linha 60".

Imagine como esta simples lógica, poderia ser usada para criar um "computador" super simples consistindo somente de uma chave elétrica (relê) em cada porta em cujo circuito estivesse uma bateria e uma lâmpada instalada na casa do fazendeiro.

A lâmpada indicaria se as portas estão todas abertas.

Um tal computador de verificação de portas teria somente três células de memória - os relês.

Exemplo 4.60

Uma certa escola utiliza para os seus alunos o seguinte critério para aprovação.

Se o aluno obtiver nota média final acima de 5 ou (OR) média das primeiras notas igual ou maior que 7 e (AND) uma nota média nos trabalhos de casa acima de 8 ele passará.

O correspondente programa é:

```
10 INPUT "NOTA MÉDIA FINAL"; F
20 INPUT "NOTA MÉDIA DAS NOTAS INICIAIS"; M
30 INPUT "NOTA MÉDIA DOS TRABALHOS DE CASA"; C
40 IF F>5 OR M >= 7 AND C>8 THEN 70
50 PRINT "NÃO PASSOU"
```



```

60 END
70 PRINT "PASSOU"
80 RUN

```

Entre com o programa no seu micro e execute (RUN) algumas vezes e veja a forma eficiente como os testes OR lógico e AND lógico "trabalham" na mesma linha do programa (40).

Isto já lhe deu uma idéia melhor do poder e da conveniência da lógica matemática?

Vejamos outros exemplos:

Exemplo 4.61

Vamos entrar com dois números e verificar se eles são ambos positivos, ambos negativos ou tem sinais diferentes.

```

10 INPUT "O PRIMEIRO NÚMERO É"; X
20 INPUT "O SEGUNDO NÚMERO É"; Y
30 IF (X>=0) AND (Y>=0) THEN 70
40 IF (X<0) AND (Y<0) THEN 90
50 LPRINT "SINAIS CONTRÁRIOS"
60 END
70 LPRINT "AMBOS POSITIVOS OU NULOS"
80 END
90 LPRINT "AMBOS NEGATIVOS"
RUN

```

Entre como verificação do programa com vários valores de X e Y e veja o que ocorre.

Na parte gráfica são também muito úteis os operadores lógicos.

Exemplo 4.62

```

10 CLS
20 FOR X=0 TO 127
30 FOR Y=0 TO 47
40 IF (X>=64) AND (Y>=24) THEN 60
50 SET (X,Y)
60 NEXT Y
70 NEXT X
99 GOTO 99

```

Antes de "dar" o RUN você consegue perceber a pequena surpresa que será causada pelo AND lógico na linha 40.

Será que você adivinhou corretamente???

O que irá ocorrer se você substituir o AND na linha 40 por um OR?

Depois de pensar um pouco, faça a mudança proposta e veja o resultado executando novamente o programa.

Será que mais uma vez você "chutou" certo???

Oh, sim o único limite é a sua imaginação.

Veja como a notação lógica torna simples o traçado ou seja o desenho de linhas.

Exemplo 4.63

```

10 CLS
20 FOR X=0 TO 127
30 FOR Y=0 TO 47
40 IF (X>=64) OR (Y>=24) THEN 60
50 SET (X,Y)
60 NEXT Y
70 NEXT X
99 GOTO 99
RUN

```

O que acontece com o programa do Exemplo 4.63 se você substituir o OR pelo AND?

Esboce o seu resultado estimado e então mude a linha 40 e experimente o novo programa.

Espero que mais uma vez você tenha acertado.

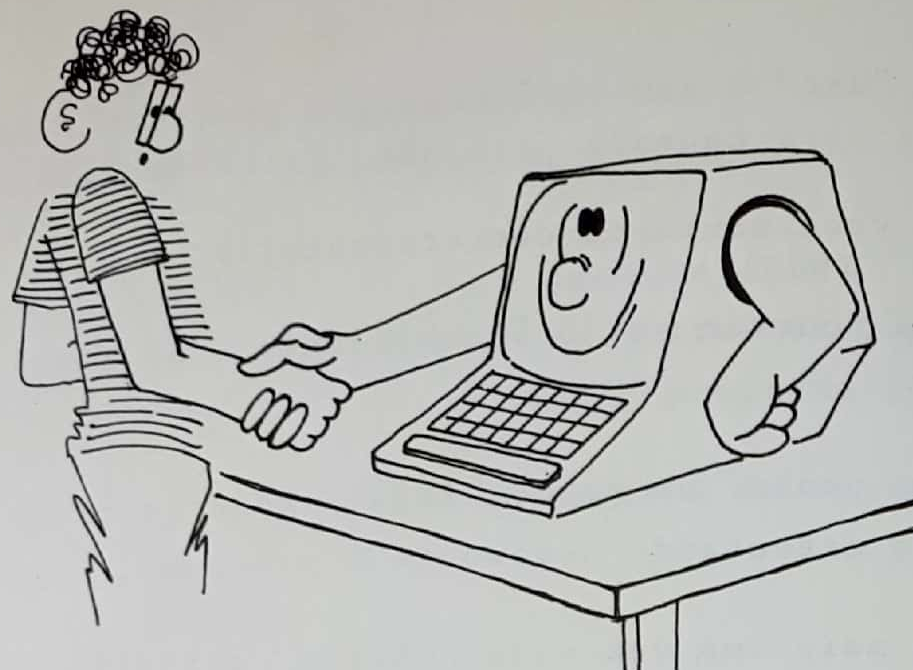


Figura 4.44 - "Não lamba os beiços porê não deixe de ponderar o quanto você melhorou tendo aprendido a linguagem BASIC e passado muitas horas com o seu micro".

Observação importante:

Aqui seria muito interessante que você utilizasse na dimensão horizontal a função

$$\text{INT}(X/16)*16-X$$

Neste caso teremos zero nessa expressão quando tivermos $X=0$, 16, 32, 48, 64, 80, 96 e 112.

Para a dimensão vertical a idéia melhor seria utilizar a função

$$\text{INT}(Y/6)*6-Y$$

e neste caso a mesma daria zero para $Y=0$, 6, 12, 18, 24, 30, 36 e 42.

Se trocarmos no Exemplo 4.63 a velha e já manjada linha 40 pela instrução

```
40 IF ((INT(X/16)*16-X)=0) OR ((INT(Y/6)*6-Y)=0) THEN 60
```

você criará um sofisticado tabuleiro de xadrez 8 por 8.

E assim se vai, se vai...

Vejamos agora uma outra aplicação dos operadores lógicos.

Exemplo 4.64

Programa "Inspeção antes de se meter na cama"

```
10 CLS
20 PRINT "RESPONDA ESTAS PERGUNTAS COM 'S' OU 'N'."
30 PRINT
40 INPUT "O GATO JÁ FOI COLOCADO PARA DENTRO"; A$
50 INPUT "O CÃOZINHO FILA TEM SEU OSSO PARA NÃO ROER A
   PORTA"; B$
60 INPUT "A LUZ DA PORTA FOI ACESA"; C$
70 INPUT "A LUZ DA SALA DE T.V, FOI APAGADA"; D$
80 INPUT "TODAS AS PORTAS E JANELAS ESTÃO FECHADAS"; E$
90 PRINT:PRINT:PRINT
100 IF A$="N" OR B$="N" OR C$="N" OR D$="N" THEN 120
110 PRINT "BOA NOITE E NÃO SONHE COM....": END
120 PRINT "ALGO NÃO FOI FEITO. NÃO VÁ PARA A CAMA"
130 PRINT "PENSE NO QUE VOCÊ ESQUECEU E RESOLVA A QUESTÃO!"
140 GOTO 40
RUN
```

Como se vê é a linha 100 que comanda cada resposta SIM ou NÃO e com isto o OR (ou) lógico desvia para a linha 120 em caso de uma "negativa" em qualquer um dos testes.

Poderíamos na saída ter algo do tipo:

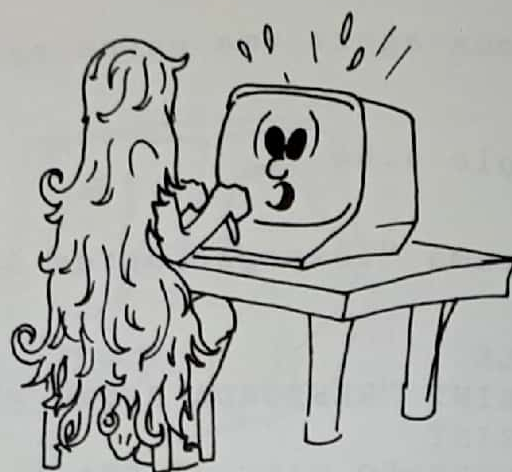
RESPONDA ESTAS PERGUNTA COM 'S' OU 'N'.

BOA NOITE E NÃO SONHE COM....
RESPONDA ESTAS PERGUNTAS COM 'S' OU 'N'.

ALGO NÃO FOI FEITO. NÃO VÁ PARA A CAMA
PENSE NO QUE VOCÊ ESQUECEU E RESOLVA A QUESTÃO!

Na maior parte dos casos os operadores AND e OR são intercambiáveis se outras partes ou melhor trechos do programa forem reescritos para acomodar a comutação.

Figura 4.45 - "É, o autor do livro está pior que a Lady Godiva, já mostrou quase tudo da linguagem BASIC e a você resta aproveitar...."



O OPERADOR NOT

Além dos operadores lógicos AND e OR tem-se também o operador lógico NOT (não).

Tem outros ainda, mas ficaremos por enquanto com estes três que já está bom demais...

Exemplo 4.65

```
10 CLS : LPRINT
20 INPUT "ENTRE COM UM NÚMERO"; K
30 L=NOT (K>6)
35 IF L=0 GOTO 50
40 LPRINT "K NÃO É MAIOR QUE 6" : LPRINT : LIST
50 LPRINT "K É MAIOR QUE 6" : LPRINT : LIST
RUN
```

Podemos ter uma das seguintes saídas:

K NÃO É MAIOR QUE 6

K É MAIOR QUE 6

Explicação sobre o Exemplo 4.65

Na linha 30 temos o operador lógico NOT.

Se a instrução na linha 30 for verdadeira mais precisamente que K não (NOT) é maior do que 6 o computador atribui a L o valor -1.

Neste caso falha o teste da linha 35 e imprime-se a mensagem da linha 40.

Se por outro lado, K é maior do que 6, a instrução da linha 30 é falsa e o micro faz com que o valor de L seja igual a zero.



Figura 4.46 - "Tcham, tcham, tcham, tcham - Se pegares o seu bondoso pai mexendo as escondidas no seu micro deixe-o, ele também merece"

TRABALHO PARA VOCÊ

- 1) Baseando-se no Exemplo 4.59 e o OR (ou) lógico escreva um programa que descreva o "status" da Mimosa conforme indicado na Figura 4.43 isto é determinado pela posição das portas X, Y e Z.
Olhe, hoje em dia até vaca tem "status"...

- 2) Entre com um número X e imprima "DENTRO DO INTERVALO" se X satisfaz a desigualdade $13 < X < 42$ e em caso contrário "FORA DO INTERVALO".

Usar obrigatoriamente o OR dentro do IF-THEN.

- 3) Químicos, físicos e as pessoas em geral estão a toda hora tomando ou analisando temperaturas dadas em alguma escala e convertendo as mesmas para outras escalas.

Na ciência as temperaturas são comumente registradas e manipuladas em cinco escalas:
Fahrenheit, Celsius (comumente chamada centígrada), Reaumur, Kelvin e Rankine.

Faça um programa que entrando com qualquer temperatura (acima do zero absoluto) em qualquer escala converta a mesma para as outras quatro escalas.

Chi, chi, chi, mas o que é temperatura absoluta?

Vá imediatamente consultar algum livro de Termologia antes de fazer o programinha...

- 4) Para animá-lo a usar os operadores lógicos repita o programa abaixo e brinque bastante com o alfabeto.

```
1 CLS
10 A$="ABCDEFGHIJKLMNOPQRSTUVWXYZ"
20 PRINT
30 PRINT "DIGITE UM NÚMERO DE 1 A ";LEN(A$);" "
40 PRINT "E EU DIREI QUAL A LETRA QUE CORRESPONDE A ESSA
   POSIÇÃO NO ALFABETO";
50 INPUT P
60 IF P>LEN(A$) OR P<1 THEN GOTO 20
70 PRINT
80 PRINT MID$(A$,P,1); "E A LETRA NÚMERO";P,"DO ALFABETO"
90 PRINT:PRINT
100 PRINT "DIGITE UMA LETRA E EU LHE DIREI"
110 INPUT "EM QUE POSIÇÃO ELA ESTÁ NO ALFABETO";X$
120 FOR N=1 TO LEN(A$)
130 IF MID$(A$,N,1)=X$ THEN GOTO 180
140 NEXT N
150 PRINT
160 PRINT "VOCÊ NÃO DIGITOU UMA LETRA DO ALFABETO!!":PRINT
170 GOTO 100
180 PRINT
190 PRINTX$,"E A LETRA NÚMERO";N;"DO ALFABETO"
200 PRINT
210 GOTO 20
RUN
```

- 5) No nosso vetusto livro de Cálculo Numérico estão indicadas várias fórmulas das somas das potências dos inteiros positivos.

Assim sabe-se que:

$$\sum_{x=1}^n x = 1 + 2 + 3 + \dots + n = \frac{n(n+1)}{2}$$

$$\sum_{x=1}^n x^2 = 1^2 + 2^2 + 3^2 + \dots + n^2 = \frac{n(n+1)(2n+1)}{6}$$

$$\sum_{x=1}^n x^3 = 1^3 + 2^3 + 3^3 + \dots + n^3 = \frac{n^2(n+1)^2}{4}$$

$$\sum_{x=1}^n x^4 = 1^4 + 2^4 + 3^4 + \dots + n^4 = \frac{n(n+1)(2n+1)(3n^2+3n-1)}{30}$$

$$\sum_{x=1}^n x^5 = 1^5 + 2^5 + 3^5 + \dots + n^5 = \frac{n^2(n+1)^2(2n^2+2n-1)}{12}$$

$$\sum_{x=1}^n x^6 = 1^6 + 2^6 + 3^6 + \dots + n^6 = \frac{n(n+1)(2n+1)(3n^4+6n^3-3n+1)}{42}$$

$$\sum_{x=1}^n x^7 = 1^7 + 2^7 + 3^7 + \dots + n^7 = \frac{n^2(n+1)^2(3n^4+6n^3-n^2-4n+2)}{24}$$

Este programa você pode ampliar até as potências de ordem 10 porém até a potência 7 é suficiente por ora, aliás essa é a opinião do Sacha e a nossa...



Figura 4.47

A hora e a vez de alguns detalhes sobre o sofisticado MAXXI

§5.1 - GRÁFICOS DE BAIXA RESOLUÇÃO

Neste capítulo daremos alguns detalhes sobre a utilização de um excelente microcomputador nacional no modo gráfico tanto de baixa resolução como no de alta resolução.

Referimo-nos ao MAXXI feito pela POLYMAX SISTEMAS E PERIFÉRICOS S/A.

Aliás repetindo o que aparece no manual do MAXXI, a princípio, como em qualquer atividade nova, como dirigir um carro, você estranhou um pouco e cometeu algumas "barbeiragens", porém ao chegar neste ponto você já é quase um piloto de Fórmula-I.

Não se assuste com essa comparação!!!

A diferença entre o aprendizado do qual você está participando e o outro com carros é que neste último qualquer erro pode ser fatal (principalmente se você se tornar piloto da Fórmula-I).

Ao se aprender programação em linguagem BASIC você não corre o risco de morrer eletrocutado e o máximo que acontece é a sinalização no seu aparelho de T.V. dos erros que você próprio comete.

Porém para que você seja um programador do tipo piloto de Fórmula-I ou seja um programador-"master" é necessário que você se torne íntimo de um micro como o MAXXI.

Bem vindo assim ao mundo da criação de maravilhosos desenhos e quaisquer outras representações gráficas.

O MAXXI utiliza a linguagem de programação POLYSOFT BASIC que é conhecida pelo nome BASIC (nossa velha conhecida) com algumas pequenas modificações em relação ao que aprendemos até agora e com muitas implementações.

Evitemos a perda de tempo, liguem-nos o mais rapidamente possível no MAXXI ou melhor deixemo-lo entrar em ação.

A sua fotografia (veja a Figura 5.1) já nos dá uma idéia que ele deve ser sensacional não é?



Figura 5.1 a

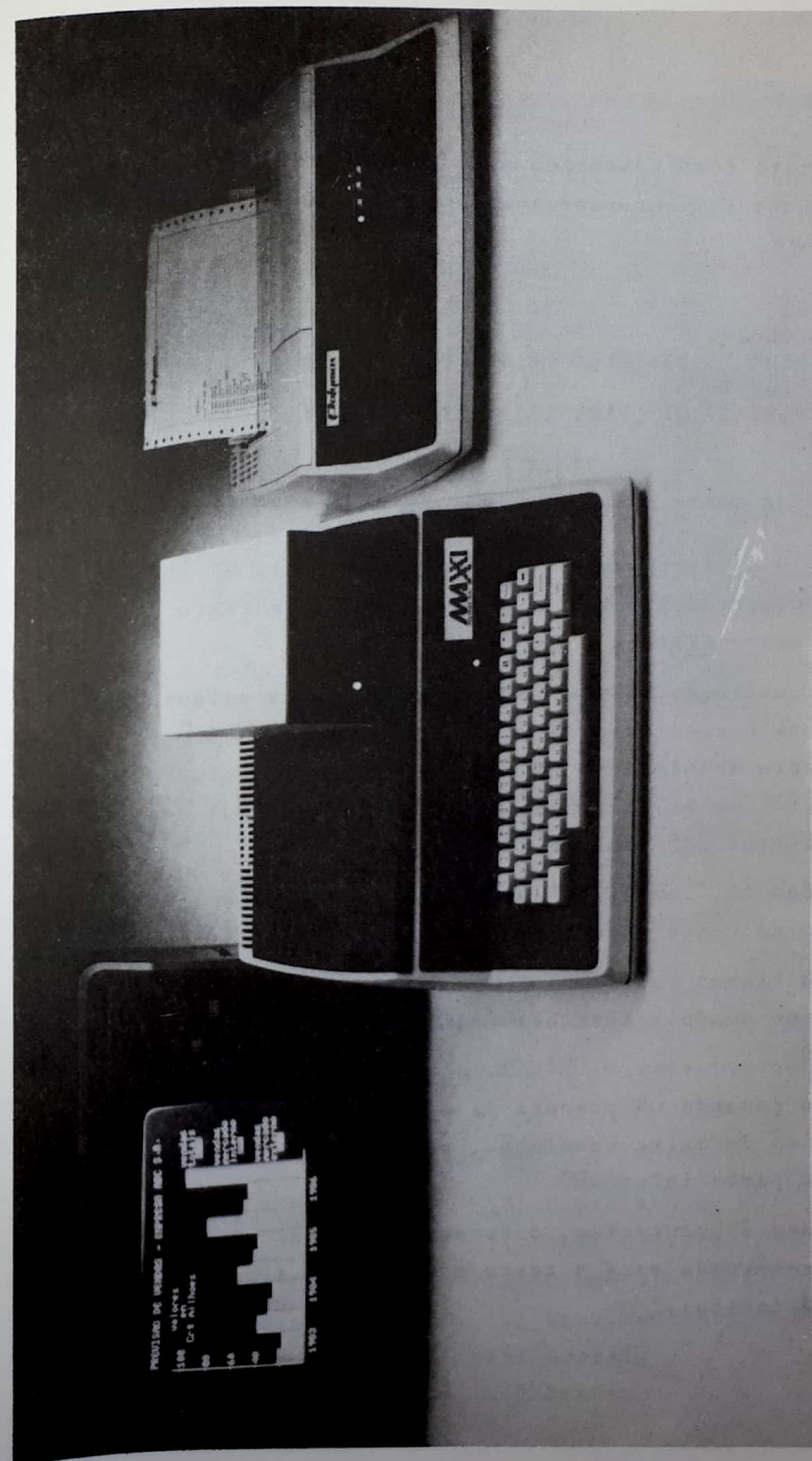


Figura 5.1 b

Exemplo 5.1

Inicialmente como homenagem sua, caro leitor, ao autor deste livro que tal entender como pode ser impressa e a cores a letra V

```
10 GR
20 COLOR=13
30 PLOT 13,30: PLOT 14,31: PLOT 15,32
40 PLOT 16,33
50 PLOT 17,32: PLOT 18,31: PLOT 19,30
RUN
```

Comentários sobre este programa

- 1) **GR** - A instrução de duas letras apenas, GR indica ao microcomputador para que passe do modo de texto (TEXT) para o modo GRáfico

Já a instrução TEXT (utilizaremos a mesma adiante) faz exatamente o contrário ou seja através dela informa-se ao micro para abandonar o modo gráfico ou seja voltar ao modo não gráfico ou seja modo TEXT (texto) no qual tem-se 40 caracteres por linha e 24 linhas.

No modo GR, tem-se 40 colunas numeradas de 0 até 39 começando no canto superior da esquerda (veja a Figura 5.2).

As 40 linhas são também numeradas de 0 até 39 com a linha de topo sendo a zero (0) e a linha inferior a de número 39.

Com o comando GR prepara-se o vídeo para imprimir no modo gráfico de baixa resolução, reservando 4 linhas para texto na parte inferior.

O vídeo é escurecido, o cursor é colocado na parte do vídeo reservada para o texto e COLOR é assinalada com o valor 0 (preto).

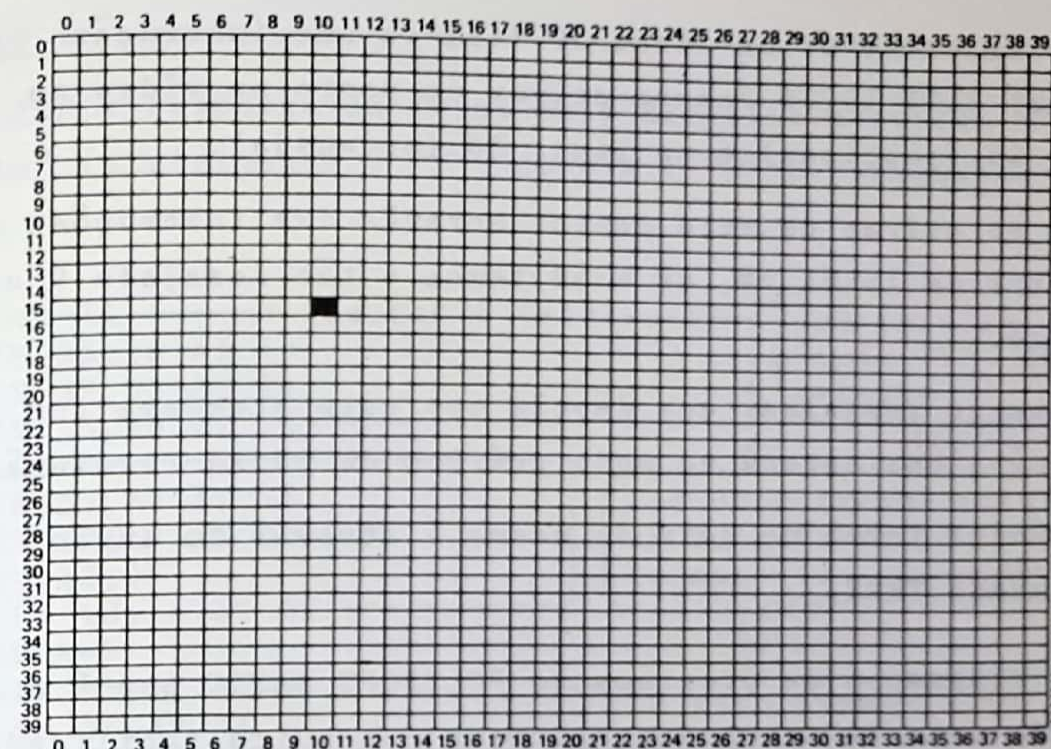


Figura 5.2 - Se escrevermos PLOT 10,15 isto fará com que o micro ative um ponto no grid onde a coluna 10 é cruzada pela linha 15.

- 2) **COLOR=N** - Este comando indica que a cor de número "N" será usada pelo próximo comando de impressão.

Assim no caso do Exemplo 5.1 COLOR=13 indica que a próxima impressão será feita na cor água marinha.

Que linda cor hein Sacha?

A sua cor (COLOR=7) é amarelo canário.

Aliás os nomes das cores no MAXXI e seus correspondentes números são:

0 - Preto	9) Azul
1 - Azul Anil	10) Cinza Escuro
2 - Verde Escuro	11) Azul Celeste
3 - Violeta	12) Creme
4 - Verde Floresta	13) Água Marinha
5 - Cinza Claro	14) Amarelo
6 - Verde Claro	15) Branco
7 - Amarelo Canário	
8 - Azul Marinho	

3) Ao se escrever na linha 30 (tem-se uma instrução múltipla) PLOT 13,30 faz-se ativar um ponto no vídeo na coluna 13 (abscissa) e na linha 30 (ordenada).

Ao se ativar os sete pontos através das instruções de números de linha 30, 40 e 50 temos o tão desejado V estilizado.

Deve-se ressaltar que depois que este programa for rodado você ainda estará no modo gráfico de baixa resolução.

A única forma de sair do mesmo é através do envio de um comando TEXT.

Na linguagem de programação POLYSOFT BASIC que é a utilizada pelo MAXXI podemos traçar com muita facilidade linhas horizontais e verticais quando estivermos no modo gráfico de baixa resolução.

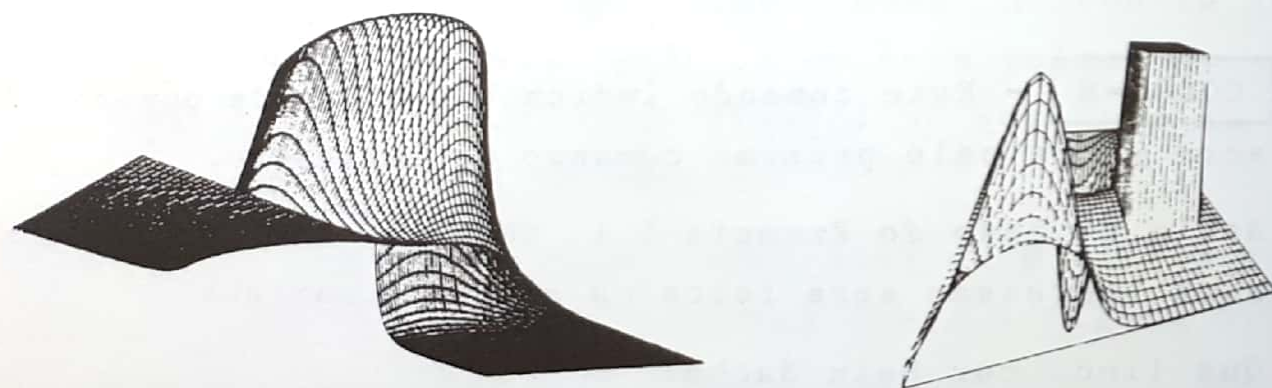


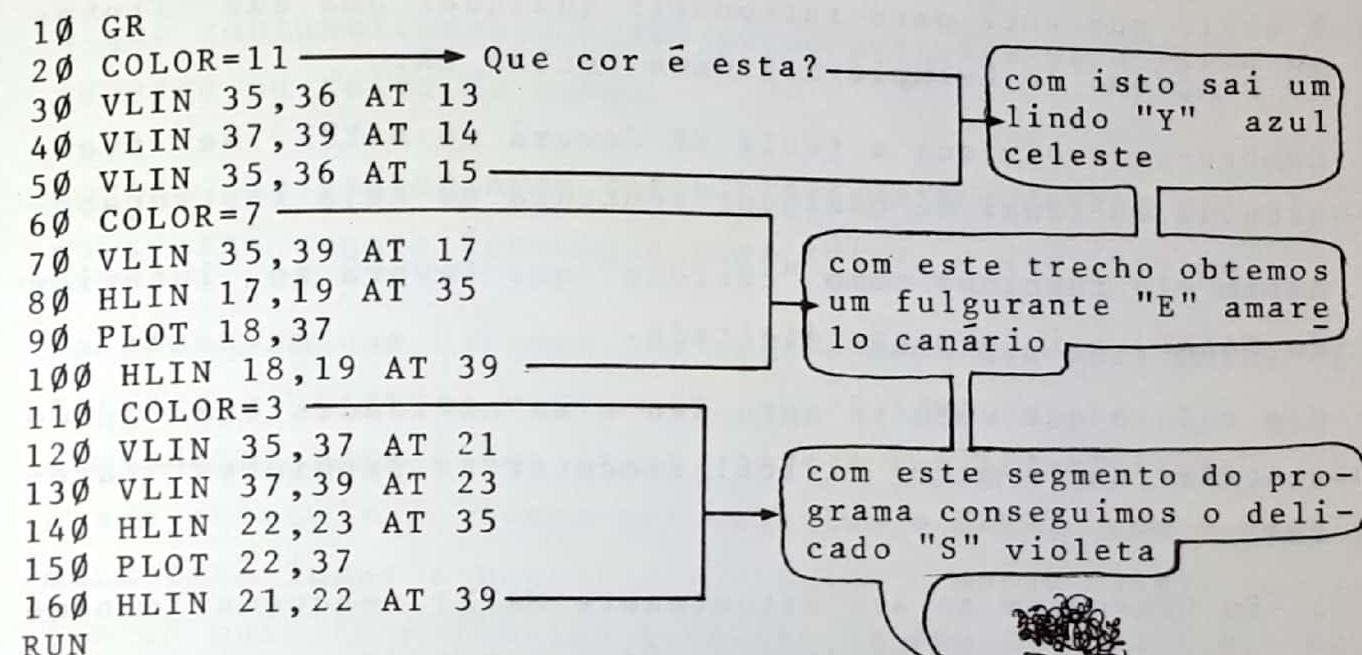
Figura 5.3 - "Desenhos deste tipo são produzidos pelo microcomputador e o limite está condicionado a imaginação e a habilidade do leitor em produzir os convenientes programas".

Exemplo 5.2

É muito comum você ser positivo e nesse caso para dar uma de políglota você diz "YES" e levanta para cima o polegar... Desenhe esta palavra na tela do seu vídeo com o auxílio do maravilhoso MAXXI.

Solução :

Vamos dar a resposta da forma mais didática possível "construindo" inicialmente o "Y", depois o "E" e finalmente o "S".



Comentários

1) Para facilitar o traçado de linhas horizontais e verticais usam-se as instruções HLIN e VLIN respectivamente.

Assim se quisermos uma linha horizontal da coluna 0 até a 39 na posição 13 basta escrevermos

HLIN 0,39 AT 13

Por outro lado se quisermos uma coluna vertical da linha 0 até a 39 na posição 7 basta escrevermos

VLIN 0,39 AT 7

Para se obter linhas de menor comprimento basta tomar números que estejam entre 0 e 39.

Dessa forma se escrevermos:

HLIN 13,27 AT 7

imprimiremos uma linha horizontal no modo gráfico de baixa resolução usando a cor mais recentemente especificada por COLOR a partir de X=13 até X=27 e tomando Y=7.

Em outras palavras a linha é impressa a partir do ponto (13,7) até o ponto (27,7) em termos de coordenadas cartesianas.

- 2) É óbvio que você para introduzir qualquer uma das linhas do programa do Exemplo 5.2 deve teclar CR.

Lembre-se assim que a tecla CR deverá no MAXXI ser pressionada ao final de qualquer sentença ou seja instrução.

Assim ela funciona como "correio" que levará ao interior do MAXXI a informação digitada.

Com tudo o que você já aprendeu e as novidades ora apresentadas não é muito difícil executar as seguintes tarefas

1. Em homenagem ao seu estonteante MAXXI escreva o nome do próprio na tela do seu vídeo com todas as letras tendo cores diferentes.

2. Faça um programa que permita a saída das palavras MUTT E JEFF sendo que o MUTT tem que sair verde, o E pode ser branco e o JEFF amarelo.

Existem vários tons de verde e de amarelo que você pode escolher para dar vazão a sua criatividade...

As letras das palavras não devem ter comprimento maior que 7 unidades quando se usar o modo gráfico de baixa resolução.

- 3) Já que no Exemplo 5.2 fizemos o "YES" como bom poliglota você deve também fazer o "NO".

Faça ele aparecer na tela no maior tamanho possível e em duas cores de sua escolha.

- 4) Escreva um programa que desenhe uma bandeira semelhante a bandeira americana e que ocupe toda a tela do vídeo.

Não é necessário que tenha as mesmas cores que a bandeira dos Estados Unidos.

OBSERVAÇÃO IMPORTANTE

O famoso educador A.S.Neill no seu livro "Liberdade sem medo - Summerhill" diz que não existe *criança* ou pessoa *preguiçosa*.

O que costumeiramente chama-se de preguiça ou é falta de interesse ou falta de saúde.

No caso de uma criança sadia ela jamais pode ser preguiçosa pois está sempre fazendo alguma coisa, o dia inteiro.

Na época em que vivemos não existe nada que gere maior interesse a crianças e pessoas que o microcomputador e nós recomendamos para o seu lazer, para a sua criatividade e para a sua economia de tempo nos cálculos o MAXXI com o qual já mais terá lugar a preguiça visto que é nosso desejo que sejam as mais transitórias e curtas as épocas em que o caro leitor ou leitora esteja acometido(a) de alguma doença.



Figura 5.4 - "É essa a vida que o autor lhe deseja durante muito tempo após um salutar trabalho no seu micro".

Exemplo 5.3

Vamos agora desenhar uma moldura quadrada com cada lado tendo um comprimento S e estando a mesma centrada no ponto (X, Y) como está indicado na Figura 5.5.

Faremos um programa que desenhe molduras de diferentes cores, tamanhos e posições.

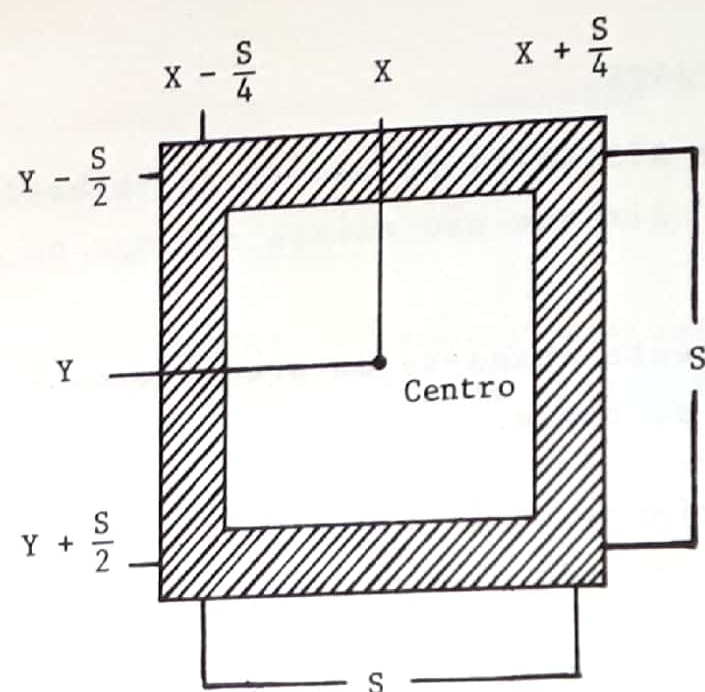


Figura 5.5

Solução:

```

10 REM VAMOS DESENHAR MOLDURAS "QUADRADAS"
20 HOME
30 GR
40 INPUT "ENTRE COM UMA COR, (1-15)"; C
50 COLOR=C
60 INPUT "ENTRE COM O CENTRO DA SUA MOLDURA X,Y (10-29)";
  X,Y
70 INPUT "ENTRE COM O COMPRIMENTO DO LADO S (2-20)"; S
75 REM FATOR DE ESCALA APROXIMADAMENTE 1/2
80 HLIN X-S/4, X+S/4 AT Y-S/2
90 VLIN Y-S/2, Y+S/2 AT X+S/4
100 HLIN X+S/4, X-S/4 AT Y+S/2
110 VLIN Y+S/2, Y-S/2 AT X-S/4
120 HOME
130 GOTO 40
RUN

```

Como se vê neste programa aparece na linha 20 a instrução HOME com a qual se move o cursor para o canto superior esquerdo do vídeo sem o espaço reservado para o texto e limpa-se todo o vídeo.



Figura 5.6 - "E vale a pena apagar o que está na tela do vídeo?"

pode ser que você queira interromper a execução ou a listagem de um programa,

Para esta finalidade digite **CTRL C**.

Isto causará o aparecimento do caractere de prontidão ("]") e do cursor.

O caractere de prontidão indica que você pode continuar digitando informações (ou comandos) para o micro.

É bem por isso que ele é chamado *caractere de prontidão*.

Uma vez que o processamento é interrompido, pode ser reiniciado.

É só você digitar

RUN CR

Resumindo, digite CTRL C para interromper o processamento e RUN CR para reiniciá-lo.

Não esqueça este conselho: para se ter sucesso é necessário ter a capacidade de trabalhar alegremente e de viver positivamente.

O domínio de um micro como o MAXXI lhe trará sem dúvida a condição de trabalhar alegremente e restará a você o complemento de viver positivamente.

E dominando o micro você terá meio caminho andado.

Vamos agora utilizar a função PDL para desenhar figuras semelhantes a um tabuleiro de xadrez.

Uma função em POLYSOFT BASIC, é alguma coisa que utiliza um ou mais números e que quando executada apresenta resultados diferenciados.

A esses números que são utilizados pelas funções, chamamos (como já sabemos) de ARGUMENTO e são colocados sempre entre parênteses depois do nome da função.

PDL é uma função do POLYSOFT BASIC que tem somente um argumento.

Se tivermos por exemplo a função PDL(1) ou PDL(0) ela retorna um número entre 0 e 255 que é o valor corrente do cabo de controle de jogos.

Um possível programa para se obter um tabuleiro de xadrez na tela do seu vídeo está no Exemplo 5.4.

Exemplo 5.4

```

10 REM OBTENÇÃO DO TABULEIRO DE XADREZ
20 GR : HOME
30 INPUT "ENTRE COM DUAS CORES"; C1,C2
40 X=PDL(0)
45 X=INT(X/7)
50 Y=PDL(1)
55 Y=INT(Y/7)
60 Z=X+Y
70 R=Z-INT(Z/2)*2
80 IF R=0 THEN COLOR=C1 : GOTO 100
90 COLOR=C2
100 PLOT X,Y
110 GOTO 40
RUN

```

Comentários sobre este programa

Na linha 30 o micro pede ou seja pergunta a você quais são as duas cores que serão usadas no seu tabuleiro.

Nas linhas 60 e 70 verifica-se se X+Y é par ou ímpar.

Caso ele seja par (R=0) então a cor que se usa na linha 80 é a C1 e em caso contrário utiliza-se a cor dada pelo número C2 (linha 90).

O ponto colorido é desenhado com a indicação da linha 100.

O programa então se desvia ou seja volta para a linha 40 onde as funções PDL entram novamente em ação...

Vejamos como se podem obter os números aleatórios que alguns chamam também de "randômicos" pois aleatório em inglês é RANDOM.

Exemplo 5.5

No POLYSOFT BASIC que o MAXXI usa ao escrevermos

```

10 PRINT RND(1)
20 GOTO 10
RUN

```

pode-se ter na saída o seguinte

```

.103456781
.987445213
.702513185
.809248735
.547133289
.321485127
.231665158 —————> Apertamos aqui CTRL C

```

Como se vê os números gerados por este programa são frações decimais entre zero que pode estar incluído e o um que não está incluído.

A importância da função RND já foi descrita nos capítulos anteriores e apenas destacaremos que no POLYSOFT BASIC do MAXXI se escrevermos RND(13) ou RND(8) na linha 10 teremos a geração de números aleatórios entre o zero e o um como anteriormente.

Porém se escrevermos RND(0) ele retorna apenas o número aleatório gerado mais recentemente.

Para cada argumento negativo, a função RND (argumento) gera um número particular, que é o mesmo, cada vez que RND é executado com esse argumento.

Subsequentes comandos RND com argumentos positivos seguirão a mesma sequência para cada argumento.

Cada vez que RND é usado com um argumento positivo, um novo número aleatório de 0 (incluído) a 1 (excluído) é gerado, a menos que uma instrução com a função RND com argumento negativo tenha sido dada anteriormente.

Exemplo 5.6

Aí vai um tabuleiro usando a função RND.

```

10 REM DESENHO DE UM TABULEIRO DE XADREZ COM CORES ALEATÓRIAS
15 REM NÃO CONFUNDIR COM ALGUM TRABALHO DE PICASSO
20 GR : HOME
30 INPUT "ENTRE COM DUAS CORES"; C1,C2
40 Y=0
45 X=0
50 R=RND(1)
60 IF R < .5 THEN COLOR=C1 : GOTO 80
70 COLOR=C2
80 PLOT X,Y
85 X=X+1
90 IF X <= 39 THEN 50
97 Y=Y+1
100 IF Y <= 39 THEN 45
110 INPUT "QUERES UM OUTRO E MARAVILHOSO TABULEIRO?
(SIM OU NÃO)"; A$
120 TEXT : HOME
130 IF A$ = "S" THEN 20
140 END
RUN

```

"QUE ENGRAÇADO,
OBTIVE O MESMO
NÚMERO QUE TIREI
NA ÚLTIMA VEZ!"



Figura 5.7

→ Você agora está apto a fazer o seguinte programa de tiras coloridas.

Entre através do teclado com as seguintes informações:

- 1) o número de tiras, N, que você quer colorir;
- 2) a largura de cada tira, L;
- 3) o comprimento de cada tira, D;
- 4) duas cores C1 e C2 com as quais as tiras são formadas.

Será que depois de ter feito o seu programa de tiras você não pode modificá-lo para que novamente se tenha a bandeira de um importante país da América?

Exemplo 5.7

Bem agora realmente você vai "endoidar" de amores pelo seu MAXXI pois usando um pouco os programas dos Exemplos 5.1 e 5.3 vamos desenhar a "cara" estilizada de um E.T. a cores e encher a tela do seu vídeo com a mesma.

```

10 REM FIGURAS MÚLTIPLAS - PARECE A CARA DE UM E.T.
20 GR : COLOR=13
30 FOR Y=0 TO 30 STEP 15
40 FOR X=0 TO 30 STEP 15
50 GOSUB 500 : NEXT X : NEXT Y
60 END
500 HLIN X,X+9 AT Y
510 VLIN Y+1, Y+8 AT X+9
520 HLIN X+9, X AT Y+9
530 VLIN Y+8, Y+1 AT X
540 PLOT X+3, Y+2 : PLOT X+6, Y+2
550 HLIN X+4, X+5 AT Y+4
560 PLOT X+2, Y+5 : PLOT X+3, Y+6
570 HLIN X+4, X+5 AT Y+7
580 PLOT X+6, Y+6 : PLOT X+7, Y+5
590 RETURN
RUN

```

Como é você conseguiu na tela do seu vídeo as nove "fotografias" do seu E.T.?

Parabéns em caso afirmativo.

TRABALHO PARA VOCÊ

- 1) Faça um programa com o qual na tela do seu vídeo saiam vários quadrados concêntricos com uma cor escolhida a sua vontade.

- 2) Faça um programa que desenhe no centro da tela uma cruz azul de 7 por 7.

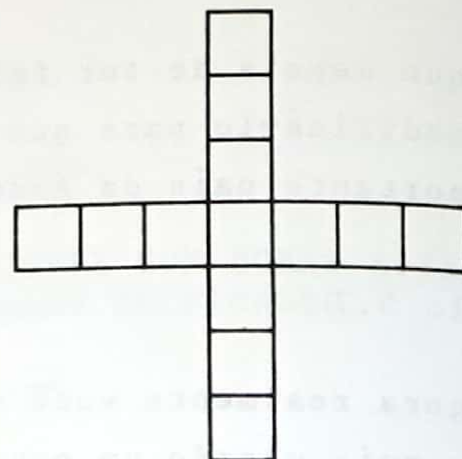


Figura 5.8

- 3) Faça um programa com o qual obtenha na tela do seu vídeo o desenho da Figura 5.9 com as partes a), b), c), d) e e) tendo cores diferentes.

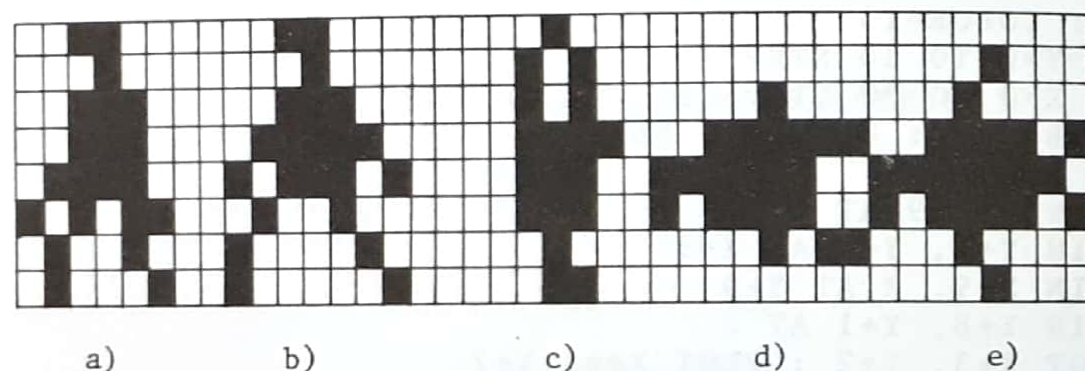


Figura 5.9

Exemplo 5.8

Vamos agora desenvolver um programa que nos forneça um gráfico de hastes múltiplas de dados econômicos indicados na Tabela 5.1 para um "certo país".

Dados \ Ano	1980	1981	1982	1983	1984 (projeção)
Inflação (I) (%)	47	68	90	133	185
Desemprego (D) (%)	8,1	9,2	10,2	11,5	12,8
Crescimento do PNB (C) (%)	5,1	4,5	3,2	1,3	0,2
Variação no ganho individual (%)	3,9	4,3	3,1	0,3	-0,3

Tabela 5.1 - Dados econômicos

Na execução do gráfico devem ser usadas as seguintes cores (gráfico de baixa resolução).

Característica	Número	Cor
Inflação	3	Violeta
Desemprego	7	Amarelo canário
Crescimento do P.N.B.	9	Azul
Variação no ganho individual	2	Verde escuro

Tabela 5.2



Figura 5.10 Aí vai o programa para esses dados econômicos

```

5 REM GRÁFICO PARA DADOS ECONÔMICOS
10 GR
15 C1 = 3: REM VIOLETA
17 C2 = 7: REM AMARELO CANÁRIO
19 C3 = 9: REM AZUL
20 C4 = 2: REM VERDE ESCURO
21 DATA 47,8.1,5.1,3.9
22 DATA 68,9.2,4.5,4.3
24 DATA 90,10.3,3.2,3.1
26 DATA 133,11.5,1.3,.3
28 DATA 185,12.8,0.2,-.3

65 X = 3
70 FOR J = 1 TO 5
80 GOSUB 300: REM DESENHO DAS QUATRO HASTES
90 NEXT J
100 GOSUB 500: REM IMPRESSÃO DE CABEÇALHOS
132 PRINT: INVERSE: SPEED = 75
133 PRINT "ESCALAS: (INFLAÇÃO 1/5 - OUTROS 2/1)"
134 PRINT "I-VIOLETA, D-AMARELO, PNB-AZUL, G-VERDE";
135 SPEED = 255
136 GOTO 136

```

→ Este é o programa principal


```

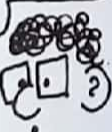
300 REM DESENHO DAS QUATRO PRÓXIMAS HASTES
310 READ I,D,C,V
320 Z=I/5: COLOR=C1
330 GOSUB 600: REM DESENHO DAS HASTES
340 X=X+1
350 Z=2*D: COLOR=C2
360 GOSUB 600
370 X=X+1
380 Z=2*C: COLOR=C3
390 GOSUB 600
400 X=X+1
410 Z=2*V: COLOR=C4
420 GOSUB 600
430 X=X+4
440 RETURN
500 REM SAÍDA DOS ANOS E OUTROS TÍTULOS
510 HOME
520 PRINT " 1980 1981 1982 1983 1984"
560 RETURN
600 REM DESENHO DE HASTES DE COMPRIMENTO Z
610 L=INT(ABS(Z) + .5)
620 IF L=0 THEN RETURN
630 IF Z<0 THEN 670
640 Y1=37
650 VLIN Y1,(Y1 - L + 1) AT X
660 GOTO 690
670 Y1=38
680 VLIN Y1,Y1 + L - 1 AT X
690 RETURN
RUN

```

Esta é a subrotina para desenhar as quatro próximas hastes do gráfico



Esta é a subrotina para a saída dos títulos



Esta é a subrotina para representar uma haste de comprimento Z



Figura 5.11 - "Com a saída do Exemplo 5.8 até o micro pensa na solução radical..."

TRABALHO PARA VOCÊ

- 1) Desenhe um cavalo azul com cara branca e patas creme (veja a Figura 5.12).

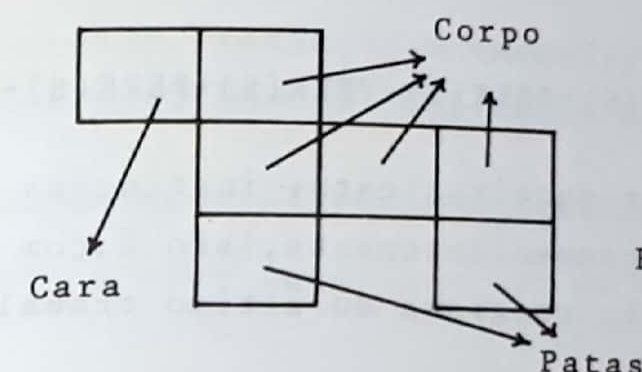


Figura 5.12

você não acha que isto é um cavalo?
Pois bem, aceite como tal.

- 2) Utilizando o programa do exercício 1 e a noção de subrotina coloque 2 cavalos na tela do seu vídeo.

- 3) Elabore um programa que faça um ponto colorido, (para você pode ser uma bola) mover-se através da tela do vídeo, à direita e à esquerda.

Não deixe nunca de lado a sua criatividade e assim não é proibido que neste programa você seja extremamente "colorido" fazendo a sua bola mudar de cor a sua vontade.

- 4) Usando o programa do exercício 3 faça como complemento a sua bola bater nas 4 "paredes" da tela do seu vídeo.

Não é proibido também neste caso que você mude o padrão dos saltos da sua bola e nem tão pouco que a bola ao bater na "parede" produza algum som.

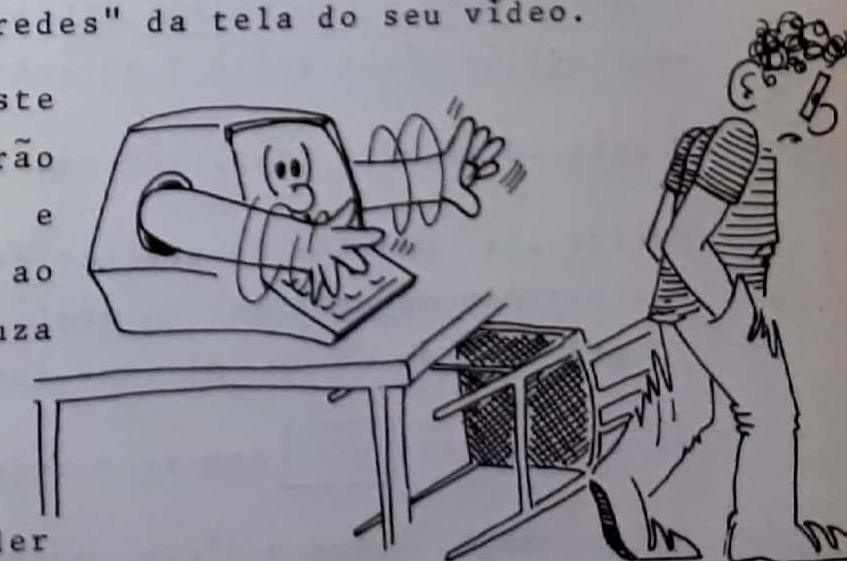


Figura 5.13 - "Se algo der errado, não desista, tente de novo..."

OBSERVAÇÃO IMPORTANTE

Para que se tenha um agradável e ressonante som no MAXXI de ve-se usar as instruções mágicas

```
10 S = -16336
20 SOM=PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)+PEEK(S)-PEEK(S)
```

Cabe a você caro leitor encaixar estas instruções (de números de linha 10 e 20) convenientemente, isto é, com os números de linha corretos no seu programa do último trabalho (aquele de nº 4).

Aliás no MAXXI diferentes números de PEEKs numa instrução produzem qualidades diferentes de sons.

§5.2 - GRÁFICOS DE ALTA RESOLUÇÃO

Vejamos finalmente alguma coisa sobre a alta resolução gráfica visto que até agora só usamos *gráficos de baixa resolução*.

No gráfico chamado de *alta resolução* a coordenada horizontal começa com 0 (zero) no lado esquerdo do vídeo e termina com 279 à direita.

Do mesmo modo a coordenada vertical vai de 0 (zero) na parte superior do vídeo até 159 na inferior.

O gráfico de alta resolução não é difícil de entender.

Muitos dos comandos do gráfico de alta resolução são os mesmos correspondentes aos do gráfico de baixa resolução, exceto pela adição de um "H" (que indica alta-resolução).

Ao se digitar HGR tem-se o modo gráfico de alta resolução.

Este comando limpa o vídeo, deixando-o preto e permitindo quatro linhas na parte inferior da tela do seu vídeo para

texto.

Com o HGR 2 dá para preencher totalmente o vídeo no modo gráfico de alta resolução (280 x 192).

O gráfico de alta resolução é verdadeiramente maravilhoso, entretanto você terá que se sujeitar a um certo sacrifício...

Existem poucas cores disponíveis neste tipo de gráfico.

As cores para a alta resolução vão de 0 (preto) até 7 (branco).

Assim as cores possíveis que você pode usar no MAXXI são:

0 - Preto 1	4 - Preto 2
1 - Creme	5 - Azul
2 - Violeta	6 - Verde
3 - Branco 1	7 - Branco 2

Estas cores variam de TV para TV e de acordo com a sua posição no vídeo.

A instrução para representar, desenhar ou seja PLOTar em modo gráfico de alta resolução é HPLOT e a cor é indicada por HCOLOR.

Exemplo 5.9

Usando o modo gráfico de alta resolução vamos obter um quadrado cujo centro está no ponto (133,87) e cujos lados tem 80 unidades de comprimento.

```
10 REM DESENHO DE UM QUADRADO
20 HGR : HCOLOR=2
30 GOSUB 300 : REM O USO DE VARIÁVEL INDEXADA
40 XCQ=133 : YCQ=87
50 HPLOT XCQ+X(0), YCQ+Y(0)
60 FOR I=1 TO M
70 HPLOT TO XCQ+X(I), YCQ+Y(I)
80 NEXT I
90 END
```



```

300 REM USANDO AS VARIÁVEIS COLETIVAS
310 M=4 : DIM X(M), Y(M)
320 FOR I=0 TO M: READ X(I) : NEXT I
330 FOR I=0 TO M: READ Y(I) : Y(I)=-Y(I): NEXT I
340 RETURN
350 DATA -40,40,40,-40,-40
360 DATA 40,40,-40,-40,40
RUN

```

Comentários: No POLYSOFT BASIC do MAXXI todas as variáveis tem o valor zero até que você guarde um valor no seu conteúdo.

Uma variável pode ter qualquer nome desejado, desde que comece com uma letra.

Alguns nomes que dizem respeito à própria linguagem POLYSOFT BASIC não são permitidos e assim não podemos representar uma variável pelo nome PRINT, DATA, INPUT, etc.

A estas palavras denominamos PALAVRAS RESERVADAS.

Da mesma forma não podemos ter nomes de variáveis que em seu interior possuam *palavras reservadas*.

Toda vez que isto acontecer, POLYSOFT BASIC lhe enviará uma mensagem de ERRO.

Não se aborreça muito, pois bastará mudar o nome da variável.

Caso você escreva

```

10 SERGIO=13
20 PRINT SERGIO
RUN

```

Na saída ter-se-á 13

Se acrescentar agora a esse programa

```

30 PRINT SERTÃO

```

e der o RUN novamente na saída ter-se-á 13.

O que aconteceu? Será que o 13 é um número "cabalístico"? Não, não é nada disso...

O que realmente ocorre é que o POLYSOFT BASIC utiliza somente os primeiros dois caracteres de cada nome de variável para distinguir uma da outra.

Dessa forma, todas as variáveis que iniciem com os mesmos dois primeiros caracteres terão o mesmo conteúdo.

Na linha 40 do Exemplo 5.9 definimos o centro do quadrado como estando no ponto (133,87) e é importante notar que a letra (Q) não tem nenhuma função como diferenciadora de uma variável da outra.

Na linha 50 temos uma instrução que faz com que se ative um ponto no canto superior a esquerda.

Na linha 70 temos a instrução HPLOT TO com a qual se pode desenhar linhas de um ponto do vídeo a outro.

Assim por exemplo se quiséssemos desenhar uma linha na margem superior do vídeo basta digitar

```
HPLOT 0,0 TO 279,0
```

Se você quiser desenhar uma linha de um canto no ponto 279,0 ao outro do vídeo, digite

```
HPLOT TO 279,159
```

e a linha aparece ao longo da margem direita do vídeo.

Observe que quando você usa esta última instrução, a nova linha parte do ponto onde parou a linha traçada anteriormente e utiliza a mesma cor escolhida quando se traçou a linha anterior.

Você pode encadear estes comandos e traçar várias linhas com uma mesma sentença.

Bem, acho que com tudo isto já deu para entender a linha 70.

Deve-se observar também que na linha 230 cada valor armazenado em Y(I) tem o seu sinal trocado.

A razão disto está ligada com o sentido do eixo Oy na tela do seu vídeo.

Não entendeu isto?

Bem, pense um pouco que você chega lá, tá...

TRABALHO FINAL, REALMENTE FINAL PARA VOCÊ

- 1) No Exemplo 5.9 dissemos que você iria obter um quadrado.

Realmente saiu um quadrado ou um retângulo?

No caso de ter saído um retângulo você poderia dizer por quê?

Dá para corrigir o programa do Exemplo 5.9 e ter realmente um quadrado?

- 2) Usando o Exemplo 5.9 e as correções que você fez em vista do exercício 1 faça um programa que permita a saída de um conjunto de quadrados concêntricos.

- 3) Elabore um programa com o qual se obtenha um círculo de raio R usando o modo gráfico de alta resolução.

- 4) Agora que você já é um "monstro geométrico" (estou supondo que conseguiu fazer os exercícios 1, 2 e 3 deste último trabalho) elabore um programa que permita desenhar um polígono. É, não é muito fácil, mas não custa você fazer alguns calos na região glútea pois em caso de sucesso valerá a pena.

- 5) Finalmente, uma distração e espero que você consiga fazer a mesma em cores.

"Estrela" é um jogo de adivinhação.

Aí vão as regras.

O micro pensa em um número de 1 a 250.

Você tenta adivinhar o número.

Após o seu chute o micro imprime uma ou mais estrelas que para você são asteriscos (*).

Quanto mais próximo você estiver do número que o micro escolheu, mais estrelas ou asteriscos ele imprimirá.

Um asterisco (*) significa que você está muito longe do número que o MAXXI escolheu.

Por outro lado (*****) significa que você está muito próximo do número que o MAXXI tem "escondido".

Permite-se não mais que seis chutes e você deve receber uma mensagem de congratulações caso acerte até seis tentativas ou então uma manifestação sincera, porém desencorajadora sobre os seus atributos matemático-visionários.

Tchau, tchau, aqui acabou o repertório do BASIC.

Porém espere mais um pouco e terá a oportunidade de se encontrar conosco em um livro dedicado exclusivamente ao MAXXI.

Até breve!!!



Figura 5.14 - "Acabou por ora o repertório do BASIC..."

O microcomputador e a educação

§6.1 - A ÉPOCA DOS MICROS E O NOSSO SISTEMA EDUCACIONAL VIVENDO AINDA NA ÉPOCA PRÉ-GUTENBERG

Em muitos estados do Brasil por incrível que possa parecer nem os professores e nem os alunos atingiram ainda a "galáxia de Johannes G. Gutenberg" ou seja o uso de um bom livro texto.

No nosso sistema educacional, na maior parte das vezes, aparece ainda o professor emitindo conceitos oralmente como se fazia na época pré-histórica.

Basicamente o professor em muitas situações não ensina mas apenas ajuda o aluno a aprender.

Sem dúvida nenhuma o desafio é a melhor forma didática para o desenvolvimento intelectual.

Ensinar é somente estimular ao desafio, adequada e gradualmente.

Acreditamos que a linguagem BASIC permitindo o uso do micro na Educação irá de fato acabar com as famosas "aulas de salivação".

Passamos a maior parte do tempo restritos às paredes de uma sala tendo a nossa frente um quadro negro e um mestre geralmente agitado falando ou escrevendo sem parar...

Esta é a imagem real do anacronismo.

A aula assim como a classe, remonta à Idade Média quando o docente era o único "Informador" disponível visto que tão pouco existiam livros.

Parece até que o nosso professor se assemelha ao famoso cão de Pavlov que condicionado ao toque de uma campainha começa a salivar.

O livro foi a grande revolução tecnológica do sistema escolar.

Entretanto a nossa Escola contemporânea não absorveu ainda totalmente a tecnologia do livro e não são poucas as Escolas que sequer possuem uma boa biblioteca.

Dessa forma a era do satélite, do micro, do vídeo-texto, do vídeo-disco, etc. encontra as nossas Instituições de Ensino em plena era pré-Gutenberg.

Que lástima, já que *educar* entre outras coisas não é mais somente a *previsão de necessidades sociais* num currículo adequado mas sim preparar os jovens para o imponderável e o imprevisível.

A revolução no que concerne aos papéis de aluno e de professor chegou.

Com a vinda do micro e a sua implantação necessária e obrigatória em todos os níveis de educação o *professor-informador* e o *aluno-ouvinte* terão que ser substituídos pelo professor-animador (o mais comunicativo possível) e o aluno-pesquisador, mutação que tem que ocorrer apesar de que isto, exigira algum investimento em recursos materiais.

O problema da *pesquisa* versus *ensino* será superada pela generalização da pesquisa.

Tudo na Escola do futuro será atividade de indagação e de desafio para a descoberta de soluções novas.

A velocidade da substituição do conhecimento eliminará a idéia de *ensino* fixo e desafiará a *pesquisa* em todos os níveis, mesmo das crianças que estão nos jardins de infância.

A Escola do Futuro que está chegando a nós, não terá currículos e programas fixos mas sim haverá a pesquisa.

Tudo que for programável será entregue aos computadores deixando à inteligência, a função de indagar e de questionar.

É um contraste, pois tudo o que é padronizado equivale a dizer é robotizado e iremos procurar em muitas coisas afastar-se da robotização.



Figura 6.1

Atualmente o professor dispõe de um auditório e toda a máquina escolar é feita para garantir uma plateia dócil ou submissa ao mesmo, sem as vezes poder indagar se o professor é um ator que mereça atenção.

Os alunos que não apreciam o espetáculo são considerados recalcitrantes.

Os professores têm, pois, um auditório-cativo por mais re-

pugnantes que sejam as suas "aulas" para os seus "ouvintes".

Talvez, se a escolha fosse livre, o professor não teria que usar *coação* (punições, provas, exames, certificados, etc.) para obter resultados.

Há quem entenda que enquanto o professor tiver o poder de "destruir" o aluno através da *reprovação*, o sistema escolar estará corrompido.

Não se trata aqui de suprimir a *avaliação* que vem com um diagnóstico-prognóstico, mas de retirar a força de destruição e atemorização, donde surge, a fraude comumente chamada de "cola".

Acreditamos que com o auxílio dos micros o professor atingirá uma posição mais dignificante qual seja a de um *perito* assessorando um grupo de trabalho e não estando mais no extremo do *carcereiro* e do *carrasco*.

Estamos ingressando em uma época em que um enorme arsenal de computadores comandados por satélites fornecerão àqueles que pedirem a consulta, qualquer informação que a humanidade tenha disponível.

Vivemos a época do instantâneo...

É óbvia então a substituição de bibliotecas tradicionais e de certas pessoas consideradas eruditas e que aliás em certos casos não passam de uma máquina ambulante de informação. É óbvio que jamais serão substituídos aqueles que criticam, interpretam, refletem e julgam. O micro não faz isto...

Com o auxílio dos micros as crianças mesmo as mais pequenas, sós ou em grupos pesquisarão por si mesmas soluções dos problemas, em alguns casos, apresentados pela primeira vez a elas.

É evidente que hoje por Escola se entende uma instituição

localizada em um edifício ou em um conjunto de edifícios.

A Escola poderá estar na casa de cada um usando por exemplo o vídeo-texto e o vídeo-disco.

É cristalino então que a *educação* do futuro será *contínua* e muito mais do que ter por meta a "*educação ajuda a ganhar a vida*" o "slogan" poderá ser a "*educação contínua faz com que se aprenda e se renove a própria vida*".

Isto entre outras coisas, nos poderá levar no fim do século XX a um dos fenômenos mais contundentes ou seja tirar da Universidade o poder de ser a guardiã da chama sagrada da sabedoria.

A Educação formal, que hoje no mundo está seccionada em três níveis (elementar, médio e superior) deverá ter outra estrutura.

Quem sabe à Escola do 1º e 2º grau caberá um papel de reflexão e integração e à atual Universidade caberá o papel crítico de projetar o futuro.

Deverá desaparecer o vestibular e isto não tem nada a ver com um sucesso de Erasmo Carlos, "Pega na mentira"...

Vestibular para o quê?

Será que o vestibular é a porta do paraíso ou do inferno?

Aquela famosa frase no dia da formatura "agora vocês vão entrar na vida" será totalmente abolida pois no lugar de se passar todos os dias da nossa vida inicial na Escola, passaremos desde o início, toda a nossa vida em contato com o mundo sem que nada dele nos separe.

A grande pergunta será, onde começará e terminará a Escola?

A resposta poderá estar no céu pontilhado de satélites, na densa rede eletrônica que se emaranha no ar, nas micro ondas cruzando o espaço nos telefones ligando todos os lares entre

si e dentro deles todo mundo tendo os seus micros, nos seus aparelhos de T.V., nas suas impressoras, nos seus discos magnéticos, bibliotecas de disquetes, etc.

Isto como se vê, impedirá a *clausura*. É verdade que aqueles que já leram Orwell duvidarão desta afirmativa.

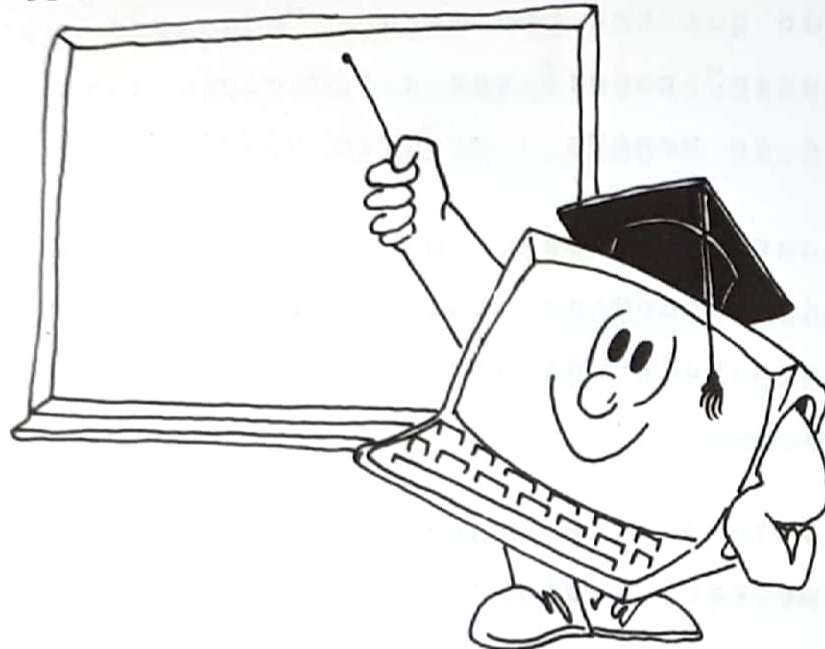


Figura 6.2 - "É o micro a chave para a Educação do Futuro? SIM, SIM, SIM,"

§6.2 - OS PROJETOS I.A.C. E I.D.C.

A I.A.C. (Instrução Auxiliada pelo Computador) refere-se ao uso dos computadores no ensino e não se envolve com o ensino sobre os computadores.

É assim, o uso dos computadores (macros, minis e micros) como um auxílio no ensino de qualquer tópico.

Existem quatro formas de sistemas I.A.C.:

- 1) Treinamento metódico e prática
- 2) Treinamento particular
- 3) Diálogo
- 4) Teste

1) Treinamento metódico e prático

Os programas de treinamento metódico e exercitação são feitos como complementação de um curso regular ensinado por um professor.

O professor introduz conceitos e apresenta as novas idéias na forma convencional.

O papel do computador é o de fornecer uma revisão regular e exercícios sobre os conceitos básicos.

Por exemplo no caso da matemática elementar cada estudante recebe diariamente um certo número de exercícios que são apresentados automaticamente, avaliados e é feita uma contagem ou seja uma pontuação das respostas certas sem a intervenção do professor.

Este tipo de treinamento individualizado é facilmente aplicável ao aprendizado de muitos tópicos de matemática elementar, ciências, linguas estrangeiras, etc.

2) Treinamento particular ou tutorial

No programa tutorial no lugar do enfoque de reforço do assunto já aprendido, apresenta-se ao aluno o material a ser aprendido diretamente iniciando-se assim o auto-aprendizado.

À medida que o estudante aprende, são feitas verificações e o computador age como um paciente professor particular fazendo-os voltar ao ponto onde cometeram enganos para o novo aprendizado dos conceitos.

Por sua vez os estudantes que já aprenderam convenientemente um conceito são também submetidos a um certo número de exercícios.

Passando por estes, recebem novos conceitos e novos exercícios.

Um tal sistema permite ao professor, que como se vê não foi eliminado, a gastar mais tempo com aqueles que tem problemas.

Um projeto deste tipo é o da Universidade de Illinois que o chamou de "Plato."

A "Control Data Corporation" está adaptando para microcomputadores o software Plato, inicialmente desenvolvido para computadores de grande porte.

Destaque-se que no momento o aluguel de um único terminal do Plato gira em torno de mil dolares, quantia equivalente ao preço de compra de um microcomputador Apple II.

3) Diálogo

Diálogo é uma forma mais sofisticada de instrução na qual o computador e o estudante são levados a manter uma conversação mútua.

A interação entre o estudante e o computador leva o primeiro a aprender ou a compreender o assunto discutido.

4) Testes

Os computadores são ideais como aplicadores de testes particularmente para combinar ou confrontar testes do tipo verdadeiro ou falso ou ainda testes de múltipla escolha.

O microcomputador apresenta a questão e o estudante fornece a resposta.

O micro verifica a resposta comparando a mesma com o gabarito de respostas corretas e indica a seguir a nota do aluno.

Uma vantagem significativa da I.A.C. é a sua flexibilidade.

Em uma sala de aula da I.A.C. podemos ao mesmo tempo ter alunos aprendendo francês, inglês, português, matemática,

química, etc.

Cada estudante progride na taxa de aprendizado que é conveniente para ele ou ela.

Instrução dirigida pelo computador - I.D.C.

Na instrução dirigida pelo computador (I.D.C.) no lugar de ensinar diretamente o estudante o microcomputador supervisiona a sua instrução e dirige o mesmo para as atuais experiências de ensino.

Assim, pode-se designar estudantes para lerem um certo livro, escutar uma certa fita, ver um certo filme e assim por diante.

Ao terminar esta atribuição os estudantes podem voltar ao terminal do computador para serem testados e receber novas tarefas.

A I.D.C. tem os seguintes objetivos.

- 1) Coletar e processar a informação do estudante
- 2) Obter a informação instrucional
- 3) Fornecimento dessa informação ao professor de forma resumida e de tal sorte que isto possa ser usado da melhor maneira para auxiliar a cada estudante.

Neste processo o micro é usado para guiar o estudante, por opção do próprio estudante, através de uma série planejada de experiências alternativas de aprendizado.

Por exemplo a professora de História quer ensinar algo sobre a guerra com o Paraguai.

A professora pode simplesmente dizer (e dirá, assim no futuro) "vã à biblioteca central computadorizada e entre com o código 1-3-7-1-3 para começar o estudo sobre a guerra com o Paraguai".

Provavelmente o micro informará antes ao estudante que ele deve escolher uma das alternativas:

- I) Escutar uma leitura gravada sobre a guerra do Paraguai.
- II) Olhar um conjunto de "slides" acompanhados de um texto.
- III) Estudar em um texto programado que venha com perguntas e respostas.

Os estudantes têm assim a opção de usar uma ou todas essas formas de aprendizado e ainda mais podem-se testar periodicamente.

O resultado do progresso de cada estudante é armazenado e enviado à professora.

Será impossível "colar"!!?

Bem, quase impossível, pois como é que fica se alguém responder por você?

Uma grande vantagem da I.D.C. é que os estudantes podem seguir, cada um no seu próprio passo.

Os estudantes, podem ver filmes, escutar gravações e fitas, ver trabalhos de arte, etc.

Além disso o micro pode resumir os resultados do teste em uma forma simples para dar ao professor ou professora um "feed-back" ou seja a correta "alimentação" para o julgamento do progresso de cada estudante e poder separar dessa forma os estudantes que precisam de um auxílio especial.

Existe uma estreita relação entre a I.A.C. e a I.D.C. visto que ambas utilizam o computador para auxiliar ou seja assistir o indivíduo que aprende.

Na I.A.C. usa-se o micro para apresentar a informação no terminal do estudante enquanto que na I.D.C. utiliza-se o terminal para gerenciar a instrução.

A I.D.C. baseia-se em especificar claramente os objetivos do comportamento ou seja o que se quer que o estudante realize, utilizando o micro como uma medida do desempenho de cada um.

A seguir prescreve uma relação de recursos instrucionais e o material conveniente para a realização desses objetivos e das necessidades do estudante.

Através dos testes, avaliações e prescrição de tarefas para estudantes os sistemas de I.D.C. são capazes de manipular ou seja levar avante muitos dos serviços rotineiros dos professores, os quais talvez desta forma, tenham mais tempo para trabalhar com particulares alunos atendendo dessa forma necessidades especiais.

Na década passada os sistemas de I.D.C. foram implementados apenas em sistemas de computadores muito grandes e portanto muito caros.

Hoje contudo, a I.D.C. está sendo implantada em sistemas com microcomputadores e estes são muito mais baratos.

O computador provocou um profundo impacto na nossa sociedade.

Uma parte desse impacto foi bastante negativo tal como a inquietação ou a importunação, e a perda da privacidade devido ao emprêgo inconveniente de dados pessoais e o desemprego das pessoas cujo serviço foi computadorizado.

Esses resultados prejudiciais da aplicação do computador devem ser ponderados contra as contribuições que o computador está trazendo para tornar a sociedade melhor e principalmente na EDUCAÇÃO.

O computador para o BEM ou para o MAL, e estamos aqui no mesmo dilema que nos é trazido por exemplo pela energia nuclear.

Não dá mais para atrasar o relógio...

As pessoas precisam posicionar a si mesmas de tal forma que elas controlem o computador.

Todos precisam entender a capacidade, o poder e as limitações desta máquina poderosa.

Todos precisam lembrar que é necessário fazer com que o computador seja sempre uma ferramenta, um servo obediente e que o mesmo estará sempre sob controle.

O FUTURO ESTÁ NAS MÃOS DAS PESSOAS E NÃO NA VONTADE DE UM COMPUTADOR. ELE NUNCA PODERÁ TER VONTADE...



Figura 6.3 - "Será que o micro vai ficar inoperante e vai continuar a tradicional situação escolar?".

§6.3 - O HISTÓRICO MUNDIAL DOS MICROCOMPUTADORES NAS ESCOLAS

Comumente, as pessoas imaginam os computadores nas Escolas como um recurso raro ao qual o estudante tem acesso ocasional.

O prof. Seymour Papert tem uma idéia totalmente diferente e é do seu livro "Mindstorm: Children, Computers and Powerful Ideas" que extraímos as seguintes considerações:

Precisamos pensar hoje em termos de um micro para cada criança e crianças tendo acesso ao micro desde pequenas.

Desta forma as idéias de hoje diferem da situação do futuro e quase tudo que se faz hoje, nesta área, será relevante no futuro apenas com um mal exemplo de como as pessoas se acostumam à modelos primitivos.

É natural começar a procurar "algo novo em educação", procurar exemplos de aprendizagem altamente bem sucedida.

Para o prof. Papert a imagem mais perfeita de aprendizagem bem sucedida é a forma como a criança aprende a falar.

Esta aprendizagem faz parte da vida e contrasta com a aprendizagem escolar de várias maneiras, sendo duas as que ele considera mais importantes.

- 1) Toda criança aprende a falar a linguagem usual do meio em que nasce.
- 2) Não deve existir técnica de escolarização, currículo, tempo para aprender, avaliação ou necessidade de profissionais especializados.

Para o prof. Papert o *micro* deve ser visto como o lápis ou uma caneta esferográfica.

O lápis é usado para rabiscar, desenhar, escrever, fazer so mas ou até como um objeto para morder e mascar.

É usado tanto para anotações ilícitas bem como para documentos oficiais.

O micro será tão onipresente, versátil e pessoal quanto o lápis na vida das crianças.

As crianças dominam o lápis e não são intimidadas por ele. Isto poderá ser igualmente verdadeiro em relação aos micros pessoais para as crianças.

O prof. Papert conta um pouco da sua experiência narrando o seguinte episódio com uma menina de quatro anos chamada Robin que no jardim de infância estava "brincando" com um programa de gráficos dinâmicos que lhe possibilitava fazer aparecer figuras na tela, movê-las, mudar de cor e juntá-las apertando uma ou outra das muitas teclas do micro.

O plano de instrução previa que quando Robin se cansasse de usar um programa, solicitaria um novo programa ao professor.

Porém Robin inteirou-se do processo e começou a teclar os caracteres de controle necessários para interromper um programa que não queria continuar e teclar o nome dos programas que ela queria utilizar, ainda que num ritmo de dois caracteres por minuto.

Essa independência de Robin em relação aos adultos enfatiza o fato de que com os micros, as crianças tornar-se-ão capazes de romper com muitos dos papéis impostos pela tecnologia primitiva dos costumes sociais.

O simples fato de Robin conseguir resultados, "teclando palavras", serve de apoio para fazê-la entender que a linguagem alfabética, tem, pela primeira vez em sua vida, um objetivo real e pessoal.

A linguagem falada e seus precursores apresenta-se nos primeiros anos de vida como um processo significativo e de interação com o mundo.

Aprender a falar empolga as crianças!

Porém para a maior parte delas, o ato de escrever serve apenas para obter a aprovação dos adultos.

Esta poderia ser a razão das crianças aprenderem a falar com mais facilidade do que aprendem a escrever.

Observando Robin, prof. Papert que é também professor do Massachusetts Institute of Technology convenceu-se que as crianças poderiam aprender a escrever tão facilmente quanto aprendem a falar, *se o ambiente no qual vivem lhes desse suporte para a linguagem escrita como lhes deu para a linguagem falada.*

Se Robin tivesse seu próprio micro, pudesse usá-lo sempre que tivesse vontade e se este micro lhe desse acesso a coisas excitantes não tem dúvidas o prof. Papert (e nos apoiamos totalmente essa conclusão) que ela dominaria o teclado em semanas, e daí, o alfabeto, a ortografia e a sintaxe necessários para colocá-la no domínio da linguagem escrita, que normalmente se adquire ao longo de alguns anos escolares!!!!

Da nossa experiência e daquela da amiga e excelente professora Josefa A. Alvarez podemos relatar o seguinte "capítulo" didático:

Nas férias de julho de 1983 promovemos um curso de iniciação à linguagem BASIC de 30h nas dependências da Livraria Nobel e inscreveram-se no horário da tarde na sua maioria crianças com idades variando entre 10 e 14 anos.

Com estas crianças inscreveram-se também três jovens que faziam Engenharia.

É óbvio que as aulas iniciais foram desinteressantes e monótonas para os três engenheirandos pois houve a necessidade de explicar lógica e operações matemáticas.

Porém qual não foi a nossa surpresa quando ao término do curso as crianças teclavam, aprendiam e modificavam os programas com a mesma versatilidade dos moços.

No aprendizado com o micro, tem-se a impressão que quanto mais jovem assimila-se mais rapidamente, porém todos devem e podem aprender.

Isto não depende da idade e sim da VONTADE.

A questão fundamental na Educação não é só melhorar as Escolas, mas sim entender porque elas são necessárias.

Porque determinados conhecimentos (como aprender a falar) são naturalmente assimilados da cultura, enquanto outros requerem, deliberadamente, instrução organizada.

Se as crianças aprendem a falar porque é uma atividade significativa na sua vida, é compreensível que não aprendam a escrever com a mesma facilidade quando a escrita não atende a uma necessidade real.

O mal causado por fazer as crianças aprenderem ritualisticamente é muito profundo, pois desenvolve os piores hábitos possíveis de aprendizagem e mina a auto-confiança individual como agente de independência intelectual e infantiliza a criança.

A passagem para objetos de aprendizagem fundamentalmente mais significativos traria consequências mais profundas do que a melhoria do domínio destes objetos.

Significaria que as crianças se tornariam aprendizes mais efetivas, com grande auto-respeito intelectual.

Se isso acontecesse, não apenas a natureza da aprendizagem da criança, mas também o papel da criança na sociedade poderia ser mudado.

Assim o efeito mais profundo do micro será o de desenvolver um novo respeito pela criança como agente intelectual independente.

porém a maioria das pessoas ligadas a Educação pensa ainda que as crianças devem ser consideradas como seres intelectualmente dependentes.

Assim para o prof. S. Papert e repito nos associamos a esta idéia, é a criança que vai programar o micro e adquirir o domínio sobre essa peça de tecnologia poderosa e moderna, estabelecendo o contato íntimo com conceitos da ciência, da matemática e da arte de construção do modelo intelectual.

Se não bastasse tudo isto vejamos algumas das previsões, ou seja, as tendências.

Nos Estados Unidos, a indústria dos computadores cresce a uma taxa de 23% ao ano.

Na Europa ela tem crescido a uma taxa de 20% e no Japão a uma taxa de 25%.

Espera-se que em 1990 o gasto com equipamentos de sistemas de computação nos Estados Unidos que em 1980 eram de aproximadamente 3% do produto bruto nacional passe em torno de 13%.

Nesta época 60% de todas as tarefas dependerão de alguma forma dos computadores e 20% de todos os trabalhadores precisarão de um conhecimento, do trabalho dos computadores para realizar as suas tarefas.

Bem, se existirem alguns milhões de possuidores de micros pessoais esses indivíduos constituirão um mercado pronto a gastar milhões em dólares, libras, yens ou cruzeiros em benefício das suas crianças.

Este fato encorajará pessoas criativas a introduzirem-se no campo da *inovação educacional* em números imprevisíveis.

Com isto aparecerá uma nova e importante classe de profissionais e de empresários e talvez mesmo de "estrelas" de

forma análoga ao que aconteceu ao longo do desenvolvimento do cinema como uma cultura.

Muitas coisas neste século tiveram grande influência sobre o comportamento humano, sobre os costumes, sobre a Educação, sobre a cultura.

Destacamos em primeiro lugar o *automóvel* que permitiu grande liberdade e privacidade ao indivíduo.

A seguir vem o *avião* que permitiu a verdadeira e rápida universalização.

O aparecimento do *cinema sonôro* fez com que a história do cinema passasse a ser a história desta cultura.

O cinema colorido, a televisão, as telecomunicações com o auxílio dos satélites tornaram real o conceito de instantâneo e imediato.

Não existem mais distâncias e quase toda informação pode ser recebida com a velocidade da luz.

E agora estamos no mundo dos microcomputadores!!!!

Na era do silício o futuro dos micros na EDUCAÇÃO não poderá ser dissociado da história das pessoas que farão a cultura dos microcomputadores.



Figura 6.4

§6.4 - FORMAS DE INTRODUÇÃO DO COMPUTADOR NA EDUCAÇÃO - EXPERIÊNCIAS EM DIVERSOS PAÍSES

Daysy Rosenthal e Jair dos Santos Lapa têm feito várias traduções e adaptações de artigos e posições atuais dos computadores na Educação.

Estes artigos tem sido divulgados no âmbito interno do Ministério de Educação e Cultura pela Secretaria da Informática com o que se cria a preocupação com o problema objetivando o debate ainda a nível interno sem obviamente aceitar os conceitos e as conclusões como posições aceitas ou assumidas pelo M.E.C.

Esperamos entretanto que estes "boletins" sirvam para que muito em breve seja firmada uma política nacional e quem sabe a criação de uma disciplina no 1º grau e no 2º grau diretamente ligada a Ciência da Computação.

Ladislav Cerych, a pedido do Ministério da Educação e Ciência da Holanda elaborou em 1982 um relatório muito interessante e é dele que extraímos os dados que vão a seguir.

Nos Estados Unidos, os gastos anuais com Informática nas instituições de Ensino Superior subiram de 500 milhões de dólares em 1971 para 1,3 bilhões em 1980.

Na Dinamarca o número de escolas do 1º grau com acesso a computadores dobrou de 1978 a 1981.

A França com o seu "plano dos 10000 micros" pretende colocar até 1984 cerca de oito micros em cada uma das 7200 escolas secundárias e para tanto o governo está também desenvolvendo um treinamento em massa de aproximadamente 10000 instrutores para ensinar a "alfabetização" em computador.

No estado de Minnesota, nos Estados Unidos a maior parte das Escolas do estado têm os seus micro Apple II ou Atari 800

ou terminais ligados a grandes computadores.

Desta forma estudantes e professores estão descobrindo que os computadores têm definitivamente lugar no nosso sistema educacional.

Na Faculdade de Engenharia da Fundação Armando Álvares Penteado, todos os alunos, a partir de 1983 tem uma disciplina com laboratório de micro em linguagem BASIC e posteriormente todo e qualquer trabalho de graduação pode ser realizado com o auxílio de terminais diretamente ligados ao IBM 4331.

Aliás a Instituição já há muito tempo vem desenvolvendo vários programas de auto-aprendizado via computador com a colaboração da IBM do Brasil.

É o que se chamou de projeto SCAI na FAAP.

A implantação do seu uso para as suas cinco Faculdades a saber Administração, Artes Plásticas, Comunicações, Economia e Engenharia estava sendo retardada pelo alto custo dos terminais. Hoje, como já comentamos no Capítulo 1, este problema está também contornado.

No momento, parece que apenas a França possui uma política central de Informática e Educação isto em vista da centralização do seu sistema educacional e à descentralização dos sistemas dos outros países como Alemanha, Estados Unidos, Inglaterra, etc.

Estima-se que em 1990 haja cerca de 25 milhões de micros pessoais nos Estados Unidos e que no final do século 80% das residências americanas irão possuir um micro.

Talvez o impacto da Informática na Educação formal não seja tão grande quanto o desejado por uns ou temido por outros, pois as Escolas e a Educação em geral são particularmente resistentes a introdução de novas tecnologias educacionais.

Porém, assim como ocorre com a televisão, a Informática influenciará a Educação mais pelo modo como os micros serão utilizados fora da sala de aula (nas residências) do que pela forma como os mesmos serão usados na Escola.

A questão, portanto, é saber se o sistema educacional irá, do mesmo modo como fez até agora com os televisores, desprezar a difusão dos computadores e todas as suas consequências isto é, se o sistema educacional irá permitir que o conhecimento e as atitudes das crianças sejam mais influenciadas pelo que aprendem fora da Escola (hoje ainda pelos televisores, amanhã pelos micros pessoais) do que pelo que aprendem na Escola propriamente dita.

Esperamos que não seja novamente este o rumo...

Os países que já experimentaram a introdução de computadores no sistema educacional seguiram diferentes tendências.

Alguns, como a França, preferiram introduzir o micro como ferramenta educacional de apoio às disciplinas regulares, inclusive às das áreas de ciências humanas.

Outros, como a Alemanha, optaram pelo ensino do computador como matéria isolada, tratada apenas nas Escolas técnicas ou vocacionais.

Finalmente houve casos de países que adotaram uma política mista como os E.U.A., a Dinamarca, a Inglaterra e o País de Gales, em virtude da dificuldade de uma separação nítida entre as duas abordagens.

A primeira abordagem permite que alunos de todos os níveis possam ter contato com o micro na sua vida escolar, preparando-se portanto para aceitar melhor este instrumento, na medida que prolifera seu uso em quase todos os setores da sociedade.

Entre os maiores inconvenientes desta forma de introdução está a necessidade de um grande número de professores especializados no uso de micros, o que é uma tarefa extremamente difícil.

Acrescente-se a isto, o custo elevado desta alternativa que alguns tentam solucionar com a utilização de micros mais simples e sem grandes recursos de linguagens, implicando numa diminuição da qualidade da educação fornecida.

Em certos casos adotam-se soluções paliativas como cursos rápidos de 1 a 3 dias para todos os professores ou cursos mais longos e completos para alguns professores que posteriormente apoiarão os demais.

Mesmo neste último caso, não se sabe ao certo quantos professores, de quais áreas e por quanto tempo devem ser treinados.

Várias tentativas de treinamento maciço de professores obtiveram resultados questionáveis talvez pela inadequação do próprio currículo desses cursos.

Deve ser salientado ainda, que os professores submetidos a um treinamento completo são atraídos para trabalhos mais rendosos na indústria e no comércio.

Assim o sucesso da Educação informatizada numa determinada Escola depende do esforço isolado de alguns professores entusiastas que dedicam seus momentos de lazer a um auto-treinamento.

Um outro problema encontrado é o da adaptação dos currículos das escolas para permitir a inclusão das disciplinas de Informática.

Estes currículos normalmente congestionados, não permitem a adição de novas disciplinas e o sistema escolar e os professores frequentemente reagem, à substituição das disciplinas

já existentes.

Deve-se também salientar a deficiência na produção de software de boa qualidade e com valor pedagógico.

Assim na França, uma experiência piloto indicou que de 400 programas educacionais desenvolvidos, apenas 20 foram considerados bons após um período de avaliação.

Na segunda abordagem, a Informática é tratada como disciplina à parte, nas escolas técnicas ou vocacionais, restringindo-se assim o universo escolar onde se introduz o micro.

Isto se traduz num custo mais aceitável para todo o sistema escolar.

Estas escolas serão responsáveis pela formação de profissionais especializados, cada vez mais imprescindíveis dada a proliferação de computadores em quase todos os setores industriais e a necessidade de se aumentar a competitividade das indústrias.

Entretanto, essa solução não coloca o computador ao acesso de pessoas não especializadas, o que dificulta a preparação de outras áreas da sociedade quanto à utilização deste recurso, que, seguramente será colocado à sua disposição no decorrer do tempo.

De qualquer forma já existem várias formas de treinamento da tecnologia do uso do computador e entre elas citamos:

- 1) Programa curricular da Universidade ou da Escola do 2º grau.
- 2) Treinamento em Escola do 1º grau (no Brasil não temos ainda).
- 3) Aprendizado em um curso particular.
- 4) Ensino no próprio serviço.

- 5) Por si próprio, que é aliás o que grande parte fará e esperamos que este livro seja uma útil ferramenta para este mister.



Figura 6.5

No seu livro "Microcomputer in the Schools" James L. Thomas dá informações muito interessantes sobre o que já se conseguiu nos Estados Unidos.

Aliás destaque-se o resumo do livro distribuído pela Secretaria de Informática do M.E.C. numa tradução e adaptação feita por Sandra Fongaro Casciano e Jair dos Santos Lapa.

O conceito da I.A.C. já existe há muitos anos e os primeiros foram iniciados em 1960 porém o tamanho e o custo dos computadores daquela época eram tão grandes que poucas instituições educacionais podiam introduzir a I.A.C.

O PLATO foi desenvolvido em 1970 e implementado em um grande sistema de computação com tempo compartilhado (time-sharing).

Terminais especiais com telas de vídeo sensíveis e com capacidade muito boa para a resolução gráfica foram usados para

se comunicar com o computador.

Até a presente data o PLATO é o projeto de I.A.C. de maior sucesso, porém ele é ainda muito caro.

Hoje graças aos baixos custos dos micros retornam-se os programas de I.A.C. com nova força.

É óbvio que paralelamente se espera que os fabricantes de software forneçam o necessário para a execução dos programas de I.A.C.



Figura 6.6 - O uso da I.A. e da I.D.C.

A "Apple Computer, Inc" desenvolveu o micro Apple II e o Lisa, que por sua versatilidade no uso de gráficos coloridos e por contar com grande quantidade de programas educativos, está sendo amplamente utilizado no setor educacional. Aliás aqui é falar novamente sobre o óbvio ou seja aqui no Brasil sobre o já comentado e "íntimo" MAXXI da Polymax.

A "Bell & Howell Company" está comercializando uma versão especial do Apple II para o qual foram desenvolvidos dois tipos de software, denominados Genis e Pass, que permitem ao leigo desenvolver o seu próprio programa de I.A.C.

A "Xerox Research Center" está desenvolvendo um computador do tamanho de um caderno e com linguagem apropriada chamada "Small talk", que permitirá, até mesmo as crianças, escreverem seus próprios programas.



Figura 6.7 - "É a criatividade saindo dos computadores".

Na área da Educação especial; a "Telesensory Systems, Inc." tem-se destacado na utilização de computadores para auxiliar no aprendizado de deficientes físicos.

Dentre os seus produtos, a empresa já está comercializando um programa educacional composto de oito micro jogos orientados para o aprendizado de deficientes visuais.

A "Utah State University" está testando, para ensino de crianças excepcionais, um sistema de micro com vídeo-disco que possui capacidade para a resposta interativa, além de um poder de motivação idêntico ao da televisão.

A "Texas Instruments, Inc." está trabalhando em computadores que dominam a síntese da voz, o que possibilita o desenvolvimento de jogos educativos como por exemplo o "fale e solete" e o "fale e leia" com extenso vocabulário.

Os exemplos acima demonstram o crescente interesse das escolas norte-americanas pelos computadores. Os educadores acreditam que os micros são a solução para os diversos problemas que ocorriam com o uso de macrocomputadores na Educação.

Parafraseando Isaac Asimov "Nós estamos atingindo um estágio no qual os problemas que precisamos resolver vão se tornar insolúveis sem os computadores. Eu não tenho medo dos computadores eu tenho medo da falta deles".

Vamos nos "alistar" no time de Isaac Asimov!!!



Figura 6.8 - "Será que assim teremos informações mais fidedígnas e desaparecerão as discriminações de ordem "pecuniária"?"

O prof. Papert chama a atenção para o fato de que os micros farão parte de cada um como hoje ocorre com a T.V., com o telefone, com o papel (que quem sabe vai desaparecer) independentemente de quaisquer decisões da comunidade educacional, pois a força impulsionadora encontra-se na indústria.

A simples presença do computador tem um potencial imenso de impacto psicológico, incluindo-se como já falamos a melho-

ria do processo de aprendizagem e afetando adultos, crianças bebês tanto nos lares como nas escolas...

Cada criança terá acesso ao poder da computação!!!

Em janeiro de 1982 na revista Phi Delta Kappa o prof. Harold G. Shane escreveu um artigo muito interessante sobre o *Microship e a Educação*.

Diz H.G. Shane que a comunicação humana ao longo do tempo, passou por quatro revoluções distintas: o desenvolvimento da *comunicação oral*; a *escrita*; a *palavra impressa* e finalmente a quarta revolução, potencialmente mais profunda e ainda em curso, que se iniciou com o rápido desenvolvimento das telecomunicações.

No final de 1981 especialistas de Nova York e Tóquio trocavam informações sobre o tratamento de glaucoma através de um simpósio, via satélite.

Nos últimos anos observamos uma proliferação dos robôs que vem se tornando cada vez mais, um elemento importante ou até indispensável à indústria.

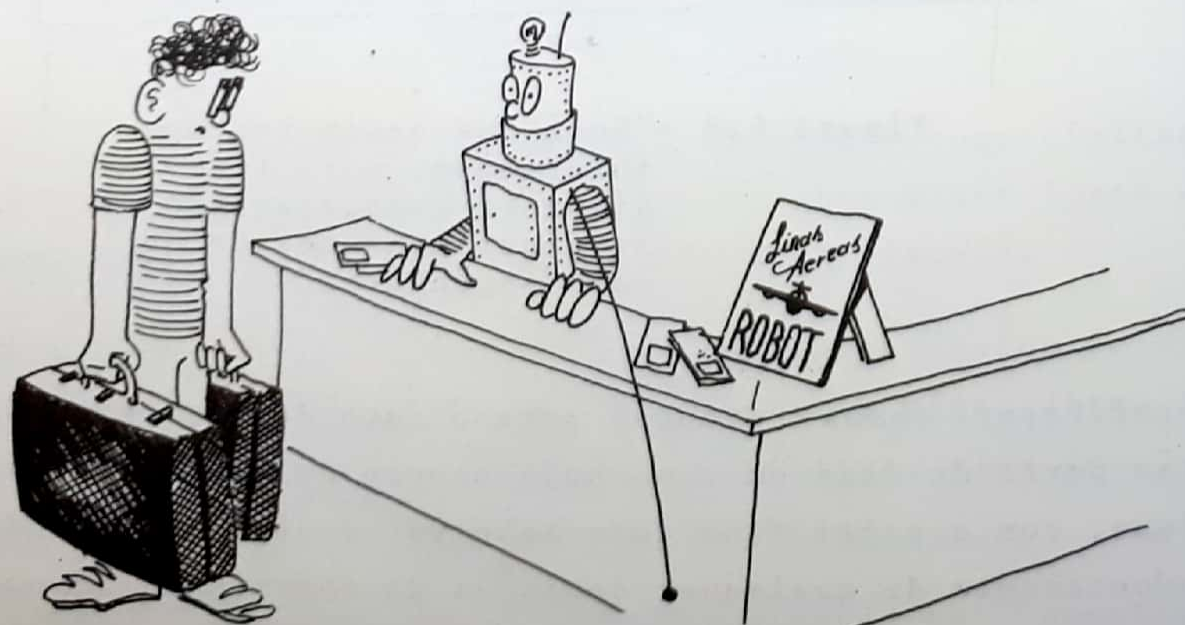


Figura 6.9

"ÁREA DE FUMANTES OU NÃO FUMANTES?
ASSENTO PERTO DA JANELA OU NA PASSAGEM?
QUE TIPO DE PERSONALIDADE O SENHOR GOSTARIA
DE TER A SEU LADO?"

A revolução das telecomunicações, provocada pelo *microchip de silício*, aliada às mudanças dela decorrentes, exige dos educadores uma nova abordagem sobre as formas de vida e interações que estão surgindo.

Em consequência, o trabalho nas escolas, a forma de pensar, a percepção do mundo, os conceitos de poder e as prioridades políticas precisam ser reanalisadas.

O computador está trazendo mudanças significativas e inovadoras tanto para o processo de ensino-aprendizagem como para a Educação em geral.

A computadorização da sociedade forçará as escolas a modificar a educação vocacional e os programas de orientação para o trabalho.

Cursos de reciclagem, reeducação vocacional e educação do lazer serão necessários para adultos que forem substituídos por robôs, abrindo novos caminhos de grande alcance social para a Escola.

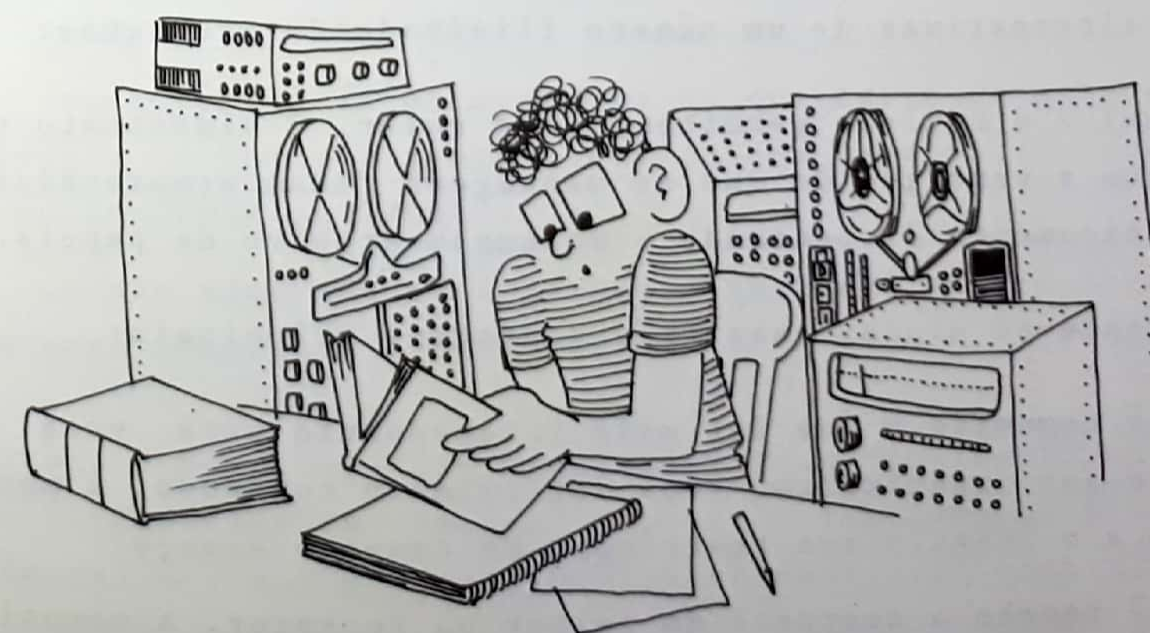


Figura 6.10 - "Todos, a partir de agora, terão a obrigação de conhecer os computadores e receber deles o auxílio para o aprendizado de qualquer tipo de assunto..."

Assim, é preciso que, professores, psicólogos educacionais e profissionais de comunicação de massa contribuam para modificar a proposta educacional, atuando de forma prospectiva em relação a esse desenvolvimento e às suas necessidades.

Em maio de 1983 com muita satisfação vimos a criação de um novo veículo de comunicação ou seja "O Univídeo" com um banco de dados educacional no novo veículo que é o vídeo-texto da Telesp.

A coordenação geral é da Faculdade Anhembi-Morumbi.

O que se espera é que os terminais do video texto estejam por todos os lados de São Paulo, nas casas, nos apartamentos, nas empresas, nas Escolas e nas ruas.

O rádio tem transmissão instantânea. O videotexto também, com a vantagem de que a informação não se perde no ar. Fica à disposição para se ver e ler quantas vezes quiser.

A televisão tem transmissão instantânea e imagem à cores. O videotexto também, com a vantagem de que o receptor passa a ter alternativas de um número ilimitado de programas.

O jornal e a revista permitem ler e reler. O videotexto também, com a vantagem de que as mensagens ficam armazenadas eletronicamente dispensando o volumoso arquivo de papéis.

Acrescente-se ainda, suas três vantagens principais:

- 1) Você comunica o que for mais interessante para você e para sua organização. Você determina o conteúdo, a mensagem e o anúncio sem restrições de tempo e espaço.
- 2) Você recebe a resposta do leitor ou receptor. A comunicação deixa de ser passiva e unilateral. Pode haver troca.
- 3) Você poderá elaborar e oferecer serviços mais especializados, como cursos por auto instrução, sistemas de compra e venda direta, controle de contas, etc.

O videotexto é um eficiente veículo de comunicação dirigida que possibilita principalmente o diálogo.

Esperamos que iniciativas como a da Faculdade Anhembi-Morumbi se multipliquem por todo o Brasil e principalmente que as tarifas telefônicas para finalidades educacionais possam gozar de preço especial para tornar esta forma de informação acessível a todos.

A minha editora, a Livraria Nobel como não poderia deixar de ser, através da visão prospectiva correta do futuro de Ary K. Benclowicz é uma das pioneiras no programa de videotexto apresentado pela Telesp.

Entretanto, as inovações tecnológicas e as novas soluções geram novos problemas como:

.necessidade de produzir especialistas e, ao mesmo tempo, salvaguardar os méritos da educação geral;

.manutenção do equilíbrio entre o volume de material de referência que teoricamente pode ser utilizado e a quantidade de realmente necessária;

.consciência da perspectiva de que cerca de metade dos empregos na indústria americana poderá desaparecer por volta do ano 2000 ou 2006;

.o uso adequado da informação que está se tornando cada vez mais abundante e acessível.

Talvez, o mais importante seja manter o equilíbrio entre o que é tecnicamente possível conseguir (através da microeletrônica) e o que é educacionalmente desejável para as situações de aprendizagem.

A longo prazo, talvez se descubra que a mutação sócio-educacional mais expressiva produzida pelos computadores em nosso ambiente eletrônico é o que podem fazer para melho-

rar a qualidade do ensino e os padrões de comportamento, já que seu valor não está no que podem ensinar, mas sim, na maneira como o fazem. A flexibilidade do computador moderno, pequeno ou grande, é infinita, seja qual for o tipo de trabalho ou finalidade, e, as tarefas que pode realizar são limitadas apenas pelo número de programas.

Para finalizar, evidenciamos a preocupação dos norte-americanos com a Educação e como solução a comissão que estudou o assunto apresentou o seguinte relatório para o presidente Ronald Reagan em junho de 1983 "Reforma do currículo que exigiria de todos os estudantes durante os quatro anos de "high school" - que mais ou menos equivale ao segundo ciclo do secundário no Brasil - tenham as cinco novas disciplinas básicas assim divididas: quatro anos de Inglês, três anos de Matemática, três anos de Ciências, três anos de Estudos Sociais e meio ano de Informática".

Sonhamos e "torcemos para que uma legislação semelhante surja no Brasil o mais breve possível introduzindo a disciplina que chamaríamos de Ciência da Computação obrigatória pelo menos durante um ano para todos os alunos do nosso atual 2º grau.

Vejamos quando é que a previsão tornar-se-á realidade?!?!?!"



Figura 6.11 - "E viva a Ciência da Computação!!!"

Bibliografia

Além dos livros que já indicamos no nosso livro Linguagem BASIC publicado pela Editora Nobel podem também ser úteis ao leitor ou leitora as boas obras dos seguintes "concorrentes" amigos:

- 1) Albrecht, B.
TRS-80 color Basic
John Wiley & Sons, Inc. - 1982
- 2) Alonso, J.R.F.
Simple BASIC Programs for Business Applications
Prentice-Hall, Inc. / Englewood Cliffs, New Jersey - 1981
- 3) Astier, J. Y.
La Conduite de L'Apple II
Le système graphique et l'assembleur de l'Apple II
Eyrolles - Paris - 1982
- 4) Astier, J. Y.
La Conduite de L'Apple II
Le BASIC de L'Apple II
Eyrolles - Saint Germain - Paris - 1982
- 5) Batista, L.-Katakura, G.M.
Elementos de Programação em Basic
Editora Edgard Blucher Ltda. - São Paulo - 1983
- 6) Bennett, W.S.-Evert, Jr.-C.F.
What every engineer should know about Microcomputers
Marcel Dekker, Inc. - New York - 1980
- 7) Birnbaum, M.-Sickman, J.
How to choose your small Business computer
Addison-Wesley Publishing Company - 1983
- 8) Blackwood, B.D.-Blackwood, G.H.
Intimate Instructions in Integer BASIC
Howard W. Sams and Co., Inc. - USA - 1981
- 9) Brown, J.R.-Finkel, L.-Albrecht R.L.
Basic for the Apple II
John Wiley & Sons, Inc. - 1982

- 10) Carris, W.
Inside ATARI BASIC
A fast, fun, and friendly approach
Reston Publishing Company, Inc. - 1983
- 11) Chance, D.
Computer Graphics with 29 ready-to-run programs
Tab Books Inc. - USA - 1981
- 12) Claudio, D.M.-Santos, J.A.R. dos
Microcomputadores e minicalculadoras
Seu uso em Ciências e Engenharia
Editora Edgard Blucher Ltda. - São Paulo - 1983
- 13) Coan, J.S.
Basic BASIC
Hayden Book Company Inc. - New Jersey - 1978
- 14) Commodore International Avalanche, Inc. - 1981
Personal Computing on the VIC 20
A friendly computer guide
- 15) Covvey, H.D.-Mc Alister, N.H.
Computer Choices: Beware of Conspicuous Computing
Addison-Wesley Publishing Company - 1982
- 16) Dwyer, T. - Critchfield M.
Basic and the Personal Computer
Addison-Wesley Publishing Company - 1983
- 17) Elgarten, G.H.-Posamentier, A.S.-Moresch, S.E.
Using Computers in Mathematics
Addison-Wesley Publishing Company - 1983
- 18) Ferguson, J.D.-Gordon, J.-Macari, L.-Williams, S.R.
Shaw, A.-Williams, P.
Programming the BBC Micro
Newnes Technical Books - England - 1983
- 19) Fox, A.-Fox, D.
Armchair BASIC
An Absolute Beginner's Guide to Programming in BASIC
OSBORNE/Mc Graw-Hill-Berkley California - 1983
- 20) Goldsmith Jr., W.B.
BASIC Programs for Home Financial Management
Prentice-Hall, Inc., - Englewood, New Jersey-1981
- 21) Haigh, R.W. - Radford, L.E.
BASIC for Microcomputers: Apple, TRS-80, PET
Prindle Weber and Schmidt - Boston - 1983

- 22) Haskell, R.
Apple Basic
Prentice-Hall, Inc. - 1982
- 23) Helms, H.L.
The Basic Book
Mc Graw-Hill Book Company - New York - 1983
- 24) Hergert, D.
Basic for Business
Sybex - 1982
- 25) Hickman, I.
Get More from your Personal Computer
Newnes Technical Books - England - 1982
- 26) Hirsch, S.C.
Basic Programming: Self-Taught
Reston Publishing Company, Inc. - 1980
- 27) Hurley, R.
More Real Applications for the ZX81 and ZX Spectrum
The Macmillan Press Ltd - London - 1982
- 28) Inman, D.
TRS-80 Color Computer Graphics
Reston Publishing Company, Inc.-1982-Virginia
- 29) James, M. - Gee, S.M. - Ewbank, K.
The Dragon 32 - Book of Games
Granada - London - 1983
- 30) Kamins, S. - Waite, M.
Apple Backpack - Humanized Programming in BASIC
BYTE Books - Peterborough - 1982
- 31) Kohl, H. - Kahn, T. - Lindsay, L. - Cleland, P.
ATARI Games and Recreations
Reston Publishing Company, Inc. - Virginia - 1982
- 32) Lima, Lauro de O.
Mutações em Educação segundo Mc Luhan
Editora Vozes Limitada - Petropolis, R.J. - 1976
- 33) Lien, D.A.
Learning TRS-80 Basic
Compusoft Publishing - 1982
- 34) Lint, W.van - Miller, A. - Belcher, B.
Introduction to Computer-Based accounting with
applications for Apple II - TRS-80 - Commodore Pet
Prentice-Hall of Australia - 1982

- 35) Lord, Jr. K.W.
Using the Radio Shack TRS-80 in your Home
Van Nostrand Reinhold Company - 1981
- 36) Manual de Programação BASIC - TK-85
Microdigital Eletrônica Ltda. - São Paulo - 1983
- 37) Manual dos Princípios Elementares do Micro Engenho
São Paulo - 1981.
- 38) Marshall, G.
Computer Languages and Their Uses
Granada - London - 1983
- 39) Mirshawka, V.
Linguagem BASIC
Livraria Nobel S.A. - São Paulo - 1983
- 40) Weill, A.S.
Liberdade sem medo - Summershill
EBRASA - São Paulo - 1968
- 41) Nevison, J.M.
Executive Computing
Addison-Wesley Publishing Company - Menlo Park-1981
- 42) North, A.
31 New ATARI Computer Programs For Home, School, Office
ARCsoft Publishers - Maryland - 1983
- 43) North, A.
101 ATARI Computer Programming Tips and Tricks
ARC soft Publishers - Maryland - 1983
- 44) Page, E.
TIMEX/Sinclair Computer Games Programs
ARC soft Publishers - Maryland - 1983
- 45) Parker, D. - Hann, M.
Computing is easy
Newnes Technical Books - 1982
- 46) Pereira, A.F.
BASIC para computadores pessoais
Livros Érica Editora Ltda. - São Paulo - 1983
- 47) Pereira, R.C.
Computes, Grilo! - Computador para todas as idades
Livros Técnicos e Científicas Editora-1983-R.J.
- 48) Pereira Filho, J.C.
Basic para micros pessoais
Editora Campus Ltda., R.J. - 1983

- 49) POLYMAX SISTEMAS E PERIFÉRICOS S.A.
MAXXI - Linguagem BASIC - São Paulo - 1983
- 50) POLYMAX SISTEMAS E PERIFÉRICOS S.A.
MAXXI - Operação e Linguagem (Nível BASIC)
São Paulo - 1983
- 51) Poole, L. - Borchers, M.
Some Common BASIC Programs
OSBORNE/Mc Graw-Hill - Berkley, California - 1979
- 52) Poole, L. - Cook, S. - Mc Niff, M. - Thomson, R.
Practical BASIC Programs - TRS-80 - Level II
OSBORNE/Mc Graw-Hill - Berkley - California - 1980
- 53) Poole, L. - Mc Niff, M. - Cook, S.
Your ATARI Computer - A Guide to ATARI 400/800
Computers
OSBORNE/Mc Graw-Hill - Berkeley - 1982
- 54) Presley, B.
A Guide to Programming in Level II Basic
Van Nostrand Reinhold Company - 1982
- 55) Price, W.T.
Elements of Basic - Plus Programming
Holt, Rinehart and Winston 1982
- 56) Silva, J.A. da
Criança também faz programas
Editora Campus Ltda. - Rio de Janeiro - 1983
- 57) Sinclair, I.
The Dragon 32 and how to make the most of it
Granada - London - 1983
- 58) Sinclair, I.
The ORIC-I
Granada - London - 1983
- 59) Spencer, D.D.
Computer Science Mathematics
Charles E. Merrill Publishing Company
A Bell and Howell Company - Ohio - 1976
- 60) Spencer, D.D.
Data Processing. An Introduction with Basic
Charles E. Merrill Publishing Company - 1982
- 61) Swanson, P.
Microcomputer Disk Techniques
BYTE/Mc Graw-Hill - 1982

- 62) Taylor, R.P.
 Programming Primer
 A Graphic Introduction to Computer Programming with
 BASIC and PASCAL
 Addison-Wesley Publishing Company - 1982
- 63) Vile Jr., R.C.
 Apple II - Programmer's Handbook
 Granada - London - 1982
- 64) Zaks, R.
 DON'T - Or How To Care For Your Computer
 Sybex - Berkley - 1981

Índice remissivo

A

Ábaco	9
ABS	179
Adição	63
AND	289, 291
ARRAY	153
ASCII	3, 218
ASC(P\$)	218
ATN	189

B

BASIC	43
Bit	2
BREAK	60
Byte	3

C

Cancela	214
CHIP	30
CHR\$	218
CLEAR	60, 223
CLOAD	95
CLS	102
Código de máquina	41
COLOR	304
Comando	60
Compilador	7, 41
Computador de 1ª geração	13
Computador de 2ª geração	13
Computadores de 3ª geração	14
Computadores de 4ª geração	14
Concatenação	222
CONT	60

COS(X)	188
CR	308
CSAVE	95
CTRL C	151, 311

D

DATA	141
Decrementar	100
DEF	182
DEF FN.....	183
DEFSTR	278
DIM	155
Divisão	63

E

EDIT	60
ELSE	137
END	60
ENTER	54
E/S entrada/saída	4
Espaço em branco	86
EXP(X)	182

F

FIX	171
Fluxograma	37
Formato de saída	106
FOR-NEXT	95
Função declaração alfanumérica	183

G

GOSUB-RETURN	122
GOTO	75
GR	304

H

Hardware	6
HCOLOR	321
HGR	320

-	321
HGR-2	307
HLIN	310
HOME	321
HPLOT	

I

I.A.C. - Instrução Auxiliada pelo Computador	332
I.D.C. - Instrução Dirigida pelo Computador	335
IF-THEN	75
Igual	64
Incrementar	99
INKEY\$	229
INPUT	82
INT	119
Instrução condicional	76
Instrução de atribuição	65
Instrução de desvio incondicional	77
Instrução não executável	59
Interpolador	7
Interpretador	41

L

LEFT\$	228
Leis da robótica	51
LEN	220
LET	65
Linguagem de alto nível	41
Linguagem de montagem	41
LLIST	60
LIST	60
LOAD	95
LogX	180
LOOP	98
LPRINT	57

M

Memória auxiliar	4
Memória de bolha	3

Microchip	14
Microprocessador	23
MID\$	227
Modalidade F	62
Modalidade G	62
Modalidade K	62
Modalidade L	62
Multiplicação	63
Multiprocessamento	48
Multiprogramação	48

N

NEW LINE	54
NOT	296
Notação científica padrão	69
Notação exponencial	69
Números aleatórios	129

O

ON-GOSUB	122
ON-GOTO	122
Operações aritméticas	63
Operadores lógicos	289
Operadores relacionais	76
OR	54, 290

P

Parênteses	71
PAUSE n	267
PDL	312
PEEK	285
PLOT	246
POKE	285
POINT	263
Ponto e vírgula	67
Potenciação	65
PRINT	54

PRINT @	251
PRINT AT	253
PRINT USING	203
Processamento em lotes	47
Programa	6
Programa armazenado	16
PROM	17

R

RAM	17
Ramificação condicional	78
RAND	132
RANDOM	132
RANDOMIZE	132
READ	142
REM	58
RESET	243
RESTORE	146
RETURN	123
RIGHT\$	228
RND	130
ROM	17
RUN	55
RUN CR	311

S

SAVE	95
SET	243
SGN	119
SHIFT	62
SIN(X)	188
Sistema binário	273
Sistema em tempo real	47
Software	6
SQR	172
STEP	98
STOP	151

STR\$(X)	226
STRING ARRAY	164
STRING	72, 149
STRING\$(X,Y)	219
Subrotina	122
Subtração	63

T

TAB	106
TAN(X)	188
Teclado	62
Tempo compartilhado	46
TEXT	304
TIME\$	232
TO	98

U

U.C.P. unidade central de processamento	1
UNPLOT	246

V

VAL(A\$)	224
Variável	64
Variável alfanumérica	149
Vírgula	67
VLIN	307

OUTRAS OBRAS DO AUTOR

Carderno de elementos de computação (Linguagem FORTRAN)
 Linguagem BASIC
 BASIC sem segredos
 TK-divertindo
 TK-lembrando
 TK-calculando
 Conhecendo e utilizando o TK 2000
 TK-2000 na matemática
 Dê um APPLE à sua vida
 Imprimindo maravilhas com a GRAFIX
 BASIC no TK 90X
 Jogos e desenhos no TK 90X
 Gráficos no TK 90X
 Conexão: GRAFIX-IBM PC e compatíveis
 Brincando com o TRS-Color

PRÓXIMOS LANÇAMENTOS

Gráficos no TK 2000
 O BASIC do DIGINET XT

1614



Impressão e Acabamento

GRÁFICA E EDITORA FCA

AV. HUMBERTO DE ALENCAR CASTELO BRANCO, 3972 - TEL.: 419-0200
SAO BERNARDO DO CAMPO - CEP 09700 - SP

O objetivo deste livro é introduzir o noviço à linguagem do microcomputador, conhecida como BASIC. Os micros, hoje, são usados por um grande número de pessoas e nas mais diversas atividades, podendo resolver problemas complexos e até ser instrumento de diversão.

O "BASIC SEM SEGREDOS" foi escrito especialmente para leitores com pouco ou nenhum conhecimento da ciência de computação, a fim de transmitir informações claras sobre o micro e sua utilização. Os principais tópicos deste livro são: aplicações e um pouco da história dos computadores; a linguagem BASIC elementar; o BASIC intermediário; o BASIC avançado; o microcomputador e a educação. Em um capítulo especial estão feitas as distinções entre a linha TRS-80 e a linha Apple.