00169
9 770101 304017

ANO XV - NO 169 - R$ 5,00

Micro Sistemas

Direct-X

INDO AONDE NENHUM OUTRO PACOTE DE PROGRAMA JAMAIS FOI

ATERRAMENTO E INTERFERÊNCIA VIA REDE

MENU PARA CLIPPER *POPDOWN*

COMO FAZER UM BOM DESIGN DE HOME PAGES PESSOAIS E CORPORATIVAS

ARQUIVOS SECRETOS.UM MÉTODO SIMPLES DA CRIPTOGRAFIA DE ARQUIVOS

Advogados ROTA Artistas ROTA Médicos ROTA Contadore
ROTA Advogados ROTA Artistas ROTA Médicos ROTA Contad
rofessores ROTA Esteticistas Engenheiros Arquitetos
ROTA Professores ROTA Esteticistas ROTA Engenheiros ROTA Arquit

Profissionais Liberais ANUNCIAR NA INTERNET É O MAIOR BARATO!!!

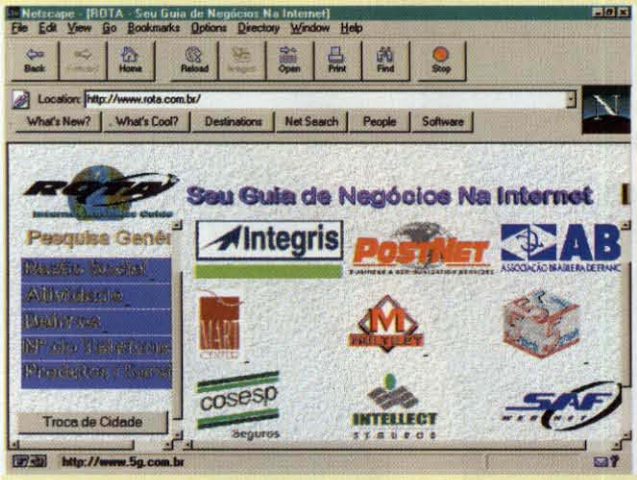
ROTA®

O Seu Guia de Negócios na Internet

Por apenas R$ 3,00* por mês Você expõe sua Empresa, Produto ou Serviço ao Mundo Inteiro Via Internet.

O **ROTA** é um Guia de Negócios Virtual, onde milhares de empresas, prestadores de serviços, comerciantes e profissionais liberais tem seu Espaço Publicitário garantido, a um **custo incrível!!!**

*custo para assinaturas anuais em uma única parcela



E tem mais!!!

Você poderá ter sua própria Home Page com Logotipo, Foto e Mensagem, por apenas

R$ 10,00 ao mês*

Você não precisa ter computador nem Internet!!!

Ligue agora e solicite uma visita de nosso representante

011 607.7124 • 011 605.9222

ROTA®

Rua Líbero Badaró, 101 - 7° andar
Centro - 01009-000 - S. Paulo - SP
Telefones: **607.7124/605.9222**

Todo assinante do ROTA receberá um Certificado garantindo a qualidade dos serviços

Acesse e Confira!!! ..... rota@rota.com.br • http:/www.rota.com.br

SUMÁRIO

ARTIGO

SEÇÕES

Diretor: Marcelo Zóchio
Editor: Leonardo Souza M. da Silva
Redação: Marcia Corrêa
Arte: Marcelo Zóchio e Julio M. F. Santos
Colaboradores: Renato Degiovane,Horacimar Pinheiro Cotrim,Marcello P. Gomes da Silva,Marcelo Elias Del Valle,Carlos Eduardo Ambrosi,Dálcio Crozero Momesso,Marcos M. Wada,Adriano C. R. da Cunha,Lígia Carvalho,Renato F. G. Carvalho .
Distribuição: Fernando Chinaglia Dist. Ltda.
Tel.: (021) 575-7725 / 575-7766
Assinaturas: 6 EDIÇÕES (R$ 30,00)

Venda de Assinaturas
São Paulo : Raul Fernandes
Tel/Fax: (011) 867-8377
Rio de Janeiro: (021) 261-7841

REVISTA MICRO SISTEMAS
Caixa Postal 18347
Rio de Janeiro - RJ - CEP 20722-970

ARTIGO

Audaciosamente Indo Onde Nenhum Outro Pacote de Programação Jamais foi

Por: Renato Degiovani

Nesta matéria vou mostrar como juntar três coisas aparentemente contraditórias: Delphi, Direct-X e Assembler 32 bits.

Não se pode negar, sob pena de cometer uma grande injustiça, que o Windows trouxe uma simplificação no ato de programar, mas só para quem cria programas comerciais do tipo controle de cadastro, contas a pagar e receber, fluxo de caixa, etc. Para quem produz sistemas direcionados ao fantástico mundo do entretenimento (jogos, para ser mais explícito) a coisa toda só piorou.

Mas, já dá para notar que alguns jogos, mesmo rodando sob Windows (e é claro, debaixo de um Pentium 100 no mínimo) apresentam um desempenho satisfatório. É o caso do recente X-Wing vs Tie Fighter da Lucas Arts. Esses novos jogos tem, em comum, uma coisinha chamada Direct-X (não confundir com Active-X, que é da mesma empresa e não tem relação alguma com o que estamos tratando aqui).

O tal do Direct-X é na verdade uma estrutura que visa disponibilizar aos programadores, de forma simples, racional e democrática, o acesso direto aos principais periféricos do micro, a saber: vídeo, som, joystick, etc. É claro que ele vai um pouco além do que isto, mas podemos ficar nesses três pontos por enquanto.

E o que significa acessar diretamente? Bem, para entender e principalmente assimilar o real alcance de tal implementação, precisaremos voltar um pouco no tempo.

Na década de 80, quando o DOS imperava absoluto como sistema operacional, o grande problema para os programadores era adequar seus produtos às inúmeras versões e tipos de periféricos. No início eram as placas e padrões de vídeo. Depois vieram as placas de som.

Cada novo produto lançado no mercado trazia embutido um novo padrão de acesso, novas tecnologias e novas complicações. Ainda hoje nos deparamos com jogos que, para serem instalados, precisam de ajustes em relação a placa de vídeo, de som e até em relação ao processador.

A proposta do Windows era tornar tudo isso transparente ao programador (esse marqueting foi usado para conse-

guir a adesão, ao Windows, das empresas produtoras de software, mundo afora). Os programas independentes - entenda-se: não abençoados pela Microsoft - fariam "chamadas" ao Windows quando quisessem acessar um periférico e então "ele" se encarregaria de acessar o periférico solicitado. É claro que a idéia era "cobrar" (e caro) pelo licenciamento de uso desta característica do sistema operacional. Perfeito e extremamente sedutor quando impresso num pedaço de papel ou na tela de um monitor, mas na prática...

Na prática isso funcionou, mas parcialmente. Quer dizer, para programas que não necessitam de desempenho e alta performance junto aos periféricos, a equação fechou perfeitamente. O Windows foi lançado e se tornou um sucesso na informática comercial. Mas, mesmo sendo a Microsoft a "dona" do mercado, o mercado não se comporta exatamente como os técnicos dela querem e a informática pessoal e de lazer passou a usar também o Windows.

Durante muito tempo, no entanto, bons jogos e Windows eram conceitos contraditórios e excludentes. Nem mesmo quando o mercado pode dispor de máquinas mais velozes esta situação se reverteu.

Para se ter uma idéia, num Pentium de 233 Mhz, quando usamos uma função interna da API do Windows para, por exemplo, colocar um pixel na tela, o computador se comporta como se fosse um velho XT de 4 Mhz. Uma mancada e tanto, se considerarmos o que representa o Windows em termos de máquinas instaladas e principalmente por se tratar de um "ambiente gráfico".

No tempo do Windows 3.1 a Microsoft tentou remendar a situação com uma pacote chamado WinG. Um remendo meio esquisito, mas que não foi adotado por quase nenhuma produtora de jogos. O Windows ainda deixava muito a desejar em termos de desempenho.

Máquinas mais potentes e novas tecnologias, além do lançamento do Windows 95, fizeram com que esse quadro se alterasse. A pressão do mercado acabou por impor uma solução mais ousada, mas como tudo aquilo que vem da Microsoft precisa ser complicado, estranhamente diferente e acima de tudo ocupar megas e megas de pura xaropada institucional...

Com o Direct-X não é diferente e veremos a razão disso. Iremos mais longe e chegaremos a uma solução bem mais interessante. Mas vamos voltar ao produto original, o Direct-X e seu pacote SDK de aproximadamente 90 megabytes compactados, que você pode downloadear gratuitamente do site da Microsoft. Não paga nada, mas prepare-se para passar umas 30 horas diante do micro.

http://www.microsoft.com/DirectX/Resources/downloads/dx5dl.htm

Basta ir lá e baixar o pacotaço. Pelo menos tiveram o bom senso de dividir o SDK em pacotes menores, que podem ser baixados ou não, dependendo do gosto do freguês. Mas espere só mais um pouco, antes de ir até lá e gastar essa baba toda em tempo de acesso. Provavelmente, se gostar do que vou mostrar mais à frente, você não precisará baixar nada. Talvez um ou dois megas, o que não custa nem 10 minutos, dependendo da hora e da velocidade de acesso.

Por enquanto vamos nos fixar nesta nova tecnologia (sim, eu baixei quase tudo que tinha lá e me arrependo até hoje).

Ao abrir o tal pacote (o SDK e mais um monte de outras coisas), vamos logo ao manual para nos informarmos acerca do que se trata. Reproduzo aqui os dois primeiros parágrafos, na íntegra e no idioma original unicamente para evitar a tentação de algum deslize durante a tradução do mesmo. São as palavras deles.

The Microsoft® DirectX Software Development Kit (SDK) provides a finely tuned set of application programming interfaces (APIs) that provide you with the resources you need to design high-performance, real-time applications. DirectX technology will help build the next generation of computer games and multimedia applications.

Microsoft developed DirectX because it wanted the performance of applications running in the Microsoft Windows® operating system to rival or exceed the performance of applications running in the MS-DOS® operating system or on game consoles. This SDK was developed to promote game development for Windows by providing you with a robust, standardized, and well-documented operating environment for which to write games.

Agora a má notícia: o SDK é basicamente voltado para quem programa em C++. Como essa linguagem é tão complicada quanto explicação de ministro da economia e, cá entre nós, só usada mesmo para castigar alunos de informática que se recusam a obedecer os professores, ficamos meio que órfãos de pai e mãe. Claro, estava bom demais para ser verdade.

Explico: não faz muito sentido nos dias atuais, aterrissar de cabeça em linguagens de programação como C e mesmo o Assembler, para escrever um programa do início ao fim e que irá rodar sob plataforma Windows. Mesmo em se tratando de jogos de alta performance.

A maior parte da programação dos jogos pode ser deixada à cargo dessas linguagens modernosas, como Visual Basic e (por que não) Delphi. Mas nem o VB nem o Delphi foram

criados para isso - fazer jogos. Então devemos (e podemos) juntar as duas coisas.

Seria pedir demais que a Microsoft juntasse o Direct-X com o Delphi. Se não fez isso até aqui nem com o VB, não faria com o seu maior concorrente neste segmento. Mas o que a M$ não faz, tem sempre alguém que resolve fazer e não é que, depois de vagar horas à fio pela mãe de todas as redes, acabei dando de cara com um objeto prontinho e que junta o Delphi ao Direct-X? Êita Internet da moléstia.

É o The Delphi Games Creator criado por John Pullen, Paul Bearne e Jeff Kurtz. O pacote pode ser baixado de graça no seguinte endereço:

http://www.ex.ac.uk/~PMBearne/download.htm

Roda em Delphi 2 e 3 e disponibiliza quase todas as funções do Direct-X diretamente via objetos do Delphi. Ainda está em fase de desenvolvimento, mas para o que queremos já dá para o gasto.

E o SDK lá na Microsoft? Esqueça-o. Não precisaremos dele. Na verdade, as funções do Direct-X já estão no seu computador. Isto, claro, se você alguma vez rodou um jogo que precisasse dele e o mesmo foi instalado. O SDK é apenas o pacote de programação. Se vamos usar o Delphi e se já temos a conexão com a biblioteca Direct-X, então não precisaremos do SDK e daquele desperdício de caracteres ASCII.

Mas, diriam os amigos leitores, acessar diretamente os periféricos usando o Pascal do Delphi não provocará perda de performance em nosso programa? Sim, eu responderia. De um modo geral a resposta é sim. Mas, nos casos em que for necessário alto desempenho usaremos Assembler. E mais, Assembler escrito diretamente dentro do Delphi e com as instruções de 32 bits. O acesso, por exemplo, à memória de vídeo é feito linearmente. Mais rápido impossível.

E depois, se temos que falar com o processador, por que usar intermediários como o C? Vamos logo ao único idioma que ele entende.

Juntando todos os pindurícalhos, podemos turbinar o nosso Delphi com coisas como page flipping, que é uma das mais usadas técnicas de animação - resumindo: podemos montar uma imagem em um buffer de tela, enquanto outro buffer mostra o resultado da montagem anterior.

Podemos usar bibliotecas de imagens, de sons, shapes e sprites (aquelas figurinhas com áreas transparentes); sons tridimensionais; stretching de imagens e mais uma montanha de recursos e possibilidades só antes sonhadas pelos programadores mais (digamos assim) delirantes.

Mas não se desiluda ao rodar os demos, tanto do SDK (se você for doido o suficiente para baixá-lo) ou do DGC. Não passam de irrisórias performances perto do que é possível fazer efetivamente.

Como e onde faremos? Bem, bem, bem... Nesta edição de Micro Sistemas mostrarei os passos iniciais de tal alquimia. Mas o leitor pode ficar tranquilo que, na TILT online - uma publicação que só existe na Internet - tem muito mais coisas do gênero. Lá poderá ser encontrado para download não apenas os exemplos mencionados aqui, como também outras implementações. Procure por uma seção chamada HARDMANIA (mania de fazer as coisas mais difíceis). O endereço da TILT online é:

http://www.tilt.net

E para aqueles que "perseguem" a Micro Sistemas desde os tempos do XT CGA, muito do que será visto aqui saiu direto de matérias especiais, como o projeto PRO KIT, Graphos III, etc.

Apertem os Cintos...

Se você é daqueles que tem medo de assombração, mula sem cabeça, quarto escuro, inflação alta, programação Assembler e outras coisas tão horripilantes quanto, então é melhor saltar logo esta página e nem ler o que vem depois do ponto final deste parágrafo.

Esta é uma matéria para programador macho, daquele que não tem medo de usar uma Int ou de esmerilhar um HD com seus melhores programas. Programador do tipo que escova bit com pano molhado, não sabe o que é Clipper nem dibeize e não chama cd-rom de cí-dí-rum.

Paulera total. Quer uma amostra grátis? Então lá vai: ligue o seu Delphi (2 ou 3) e que, nesta altura do campeonato já deverá estar com os objetos do DGC instalados

Crie num formulário novo o objeto chamado TDGCScreen; ajuste a tela para 800 x 600 e vamos ao trabalho. Esta será a nossa estrutura básica, de onde decolaremos para nossas aventuras no reino encantado da programação.

Preste atenção na procedure a seguir:

```
//Implementa a palette de
256 cores
procedure Impall;
begin
    asm
        mov   ecx,256
        mov   edx,3c8h
        mov   ebx,offset Pal0
    @@PL0:
        mov   al,0
        sub   al,cl
        out   dx,al
        inc   dx
        mov   al,[ebx]
        inc   ebx
        out   dx,al
        mov   al,[ebx]
        inc   ebx
        out   dx,al
        mov   al,[ebx]
        inc   ebx
        out   dx,al
        dec   dx
        loop  @@PL0
    end;
end;
```

Um doce se disser o que esta procedure faz (bem, está escrito lá em cima). Não entendeu nadica de nada? Vamos por partes: em primeiro lugar, iremos administrar as cores dos nossos programas acessando diretamente as portas que controlam a placa de vídeo (o pessoal da M$ vai ter dor de barriga se vir isso).

Aqui cabe um parêntese: isto é padrão SuperVGA e já foi testado em inúmeros modelos de placas. Mas pode ser que a sua seja uma daquelas que não obedecem a nenhum padrão, então...

Bem, não sei o que fazer nesses casos, mas se este for o seu, entre em contato conosco. Vamos pensar numa solução em conjunto. Na pior das hipóteses, você troca de placa, de cpu, de monitor e até de profissão.

As portas do paraíso estão nos endereços I/O 03C8h e 03C9h. O acesso aos registradores internos da placa é feito selecionando o registrador desejado na porta 03C8h e escrevendo o valor que desejamos colocar nele na porta 03C9h.

Não sabe o que é porta (não é aquele negócio que serve para atravessarmos as paredes)? Não sabe o que é I/O? Não sabe o que é endereço? Xiiiii.....

Ah, sabe né? Ufa! Ainda bem. Continuemos então. Fizemos um looping com 256 voltas para mandar para os registradores da placa de vídeo os valores RGB de cada uma das 256 cores indexadas, que espertamente manteremos numa matriz chamada "Pal0" (de Palette 0). Matriz global (não, não é da Rede Globo não) para todo mundo ter acesso à ela.

O resto da rotina é bico. Auto explicável, mas se tiver dúvida, não hesite em escrever para mim. Agora você deve estar se perguntando por que razão fizemos tudo isso, não é mesmo? Bem, eu diria que é por pura ironia. Os caras gastam caminhões de dinheiro para bolar um sistema cheio de mumunhas e aí um tapuia brasileiro vem e muda tudo - faz do jeito que quer. Na verdade acho que eles nem sequer sabem onde fica o Brasil, então podemos seguir nosso caminho com uma certa tranquilidade e mudar as coisas como bem entendermos.

Como estamos fazendo ao nosso gosto, vamos mudar outras coisas também. Em primeiro lugar, iremos trabalhar em 256 cores (e na resolução 800 x 600) e decidi reservar as primeiras 16 cores (índice 0 a 15) para usos do nosso sistema. Veja a seguir quais são essas cores. A razão delas? Gosto. E gosto não se discute (quando muito, lamenta-se).

Aqui está uma matrizinha especial, que contém os códigos RGB dessas cores:

```
RegCor: array [0..47] of
byte = (00,00,00,
                  00,00,33,
                  00,42,00,
                  00,36,36,
                  41,00,00,
                  00,17,25,
                  30,15,00,
                  41,41,41,
                  23,23,23,
                  00,00,63,
                  21,63,21,
                  00,63,63,
                  63,15,15,
                  59,59,56,
                  63,63,00,
                 63,63,63);
```

Ela foi definida na área de constantes globais, mas nem precisava. Faça uma rotina para passar esses valores para a matriz Pal0, antes de chamar a procedure que implementa as cores da palette. Um for next já dá pro gasto. Assim, sempre que quisermos "adequar" as cores de uma imagem ao nosso sistema, chamamos essa procedure de transferência. Adianto que este é o padrão adotado pelo programa Graphos III e que iremos montar numa outra ocasião (ih! falei demais).

Bom, já definimos nossas cores básicas e a instalação delas na tela que está sendo mostrada - o Direct-X trabalha com dois buffers de tela, mas isso é assunto para mais tarde. Por enquanto ficaremos no feijão com arroz.

A tela que veremos, ao rodar o programa, está na memória de vídeo da placa. Para acessá-la diretamente, só conhecendo a dita cuja profundamente. E isso não é garantia de que tudo irá funcionar corretamente.

No caso do Direct-X, existe uma "cópia" dessa tela na memória convencional e é nela que colocaremos nossos deditos. O truque para isso eu vou mostrar agora.

A tal cópia que mencionei fica numa área da memória endereçável, ou seja, nossa primeira providência é saber exatamente onde. Para isso criamos uma variável pointer global, como ilustrado abaixo:

```
var
    EndTel0: pointer;
    Pal0: array[0..767] of
byte;    //Palete da tela 0
```

Ih! Olha a matriz da nossa palette de 256 cores.

Agora, no evento on activate, do form1, colocamos os seguinte procedimentos:

```
{- Ativa o formulário pela primeira vez -}
procedure TForm1.TelaInitialize(Sender: TObject);
var
   Tmp: integer;
begin
    EndTel0:= Tela.Front.GetPointer;
    Tela.Front.ReleasePointer;
   Impal16;
end;
```

Até aqui pegamos apenas o endereço de um dos buffers de tela, justamente aquele chamado de Front. Rodando o que foi visto até aqui, você verá a tela ser inicializada e o Direct-X ser ativado. Vai dar preto total - tudo escuro. Não se desespere e pressione ESC que tudo volta ao normal e você é ejetado para fora do programa.

Mas, para testar a nossa palette, vamos colocar um looping de cls para as 16 cores que definimos. Pode ser colocado no evento on click, do Form1, assim ao clicarmos sobre a tela, ela auto-

maticamente mudará de cor. E por falar em cor, vamos definir mais duas variáveis (byte). Uma (Paper) para guardar o índice da cor de fundo que usaremos e outra (Ink) para guardar a cor da escrita.

```
//Limpa a tela com a cor do paper
procedure ClsTel;
begin
   asm
       mov   ebx,[EndTel0]
       mov   edx,600
       mov   ecx,0
       mov   al,[Paper]
    @@Lp0:
       mov   cx,800
    @@Lp1:
       mov   [ebx],al
       inc   ebx
       loop  @@Lp1
       add   ebx,1024 - 800 {224}
       dec   dx
       cmp   dx,0
       jnz   @@Lp0
   end;
end;
```

Fale a verdade, é moleza não é mesmo? Um laço dentro do outro - claro, primeiro "pintamos" cada pixel da linha e depois linha a linha repetimos o processo. Não acredito que você possa ter algum tipo de dúvida, mas se tiver, estarei aqui mesmo, no meu bat e-mail oficial:

degiovani@orlatec.com.br

Mas espere aí, onde você vai? Ainda não acabou não.

Um novo cursor para o mouse

Divertiu-se com nossa primeira incursão ao fantástico mundo do acesso direto e instantâneo, via Direct-X e Assembler 32 bits? Achou pouco, não é mesmo? Bem, então vamos ver mais um "brinquedinho".

Você deve ter percebido que, ao acionar o Direct-X o mouse some (e que fique sumido para sempre). Podemos criar o nosso próprio cursor, usando cores e os recursos que mais nos interessar. Proponho um cursor com o formato 20 x 20, usando as nossas 16 cores básicas e definindo o índice 255 para indicar transparência. Arrá, já sei, quer ver a matriz da seta cursor. Aqui vai ela (constante global):

```
Cursor: array [0..399] of
byte = (
  000,000,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,000,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,000,255,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,003,000,255,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,003,003,000,255,255,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,011,003,003,000,255,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,011,011,003,003,000,255,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,011,011,011,003,003,000,255,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,011,011,011,011,003,003,000,
  255,255,255,255,255,255,255,255,255,255,
  000,003,003,011,011,011,011,011,003,003,
  000,255,255,255,255,255,255,255,255,255,
  000,003,003,011,011,011,011,011,011,003,
  003,000,255,255,255,255,255,255,255,255,
  000,003,003,011,011,011,011,011,011,011,
  003,003,000,255,255,255,255,255,255,255,
  000,003,003,011,011,011,011,011,011,011,
  011,003,003,000,255,255,255,255,255,255,
  000,003,003,003,003,003,011,011,003,003,
  003,003,003,000,255,255,255,255,255,255,
  000,000,000,000,000,003,011,011,003,000,
  000,000,000,000,255,255,255,255,255,255,
  255,255,255,255,000,003,003,011,003,003,
  000,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,000,003,011,011,003,
  000,255,255,255,255,255,255,255,255,255,
  255,255,255,255,255,000,003,003,011,003,
  003,000,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,000,003,003,003,
  003,000,255,255,255,255,255,255,255,255,
  255,255,255,255,255,255,255,000,000,000,
  000,255,255,255,255,255,255,255,255,255);
```

Agora que o cursor do mouse não está mais sob o controle do sistema, precisamos tomar muito cuidado ao efetuar operações com a imagem, pois o cursor é parte integrante dela. Vamos começar com coisas bem simples, mas antes uma ligeira explicação.

Ao mover o cursor pela tela, provocamos duas situações distintas: colocar o desenho do cursor nas coordenadas correspondentes e salvar o que originalmente estava naquele local. São duas operações distintas mas que faremos numa mesma procedure - enquanto vai plotando a figura do cursor, nossa rotina irá salvando o conteúdo original da posição. Salvar onde? Num buffer global (var), é claro.

```
Var
    PmLin,PmCol: word;                //Coordenadas do cursor do
mouse
    EndCur: dword;                     //Endereço da área
     BufCur: array[0..399] of byte; //Área da tela sob o
cursor
```

As coordenadas X,Y do cursor deverão estar em duas variáveis globais, que são atualizadas pelo evento OnMouseMove, do Form1.

```
begin
  PmLin:=Y; PmCol:= X;
  PoeCursor;
end;
```

Primeiro calculamos o endereço real dessas coordenadas, lembrando que uma linha de pixels, no Direct-X, independente da resolução escolhida tem exatos 1024 bytes no modo 256 cores. Depois salvamos esse endereço para uso futuro (devolver o conteúdo da área sob o cursor) numa outra variável global (EndCur). O resto está comentado na rotina a seguir.

```
procedure  PoeCursor;
begin
   asm
       mov   eax,1024
       mov   ebx,0
       mov   bx,[PmLin]
       mul   ebx              //1024 x qt de linhas
       mov   bx,[PmCol]
       add   eax,ebx            //mais as colunas
        add   eax,[EndTel0]   //mais o endereço inicial
       mov   [EndCur],eax     //da tela
       mov   ebx,eax
        mov   esi,offset BufCur
        mov   edi,offset Cursor
       mov   ah,20             //20 linhas
    @@Lp0:
       mov   ecx,20            //20 colunas
    @@Lp1:
```

```
      mov   al,[ebx]     //Pega um pixel original
      mov   [esi],al     //Salva no buffer
      mov   al,[edi]     //Pega o pixel do cursor
      cmp   al,255        //Salta se for tansparente (= 255)
      jz    @@Lp2
      mov   [ebx],al     //Coloca na tela
  @@Lp2:
      inc   ebx          //Próximo pixel
      inc   edi
      inc   esi
      loop  @@Lp1        //20 vezes (colunas)
      add   ebx,1024 - 20    //Salta o restante da linha
      dec   ah
      jnz   @@Lp0        //20 vezes (linhas)
```

Se executar o programa agora vai dar Tilt (he, he he...). Quer dizer, vai encher a tela de setas. Precisamos, ao ocorrer um movimento do mouse, devolver o conteúdo original da área antes de desenhar novamente a seta. A rotina é bem simples:

```
procedure  RepoeArea;
begin
    asm
        mov   ebx,[EndCur]
//Pega  o endereço do local
        mov   edx,offset
BufCur
        mov   ah,20
    @@Lp0:
        mov   ecx,20
    @@Lp1:
        mov   al,[edx]
        mov   [ebx],al
        inc   ebx
        inc   edx
        loop  @@Lp1
        add   ebx,1024 - 20
        dec   ah
        jnz   @@Lp0
    end;
end;
```

O evento OnMouseMove, do Form1, passa a ser então o seguinte:

```
begin
    RepoeArea;
    PmLin:= Y;  PmCol:= X;
    PoeCursor;
end;
```

Mas para não dar Tilt novamente, temos que inicializar o cursor tão logo o programa entre em execução. Vamos mudar o evento OnInitialize do componente Tela:

```
begin
    EndTel0:=
Tela.Front.GetPointer;
    Tela.Front.ReleasePointer;
    PmLin:= 0;  PmCol:= 0;
    PoeCursor;  Impal16;
end;
```

Se rodar agora o cursor vai passear pela tela em toda sua beleza e esplendor. Mas, se clicar o mouse... Mais Tilt.

A razão é simples: temos um evento OnClick, pendurado no Form1. Quando clicamos o botão do mouse ocorre um CLS. Tudo foi apagado, inclusive o cursor do mouse, mas o antigo conteúdo original ainda está naquele buffer especial. Quando o evento OnMouseMove vai plotar a nova seta, antes repõe o conteúdo do buffer. É aí que mora o perigo.

Vamos então escrever uma procedure esperta, apenas para atualizar o tal buffer de área. Aqui está ela:

```
procedure  GetArea;
begin
    asm
        mov   edx,[EndCur]
        mov   ebx,offset
BufCur
        mov   ah,20
    @@Lp0:
        mov   ecx,20
    @@Lp1:
        mov   al,[edx]
        mov   [ebx],al
        inc   ebx
        inc   edx
        loop  @@Lp1
        add   edx,1024 - 20
        dec   ah
        jnz   @@Lp0
    end;
end;
```

Moleza. Agora temos que macetear a nossa procedure de CLS. Veja como ela ficará:

```
//Limpa a tela com a cor do paper
procedure  ClsTel;
begin
    asm
        mov   ebx,[EndTel0]
        mov   edx,600
        mov   ecx,0
        mov   al,[Paper]
    @@Lp0:
        mov   cx,800
    @@Lp1:
        mov   [ebx],al
        inc   ebx
        loop  @@Lp1
        add   ebx,1024 - 800, //224
        dec   dx
        cmp   dx,0
        jnz   @@Lp0
    end;
    GetArea;
end;
```

Viu só como é simples?

O Nosso Alfabeto Particular

Agora que dispomos de um cursor para passear pela tela, que tal escrever alguma coisa nela. Já sei, você vai dizer que isso é fácil pois podemos usar funções como TextOut, fontes True Type, etc. Não vou dizer que você está errado e se podemos usar isso em nossos programas, muito que bem. Mas, onde fica nosso "aprendizado" de como as coisas realmente funcionam? Vamos lá então.

Em primeiro lugar, uma letra nada mais é do que um desenho que deve ser plotado da mesma forma que um tipo qualquer de desenho: pixel a pixel. E para saber como desenhar, precisamos de uma matriz, ou seja, um desenho prévio de como é a letra.

Estamos implementando agora um alfabeto que foi projetado especificamente para esta matéria. Apenas como ilustração, apresento um trecho da matriz das letras que usaremos em nossas produções (não esqueça: constante global):

```
Letras: array [0..1595] of byte = (
     000,000,000,000,000,000,000,000,000,000,000,000,  //Char 32
     000,000,048,048,048,048,048,048,000,048,000,000,  //Char 33
     000,000,108,108,108,000,000,000,000,000,000,000,  //Char 34
     000,000,036,036,126,036,036,126,036,036,000,000,  //Char 35
    ...
     000,000,000,000,000,000,000,000,000,000,000,000,  //Char 163
     000,000,000,000,000,000,000,000,000,000,000,000,  //Char 164
     000,000,000,000,000,000,000,000,000,000,000,000);//Char 165
```

Vamos às considerações. Em primeiro lugar, sabemos que as letras do alfabeto ocupam geralmente do caracter espaço (código 32) até o caracter 127. Acima disso é considerado como ASCII estendido e é exatamente onde estão as letras acentuadas (claro, não foi um brasileiro que inventou o computador, portanto...).

Comos as letras acentuadas estão espalhadas no final do conjunto de códigos, vamos reagrupá-las em seqüência, definindo como seus códigos os valores entre 128 e 154. Daí até o código 165 deixaremos "vago" para usar como caracteres redefiníveis. Assim, se precisarmos de um tipo de letra especial, ao invés de trabalharmos com um arquivo de figuras, definimos a dita cuja como um caracter.

Os códigos de acentuação passam a ser então os seguintes:

128 = Ç	135 = Í	142 = Õ	149 = ú
129 = ç	136 = Ó	143 = ù	150 = â
130 = vago	137 = Ú	144 = à	151 = ê
131 = Ü	138 = Â	145 = á	152 = ô
132 = À	139 = Ê	146 = é	153 = ã
133 = Á	140 = Ô	147 = í	154 = õ
134 = È	141 = Ã	148 = ó	155 = vago

Este sistema irá exigir então que, ao solicitarmos a impressão de um caracter acentuado pelo padrão Windows (para não alterarmos a digitação também), ele seja convertido para os valores da nossa nova tabela.

Mais uma matriz na área de constantes globais e uma rotina bem simples:

```
Cods: array[0..26] of char = (
  'Ç','ç','|','Ü','À','Á','È','Í','Ó','Ú','Â','Ê','Ô',
  'Ã','Õ','ü','à','á','è','í','ó','ú','â','ê','ô','ã',
   'õ');
```

```
procedure LetraAcent;
begin
  asm
    mov esi,OFFSET Cods //Endereço da tabela
    mov ecx,27          //27 caracteres
    mov ah,128          //Começando pelo 128
  @@Letra0:
     cmp al,[esi]       //É esse?
     jz  @@Letra1       //Sim
     inc esi            //Não, testa o próximo
     inc ah
     loop @@Letra0
     mov ah,al
  @@Letra1:
     mov al,ah          //Muda o código
  end;
end;
```

Nossa procedure é chamada de dentro de um bloco Assembler, com o registrador AL contendo o valor do código a ser testado (se ele for maior que 128, senão não precisa testar).

A procedure que imprime a letra está logo a seguir (na íntegra):

Considerações: em primeiro lugar, vamos definir a questão das cores. A letra é sempre impressa com a cor presente na variável Ink. Mas o fundo dela depende de uma flag, chamada FlgFun. Se ela for 0, então o fundo é deixado intacto, ou seja, a letra é transparente; se for diferente de 0, então a cor definida na variável Paper será usada para pintar o fundo da letra.

Note que a nossa procedure obteve o endereço de impressão numa variável (global) chamada EndVid. Esta variável contém o endereço (32 bits) do canto esquerdo superior, na tela, onde a letra será impressa.

Para efeito de localização, dividimos nossa tela em células de 8 colunas por 16 linhas de pixels. Cada célula dessas será a posição de impressão de um caracter. Em nossa resolução de 800 x 600 temos 100 colunas por 37 linhas de impressão

```
procedure ImpChrs;
begin
   asm
       cmp  al,128             //É letra especial?
       jc   @@GRAP4            //Não
        call LetraAcent         //Sim, testa letra acentuada
    @@GRAP4:
       push ax
        call Posic              //Calcula posição de impressão
       xor  eax,eax            //eax = 0
       pop  ax
        mov  esi,OFFSET Letras
        mov  ecx,12            //Altura das letras
        sub  al,32              //Desconta os primeiros 32 chrs
       mov  ah,0
       mul  cl
        add  esi,eax            //Endereço na matriz das letras
        mov  edi,[EndVid]      //Endereço na tela
    @@GRAP0:
       push edi
       push ecx
        mov  ah,[esi]           //Pega uma matriz (8 bits)
       inc  esi
       mov  ecx,8
    @@GRAP1:
        rcl  ah,1               //Rotaciona para a esquerda
        mov  al,[Ink]          //Imprime cor Ink se bit for 1
        jc   @@GRAP2
        mov  al,[Paper]        //Imprime cor Paper se bit for 0
        cmp  [Flgfun],0        //mas apenas se Flgfun for <> 0
        jz   @@GRAP3
    @@GRAP2:
        mov  [edi],al          //Plota o pixel
    @@GRAP3:
        inc  edi                //Próximo pixel
        loop @@GRAP1           //Oito vezes
       pop  ecx
       pop  edi
        add  edi,1024           //Salta para próxima linha
        loop @@GRAP0           //Doze vezes
        call Passo              //Avança uma posição de impressão
   end;
end;
```

de texto. No entanto, o alfabeto é definido com uma área útil de impressão de 8 colunas por 12 linhas de pixels.

Um conjunto de variáveis irá guardar as coordenadas atuais de impressão e os parâmetros de definição da nossa tela, ou área de texto. São elas (todas globais):

```
Coluna: byte = 0;       //Coordenadas de texto
 Linha: byte = 0;
 Ultcol: byte = 100;     //Última col+1 impressa
 Ultlin: byte = 37;      //Última lin+1 impressa
 Mrgcol: byte = 0;       //Margem de coluna
 Mrglin: byte = 0;       //Margem de linha
 FlgFun: byte = 0;      //Flag de impressão do fundo
 Posx: word = 0;         //Coordenada de pixel (X)
 Posy: word = 0;         //Coordenada de pixel (Y)
 Deslet: byte = 0;       //Deslocamento para baixo
 EndVid: dword = 0;      //Endereço de plotagem
```

Para definir um local de impressão, basta colocar as coordenadas desejadas nas variáveis Linha e Coluna. Agora precisamos de uma procedure que calcule, a partir desses valores, o local exato onde a impressão acontecerá. Aqui está ela:

```
procedure Posic;
begin
   asm
       mov   al,[Linha]        //Linha de texto (100 x 37)
       add   al,[Mrglin]       //Compensa a margem
      mov   dl,16              //Altura das letras
      mul   dl
       mov   [Posy],ax         //Coordenada Y de plotagem
       mov   al,[Coluna]       //Coluna de texto
       add   al,[Mrgcol]       //Compensa a margem
      mov   dl,8               //Largura das letras
      mul   dl
       mov   [Posx],ax         //Coordenda X de plotagem
       xor   ebx,ebx           //Zera o registrador ebx
       mov   eax,1024          //Tamanho da linha de pixels
       mov   bx,[Posy]
       mul   ebx               //Calcula endereço
        mov   edi,[EndTela0]   //EDI início da área de tela
       add   edi,eax
       mov   bx,[Posx]
       add   edi,ebx
        add   edi,[Deslet]      //Acrescenta deslocamento
       add   edi,2048
        mov   [Endvid],edi
   end;
end;
```

É para finalizar, precisamos de uma procedure para atualizar o deslocamento da impressão, sempre que um caracter é impresso.

```
procedure Passo;
begin
    asm
        mov   bl,[Coluna]       //bl = coluna atual
        mov   bh,[Linha]
       inc   bl
        cmp   bl,[Ultcol]       //Chegou no canto direito?
       jc    @@PASS0            //Ainda não
      mov   bl,0                //Primeira posição à esquerda
       inc   bh                  //Próxima linha
        cmp   bh,[Ultlin]       //Chegou na última linha?
       jc    @@PASS0            //Ainda não
      dec   bh                   //Volta uma linha
       mov   bl,[Ultcol]       //Última possição de impressão
      dec   bl
    @@PASS0:
        mov   [Linha],bh        //Redefine as variáveis
        mov   [Coluna],bl
   end;
end;
```

Só falta testar o conjunto de impressão de caracteres. Pelo exemplo a seguir, colocado no evento OnClick, do Form1, deveremos ter como resultado a tela toda tomada por um bando de letras "A". E não esqueça o principal, apagar antes o cursor do mouse e recolocá-lo na posição logo após a impressão.

```
procedure TForm1.FormClick(Sender: TObject);
begin
    Paper:= 4; Ink:= 14;
    RepoeArea;
    asm
       mov   [Linha],0
       mov   [Coluna],0
       mov   ecx,5000
    @@Lp:
       push ecx
       mov   al,'A'
       call ImpChrs
      pop   ecx
       loop @@Lp
    end;
    GetArea;
end;
```

Imprimindo mensagens

Já dispomos de um alfabeto e portanto só falta mesmo definir a estrutura de impressão de strings. Antes porém as considerações...

Na programação em linguagem Assembler estamos sempre diante de situações simples, como por exemplo definir frases que possam ser impressas (men-

sagens, avisos, controles, títulos, etc). Desenvolvi uma prática bastante heterodoxa e que causa arrepio em 100 por cento dos acadêmicos da programação.

Se você é aluno de faculdade, por favor e para seu próprio benefício, não mostre ou comente com seus professores o que verá aqui. Eles certamente o olharão com desconfiança e temerão por sua sanidade mental, mas é puro preconceito deles.

Tudo isso se baseia no fato de que eu misturo data com código, um crime inafiançável na academia. Mas o computador é meu, o programa é meu, o compilador também é meu e portanto não tenho que dar satisfações à ninguém sobre como misturo as coisas. Tem dado certo até hoje e ponto final.

Vamos ao abacaxi. Por exemplo, num dado momento precisamos escrever uma frase qualquer na tela. Vou usar, dentro de um bloco ASM, o registrador EBX para apontar para o início da frase:

```
asm
      mov   ebx,OFFSET @@Msg1
      call  Tprint
      jmp   @@Lp0
  @@Msg1:  db  'Alo',252
  @@Lp0:
     nop
```

@@Msg1 é o início da frase e Tprint a procedure que irá imprimí-la. Como ela está logo abaixo da chamada CALL, precisei criar um salto sobre a mesma, para que o processador não executasse as letras como se fossem instruções Assembler.

Decididamente não está nada elegante, mas isso só é usado quando queremos separar um local para definir um monte de variáveis e de frases que são usadas por diversos pedaços da nossa rotina principal. Essa definição poderia ficar logo no início da procedure:

```
asm
      jmp   @@Inicio
   @@Msg1:  db  'Alo',252
   ...
   ...
   @@Inicio:
      mov   ebx,OFFSET @@Msg1
      call  Tprint
      ...
```

Note que a primeira instrução faz com que o processador "pule" por sobre as variáveis da nossa rotina. No final, isso não é nada diferente do que já fazemos nas procedures, separando constantes, variáveis, labels, etc. Não justifica nenhum ataque histérico, por parte dos acadêmicos.

Mas o que vem a seguir... Existem momentos nos quais precisamos imprimir, sem mais delongas, uma pequena frase. Algo bem simples e que provavelmente não se repetirá mais por todo o nosso programa. Aí é que o Chupa Cabra ataca:

```
asm
   call Disp
      db 12,10,10,'Testando... 1,2,3!',252
      ...
end;
```

Cadê o salto sobre a frase? Cadê a lógica disso tudo? Cadê início da frase? Cadê o Wally?

Arrá, não falei que era piração pura? Mas calma que tem explicação plausível. Vamos por partes, como diria Jack, o estripador.

Notou o valor byte 252 no final de ambos os procedimentos? Então, esse valor indica para a nossa rotina de impressão que alí é o fim da frase. Não usei os tradicionais 13,10 (carriage return). A razão? Sei lá, gosto mais do 252 (não é nada disso, mas um dia eu conto em detalhes porque 252).

A rotina (procedure) que imprime chama-se Disp e será mostrada daqui a pouco. Ela funciona em conjunto com a Tprint. Na verdade, são ambas uma só rotina, mudando apenas como é feita a chamada a cada uma delas.

Os códigos 12, 10 e 10 (existem outros que veremos também) são ações simplificadas para procedimentos prá lá de manjados. Por exemplo, o valor 12 indica que os dois bytes a seguir deverão ser usados para determinar a linha e a coluna onde a mensagem será impressa.

Por exemplo: 9, 10, 65 indica que (9) serão impressas 10 letras "A"; 5, 7 indica que a variável ink (5) será definida com a cor cujo índice é 7; 6, 10 é o mesmo, só que para a cor paper; e o valor 7 chaveia (on/off) a impressão do fundo da letra (veja na parte anterior mais sobre a impressão da cor de fundo da letra).

```
procedure Tprint;  //Imprime uma frase apontada por ebx
begin
  asm
  @@PRIN0:
    mov al,[ebx]    //Pega o caracter
    inc ebx         //Próxima posição
    cmp al,252      //Terminou a frase?
    jz  @@PRIN8     //Sim
    cmp al,9        //É repetição de letras?
    jz  @@PRIN3     //Sim
    cmp al,12       //É código de posicionamento?
    jz  @@PRIN1     //Sim
    cmp al,5        //Define cor de escrita?
    jz  @@PRIN5     //Sim
    cmp al,6        //Define cor de fundo?
    jz  @@PRIN6     //Sim
    cmp al,7        //Liga/desliga impressão do fundo?
    jz  @@PRIN7     //Sim
    push ebx
    call ImpChrs    //Imprime o caracter válido
    pop  ebx
    jmp @@PRIN0     //Próximo caracter
  @@PRIN1:
    mov ah,[ebx]    //Ajusta impressão para linha
    mov [Linha],ah  //e coluna, á seguir
    inc ebx
    mov ah,[ebx]
    mov [Coluna],ah
  @@PRIN2:
    inc ebx
    jmp @@PRIN0
  @@PRIN3:
    mov ecx,0
    mov cl,[ebx]    //Quantidade de repetições
    inc ebx
  @@PRIN4:
    mov al,[ebx]    //Pega o caracter
    push ebx
    call ImpChrs    //Imprime
    pop  ebx
    loop @@PRIN4
    jmp @@PRIN2
  @@PRIN5:
    mov al,[ebx]    //Ajusta a cor de impressão
    inc ebx
    mov [Ink],al
    jmp @@PRIN0
  @@PRIN6:
    mov al,[ebx]    //Ajusta a cor do fundo
    inc ebx
    mov [Paper],al
    jmp @@PRIN0
  @@PRIN7:
    not [Flgfun] //Liga/desliga impressão do fundo
    jmp @@PRIN0
  @@PRIN8:
  end;
end;
```

Viu como a rotina é simplezinha? EBX está sempre apontado para o caracter a ser impresso ou processado como uma espécie de código de controle. Quando ele acha o valor 252, sinaliza para a rotina que a mensagem acabou.

Mas note o detalhe, no início da rotina:

```
mov   al,[ebx]      //Pega o caracter
      inc   ebx     //Próxima posição
      cmp   al,252  //Terminou a frase?
```

Quando ela compara e identifica o 252, EBX já está apontando para o próximo byte. Para a rotina Tprint isto não tem a menor importância, mas para a rotina Disp...

```
procedure Disp;     //Imprime a mensagem
definida à seguir
begin
   asm
      pop  ebx
      call Tprint
      push ebx
   end;
end;
```

Nossa mãe, não entendeu nada, né não? Bem, a explicação é a seguinte: sabemos que o processador tem um registrador que é usado para indicar (para ele mesmo) o endereço de qual instrução está executando. Depois de executada, este registrador (que se chama IP - Instruction Pointer e que não deve nunca (he, he, he...) ser manipulado pelo programa) guardará o endereço da próxima instrução.

Ao pintar uma chamada à procedures ou rotinas, o valor desse registrador vai para uma área chamada stack pointer e fica guardado lá até que o processamento precise voltar para o ponto de onde foi desviado. É uma espécie de endereço de retorno, senão o programa fica maluco e não saberá voltar para a instrução imediatamente posterior à chamada.

O que nós vamos (sacanamente) fazer é "pegar" esse endereço de retorno (nossa mensagem está logo depois da chamada, lembra?), usá-lo como se fosse o EBX da rotina Tprint e depois devolvê-lo atualizado para o stack pointer.

Esse tipo de coisa merecia o Oscar de efeitos especiais em programação. É de uma simplicidade quase angelical, apesar de todos os perigos que cerca a manipulação de um endereço de retorno. Mas acredite, funciona como um relógio suíço.

E digo mais, como é uma coisa muito fora do normal, qualquer hacker que tentar entender o seu programa, vai fundir o cérebro.

Bão, aqui está um pedaço de código para testar nossa rotina de impressão de strings, não esquecendo do que antes precisamos remover o cursor do mouse e depois recolocá-lo novamente.

```
procedure TForm1.FormClick(Sender: TObject);
begin
   Paper:= 4; Ink:= 14; Linha:= 20; Coluna:= 40;
   RepoeArea;
   asm
      mov  ebx,OFFSET @@Msg1
      call Tprint
      jmp  @@Lp0
   @@Msg1: db   'Alo',252
   @@Lp0:
       call Disp
       db   12,10,10,'Testando... 1,2,3!',252
   end;
   GetArea;
end;
```

É isso aí. Gostou, não gostou? Só falta agora aquele hilário "dãmmmm". Nada disso. Aqui não ocorreu nada de sobrenatural mas tão somente reminiscências de um tempo que a memória não passava de 1 Kbyte e a velocidade de processamento não ia a mais que 2 Mhz.

Mesmo hoje, pilotando um potente 300 Mhz, você pode sentir-se tentado a achar que está montado na máquina. Mas eu lhe digo: não está não. A menos, é claro, se puder descer até a casa das máquinas e dar uma boa futucada no motor do barco. E isso, cursinho nenhum de final de semana vai lhe proporcionar. Só aqui na Micro Sistemas e (é claro) na TILT online.

RENATO DEGIOVANI *é autor de diversos trabalhos editoriais na área de programação de jogos em computador. Pode ser contactado pelo e-mail: degiovani@orlatec.com.br*

ARTIGO

Menu para Clipper PopDown

Por: Horacimar Pinheiro Cotrim

Aprenda com Horacimar Pinheiro Cotrim, como desenvolver funções para a utilização de menus especiais para clipper popdown.

Todo programador de Clipper que alguma vez já executou programas que apresentavam menus com rolamento do tipo PopDown, como pôr exemplo o editor Edit® ou até mesmo os programas para Windows® já pensou alguma vez em transportá-los para suas aplicações em Clipper.

No entanto, esse desejo nem sempre se tornava realidade, porque ó Clipper não apresenta uma função deste tipo para satisfazer os sonhos dos amantes do Clipper.

Mas existem no mercado algumas bibliotecas que apresentam funções deste tipo, como a Fast e outras mais. O problema era que você teria que desembolsar uma boa quantia de dinheiro para obtê-las, não esquecendo que teria que fazer contato com um outro país.

Um outro problema era que funções deste tipo eram normalmente escritas em linguagem Assembly, o que tornava realmente difícil montá-las. "Não é qualquer um que programa em Assembly".

Foi pensando nestes fatos e em outros que resolvi desenvolver uma função que tornasse possível a utilização destes menus especiais. O código foi escrito totalmente (100%) em Clipper e para construí-la foi necessário a utilização de matrizes multidimencionais.

A função abrange as versões superiores e inclusive a Clipper 5.01. Para versões inferiores deve-se fazer algumas modificações.

Como a função funciona ?

Função

Menu_Edit(<Lin>,<Col> , <MenuH>, <MenuV> , <Vert-Log> , <FuncaoV>)

Parâmetros de Entrada

Lin.....: Linha onde será escrito o menu horizontal.
Col.....: Coluna onde será escrito o menu horizontal.
MenuH.....: Matriz que contém os dados do menu horizontal.
MenuV.....: Matriz que contém os dados do menu vertical
Vert-Log..: Matriz que contém os dados dos itens selecionados ou não.

FunçãoV..: Matriz que contém os nomes das funções a serem executadas.

Parâmetros de Saída

Retorna valor nulo.

Compilação

Clipper Exemplo /m /n.

Clipper Menu /m /n.

Link-Edição

Rtlink Fi Exemplo, Menu
Ou
Blinker Fi Exemplo, Menu
(mais recomendável).

Até a próxima!

***Horacimar P. Cotrim**, tem 17 anos e programa em Clipper e Borland C++. Programa em Clipper desde 1996.*

Eventuais contatos com Horacimar podem ser feitos pelo endereço :
Rua Euclides da Cunha, 866.
Bairro: Centro.
CEP 17800-000 Adamantina – SP.

=== Listagem 1 : Menu.Prg ===

```
/*
   Autor : Horacimar Pinheiro Cotrim

Menu PopDowm -Versão 1.0 -CopyRight by Horacimar  P.  Cotrim
   100% CA - Clipper 5.2e
*/

#Include "Inkey.Ch"
#Include "Achoice.Ch"

#Define Max(X,Y)            ( If(X > Y , X, Y ) )
#Define Min(X,Y)            ( If(X < Y , X, Y ) )

Function Menu_Edit(Nlin,Ncol,H_Menu,V_Menu,V_Mlog,Caction)
Local Nmenu := Len(H_Menu)
Local Cmenu := H_Menu
Local Npop  := Len(V_Menu)
Local Cpop  := V_Menu
Local Tpop[Npop]
Local Ntmenu[Nmenu]
Local Ninc, Copcao

Set Wrap On

If !(Npop == Nmenu)
   Return Nil
Endif

Ninc:=1
For I:=1 To Npop
  Tpop[Ninc] := Len(Cpop[I])
  Ninc++
Next

For I=1 To Nmenu
  Cmenu[I] += Space(2)
Next

Ninc:=1
For I=1 To Nmenu
   Ntmenu[Ninc] := Len(Cmenu[I])
  ++Ninc
Next

Ninc:=1
For I=2 To Nmenu
   Ntmenu[I] += Ntmenu[Ninc]
  ++Ninc
Next

While .T.
   If Lastkey() == K_RIGHT .Or. Inkey() == K_RIGHT
        Keyboard  Chr(K_RIGHT) + Chr(13)
     Elseif Lastkey() == K_LEFT .Or. Inkey() == K_LEFT
        Keyboard  Chr(K_LEFT) + Chr(13)
   Endif

   @ Nlin, Ncol Say Replicate(" ",80)

   Ninc:=1
   For I=1 To Nmenu
     If I=1
       @ Nlin,Ncol Prompt Cmenu[1]
     Else
       @ Nlin,Ncol+Ntmenu[Ninc] Prompt Cmenu[I]
      ++Ninc
     Endif
   Next

   Menu To Copcao
   Pop:={}

   I:=1
   While I <= Npop
     Ninc:=1

     While Ninc <= Tpop[I] .And. I == Copcao
        Aadd(Pop,Cpop[I][Ninc])
       Ninc++
     Enddo
    I++
   Enddo

   If Lastkey() <> 27 .And. Copcao<>1
  Popdown(Nlin, Ncol, Pop,Tpop[Copcao], Ntmenu[Copcao-
1],Copcao,V_Mlog[Copcao],Caction)
   Elseif Copcao == 1 .And. Lastkey() <> 27
                  Popdown(Nlin,Ncol,Pop,Tpop[Copcao],
Ntmenu[Copcao],Copcao,V_Mlog[Copcao],Caction)
   Else
     Exit
   Endif

   Enddo
   Return Nil

   Function Popdown( Lin ,
Col,Mpop,Npop,Col_2,Key,Llogic,Caction)
   Local Oldscreen
   Local Citem := Mpop
   Local L1 := Lin + 1
   Local L2 := L1 + Npop
   Local Ntam[Npop]
   Local C1,C2
   Local Ninc, I
```

```
If Key == 1
   C1 := Col
Else
   C1 := Col_2
Endif

Ninc:=1
For I:=1 To Npop
  Ntam[Ninc]:= Len(Citem[I])

  Ninc++
Next

Ninc:=0
I:=1

While I <= Npop
If Ntam[I] > Ninc

  Ninc:=Max(Ntam[I],Ninc)
Endif

  I++
  If I=10
    Exit
  Endif
Enddo

C2:= C1 + Ninc + 2

Oldscreen:= Savescreen(L1,C1-1,L2+3,C2+4)
@ L1,C1 Clear To L2+1,C2+2
@ L1,C1 To L2+1,C2+2

Cop:= Achoice(L1+1,C1+1,L2,C2,Citem,Llogic,"Controle")

If !(Cop == 0) .And. !(Caction == Nil)
       Eval(Caction[Key][Cop])
Endif

Restscreen(L1,C1-1,L2+3,C2+4,Oldscreen)
Return Nil

Function Controle(Modo,Elemento,Pos)

Local Nretval:= AC_CONT
Local Nkey   := Lastkey()


Do Case
    Case Modo == AC_EXCEPT
      Do Case
         Case Nkey == K_RETURN
           Nretval := AC_SELECT
         Case Nkey == K_RIGHT
            Nretval := Chr(K_RIGHT) + Chr(13)
         Case Nkey == K_LEFT
            Nretval := Chr(K_LEFT) + Chr(13)
         Case Nkey == K_ESC
            Nretval := AC_ABORT
         Otherwise
            Nretval := AC_GOTO
      Endcase

Endcase
Return Nretval
```

=== Listagem 2 : Exemplo.Prg ===

```
Function Main
Local Pop[5]

Local Logico[5]

Local Acao[5]


Clear
Setcolor("W/+G,Bg+/b,,,W/g")


Dados:={"Arquivo" ,"Editar","Utilit rios","Sobre","Especial"}


Pop[1]:={"Novo","Abrir","Apagar","-----","Sair"}
Logico[1]:={.T.,.T.,.T.,.F.,.T.}
Acao[1] := {{|| Vazio()},{|| Vazio()},{|| Vazio()},{|| Vazio()},{|| Sair()}}


Pop[2]:={"Inserir","Alterar","Apagar"}
Logico[2]:={.T.,.T.,.T.}
Acao[2] := {{|| Vazio()},{|| Vazio()},{|| Vazio()}}


Pop[3]:={"Otimizar","Calculadora","Calendario"}
Logico[3]:={.T.,.T.,.T.}
Acao[3] := {{|| Vazio()},{|| Vazio()},{|| Vazio()}}


Pop[4]:={"Ajuda","Sobre"}
Logico[4]:={.T.,.T.}
Acao[4] := {{|| Vazio()},{|| Sobre(1)}}


Pop[5]:={"Opcional"}
Logico[5]:={.T.}
Acao[5] := {{|| Sobre(2)}}

Menu_Edit(0,0,Dados,Pop,Logico,Acao)

Return Nil


Function Sair
Setcolor("W/N")
Clear
Quit
Return Nil


Function Sobre(n)
If N == 1
   ? "(c) Horacimar P. Cotrim"
Elseif N == 2
   ? "Micro Sistemas"
Endif
Return Nil


Function Vazio
Return Nil


* Fim Exemplo.Prg
```

O MELHOR DA INFORMÁTICA NACIONAL

Quem quer ficar "por dentro" da informática nacional não pode deixar de ler ***Micro Sistemas****. Sempre atenta aos acontecimentos e tendências, Micro Sistemas é a única revista que mais reflete o que acontece no mercado brasileiro. Feita por brasileiros e para brasileiros* ***Micro Sistemas*** *traz sempre em suas páginas os assuntos mais quentes do momento, programas em diversas linguagens, rotinas, livros, cartas de leitores, etc.*

Por tudo isso você não pode deixar de ler ***Micro Sistemas****. .Não perca mais tempo! Garanta o seu exemplar fazendo uma assinatura anual de* ***MICRO SISTEMAS*** *por apenas:*

ASSINATURA ANUAL
(6 EDIÇÕES)
R$ 30,00

Para adquirir números anteriores enviar R$ 5,00 para cada exemplar desejado.

NÃO PERCA TEMPO!
FAÇA JÁ SUA ASSINATURA de MICRO SISTEMAS

Nome: .....
Endereço: .....
Bairro: ..... Cidade: ..... UF: .....
CEP: ..... Telefone: .....
Assinatura: ..... Data...../...../.....
☐ Estou enviando cheque cruzado nominal à PRB informática Editora Ltda, referente à uma assinatura de MICRO SISTEMAS
Banco: .....Ag: .....cheque nº.: .....

Micro Sistemas
Caixa Postal 18347
Rio de Janeiro - RJ
CEP 20722-970
Tel/Fax:(011)261-7841

REPRESENTANTE EM SÃO PAULO:
RAUL FERNANDES
Tel/Fax: (011) 867-8377 -
Bip. (011)3170-0500 e 253-4545 Cod.72370

BITS & bytes

NOVIDADES

CHEGOU O FLASH CAR SAMURAI, O CARTÃO DE MEMÓRIA QUE VAI VIRAR A CABEÇA DO SEU COMPUTADOR PORTÁTIL

Já está no mercado brasileiro o Flash Car Samurai, um cartão compacto de memória "flash"inteligente de última geração, que cabe na sua carteira ou porta-moedas. Fácil de usar, ele chega, segundo a Samurai, como a solução ideal para quem precisa armazenar informações de forma segura, confiável e compatível com os padrões atuais.

O Flash Car Samurai, garante a empresa, vai economizar tempo e encurtar as distâncias, tornando tudo mais rápido, de tal maneira que quem utilizá-lo poderá fazer em alguns meses ou dias, o que muitas vezes leva anos. Afinal, o Flash Car Samurai é, segundo a empresa, a possibilidade de você guardar todos os seus arquivos - dados, voice mail, vídeo clips, e-mails, fotografias e imagens - em um simples e compacto cartão de memória, de fácil consulta, a qualquer momento.

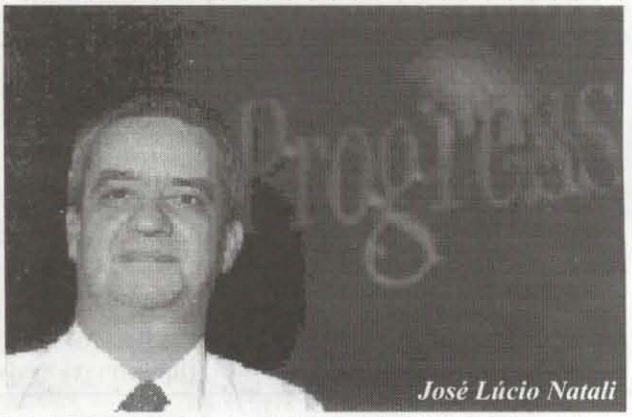
Mais do que isso, o Flash Car Samurai é também o futuro de hoje: ao longo dos anos, todos irão portá-los. As crianças os terão espalhados pela casa, com todas as informações de que precisam. Enfim, o Flash Car Samurai é, de acordo com a Samurai, a informação segura e disponível a qualquer momento, o tempo todo, armazenada num pequeno cartão de memória compacto e inteligente de memória "flash" compatível com os padrões "compactflash" e "PCMCIA".

Fabricado nos EUA, o Flash Car Samurai é o único comercializado no Brasil com uma marca brasileira, a Samurai, e chega ao consumidor através de distribuidores e revendas especializadas em "notebooks", "laptops", "palmtops", ou "PDAS",. Atualmente, está sendo vendido, tanto no varejo quanto no atacado, em unidades de 4 mb, (com possibilidade de armazenamento de 3 disquetes comuns) e 10 mb (equivalente a sete disquetes comuns). Há ainda muitas outras unidades, de 2 até 40 mb, mas só disponíveis a partir de encomendas.

PROGRESS ASSUME AS OPERAÇÕES DA PGS NO BRASIL E INVESTE US$ 20 MILHÕES

A Progress Software Corporation está assumindo as operações , com seus produtos, diretamente no mercado brasileiro. O investimento previsto para os próximos cinco anos é de US$ 20 milhões , entre contratação de pessoal, treinamento de profissionais, instalações de novos serviços, informa a managing director recén-contratado da Progress, José Lúcio Natali.

A atuação direta no mercado brasileiro de 1993 a 1997, a PGS Software distribuiu esses produtos no país - vai representar maior suporte aos usuários, parceiros nacionais e a vinda de software-houses internacionais, informa Natali. Ainda em 1998, a Progress vai contratar 12 novos profissionais para para as áreas técnicas e de vendas, aumentando o seu quadro de pessoal dos atuais 35 para 45 - um crescimento de 36%.



José Lúcio Natali

BITS & bytes

N O V I D A D E S

LIMPADOR DE TELA DA 3M FACILITA TRABALHO EM MICROCOMPUTADORES

Poeira, umidae,. impressões digitais são algumas das inimigas das telas e filtros de computador, de monitores,de retroprojetores e de espelhos. Para limpar adequadamente essas telas e espelhos, a 3M lançou no Brasil o Limpador de Telas 675, que não provoca riscos, nem é inflamável.

O grande diferencial deste Limpador de Telas é o tratamento anti-estático (contra eletricidade), que minimiza a ação dessa força elétrica, uma das principais responsáveis pelo acúmulo de pó nas telas de monitores, pois a eletricidade "atrai" a poeira para a tela.

Para limpar uma tela padrão de 14 polegadas, basta borrifar o Limpador 675 uma vez. O Limpador 675 é vendido em frasco plástico com aplicador e uma almofada com 25 lenços (também com tratamento anti-estático). Uma fita dupla-face acompanha o produto, para que possa ser fixado em local conveniente para o usuário.

NOVOS RETROPROJETORES 3M SÃO ECONÔMICOS, COM ALTA QUALIDADE E RECURSOS AVANÇADOS

A 3M está lançando novos Retroprojetores modelos 1707 e 1710, desenvolvidos para facilitar a vida de empresários e executivos de pequenas e médias empresas, ou profissionais da área de educação, que procuram economia e excelente desempenho em suas apresentações. Além disso, são equipamentos com desin moderno, econômicos tanto na hora da compra quanto na manutenção: a vida útil da lâmpada é prolongada, em média, o dobro da durabilidade das lâmpadas EN, 130 horas.

Os dois retroprojetores têm 2.000 lúmens de luminosidade e possibilitam uma maior aproximação de tela de projeção, devido às suas lentes grande angular. Com maior disponibilidade de espaço para o auditório, são ideais para pequenos ambientes. Os modelos 1707 e 1710 têm placa de exposição com baixa temperatura, o que representa mais conforto para o apresentador e evita o ondulamento das transparências que estão sendo projetadas.

A base desses retroprojetores é construída em metal e em plástico de alto impacto, aumentando a leveza eą durabilidade dos equipamentos. A voltagem dos retroprojetores é 127 V. O modelo 1710 tem como diferencial a troca instantânea da lâmpada: se ela queimar durante a projeção, não haverá interrupção da apresentação. Basta girar um botão e a outra lâmpada se acenderá.

APRENDA INFORMÁTICA SEM SAIR DE CASA COM RICARDO FLORES

Mais informações? Solicite pelo correio Convencional ou visite minha Home Page no endereço HTTP://W3.OPENLINK.COM.BR/RICARDOFLORES

CADA CURSO É COMPOSTO DE:

- Cronograma de aula a ser seguido.
- Apostila pelo Método Tutorial "Passo a Passo" para você treinar cada exercício diretamente em seu PC.
- Disquete contendo os exercícios prontos do Tutorial, para você tirar suas dúvidas.

Desejo receber por carta registrada, os cursos:

CURSO	De R$	Por R$		Qde.	R$
Introdução à Informática até o MS/DOS 6.2	38,00	22,80	x		
Windows 3.1	44,00	26,40	x		
Para Windows 3.1, 3.11 ou 95 ou Cairo:					
Paintbrush dos 8 aos 80 Anos	27,00	16,20	x		
CorelDRAW! 5.0	47,00	28,20	x		
Programação em Visual Basic 16/32 Bits	44,00	26,40	x		
Programação em Delphi 16/32 Bits com SQL	58,00	35,00	x		
HTML - Criando sua Home Page na Internet	27,20	16,30	x		
Word 6.0/7.0 - Editor de Texto	44,00	26,40	x		
Excel 5.0/7.0 - Planilha de Cálculo e Gráficos	45,00	27,00	x		
Para DOS:					
dBase III Plus - Banco de Dados Interativo	27,00	16,20	x		
Programação em dBase III Plus	27,00	16,20	x		
Programação em Clipper 5.01 / 02 (Básico)	35,00	21,00	x		
TOTAL DO PEDIDO:					

Não perca!

Promoção Arrasadora!

Desconto de 40%

Aproveite! Faça já seu pedido!

Pagamento Via Banco

Depósito em numerário a favor de Audit System Serviços Ltda.
Banco Itaú, agência 0934 Vila Isabel, Rio de Janeiro, C/C 22.142-5
Ligue ou passe um Fax para (021) 571-5903 - Ricardo Flores, informando:
seu nome, endereço completo e confirmando o crédito no valor TOTAL DO PEDIDO,
já incluídas as despesas postais.
Prazo de Entrega: **Imediato**.

e-mail
ricardoflores@openlink.com.br

Pagamento Via Correio

Cheque cruzado e nominal à Audit System Serviços Ltda.
Envie para CAIXA POSTAL 25.096 - RIO, CEP 20.552-970,
no valor TOTAL DO PEDIDO, ***já incluídas as despesas postais.***
Prazo de Entrega: **Após a compensação do cheque**.

Nome:		(Fone:
Endereço:		Bairro:
Cidade:	Estado:	CEP:
Data:	Assinatura:	

Optando por enviar cheque pelo Correio, escreva seus dados em letra de forma. Bem legível!

PROJETOR MULTIMÍDIA 3M MP8620 É EXTREMAMENTE LEVE E SUPERPORTÁTIL

O Projetor Mulimídia 3M MP8620, que está sendo lançado no Brasil, é a mais avançada tecnologia em projetores ultra-portáteis no mercado, desenvolvidos especialmente para atender as necessidades dos executivos em constantes ou para uso em salas de reuniões. O Projetor MP8620 pesa menos de 5Kg, tem alta luminosidade (450 ANSI lúmens) e excelente qualidade de imagem.

Tem um prático mecanismo de fechamento, que proteje o equipamento e facilita o seu transporte. É SVGA real (permite o uso de imagems XGA comprimidas, ou VGA expandidas). Projeta 16,7 milhões de cores e tem alto-falantes embutidos estéreo. Tem, também, controle remoto mouse virtual com joystick, que possibilita a emulação do mouse do computador e o comando total da apresentação à distância, de quase todos os pontos da sala de reunião.

É fácil de se instalar e utilizar e sua lâmpada tem vida útil prolongada (1.000 horas). A fonte de energia do Projetor MP8620 é universal (90 a 132V, 198 a 264, 50/60Hz).

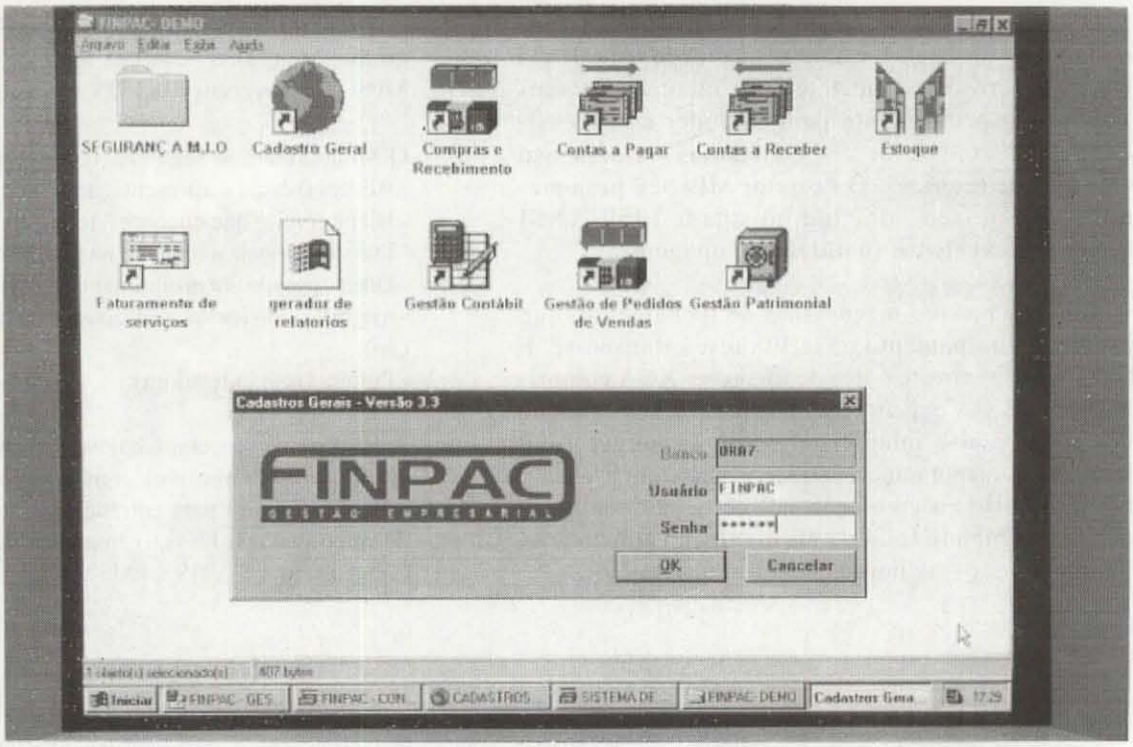
O MP8620 tem os seguintes recursos para apresentação:
- Reveal (revela a apresentação tópico a tópico);
- Blank (efeito que encobre a tela);
- Freeze (congela a imagem na tela);
-Timer (possibilita programar o tempo de apresentação);
- magnification (destaca em zoom uma determinada informação);
- Pointer (seta indicadora).

Além desses recursos, o novo Projetor MP8620 tem dois canais um compatível com microcomputador PC ou Macintosh; outro para entrada de vídeo, que aceita sinais de videocassetes. TVs ou câmaras de vídeo padrões NTSC, PAL, SECAM, S-VHS, e PAL-M.

SISTEMA FINPAC PROMETE SOLUÇÃO PARA ÁREA ADMINISTRATIVA-FINANCEIRA

"Além de apresentar uma proposta de relação custo-benefício mais atraente que os concorrentes, o produto possibilitou uma rápida integração com nosso sistema de tarifação".

Eliane Santina Arcaldi, da Internetcom.



Os softwares de gestão empresarial se tornaram poderosas armas, auxiliando executivos a driblar a competitividade e caminhar rumo a globalização. São tecnologias indispensáveis dentro de qualquer empresa. De olho nesse mercado promissor, a People Solutions, empresa especializada em consultoria gerencial, desenvolveu o Finpac, um sistema administrativo-financeiro e comercial que respeita as características típicas de nossa economia e legislação.

O Finpac é uma solução para a área administrativa-financeira que une conceitos avançados de administração e alto grau de informação. O sistema compõe 11 módulos integrados, que podem ser implantados também isoladamente. São eles: Gestão Contábil, Contas a Pagar, Contas a Receber, Tesouraria, Caixa e Bancos, Gestão Patrimonial, Pedidos, Faturamento, Gestão de Estoques, Compras e Livros Fiscais. Por ser um produto nacional, o Finpac está totalmente direcionado à realidade brasileira, atendendo as necessidade locais da legislação fiscal tributária, além de particularidades administrativas e operacionais. Essas características são entre outras, as principais vantagens sobre os softwares importados, que precisam ser localizados para se adequar ao nosso mercado.

ABRA SUA EMPRESA DE SOFTWARE COM 50.000 PROGRAMAS!

É isso aí. São mais de 50.000 programas entre aplicações gráficas, utilitários em geral, programas comerciais, editores de textos, programas matemáticos, educacionais, utilitários para internet, jogos, etc. Adquira uma cópia deste nosso fantástico acervo de programas, gravados em 15 CD-ROMs, juntamente com o programa de cópia que facilita a transferência dos programas dos CDs para disquete (ou para seu winchester), um programa localizador que acha com facilidade o programa que você precisa em um banco de dados com a listagem completa dos programas incluindo nome, descrição, data, tamanho em bytes, categoria e código.

SIM, todos os 50.000 programas serão seus, para você usar, copiar, vender ou alugar a qualquer hora?

E quanto custa este pacote com mais de 50.000 programas e todos os outros itens descritos acima? Apenas R$1.500,00 que também podem ser pagos em 3 parcelas de R$600,00 ou seis parcelas de R$350,00. Todo o material tem garantia de 5 anos e vem acompanhado de nota fiscal para sua segurança e tranquilidade.

NEMESIS INFORMÁTICA LTDA.
Rua Sete de Setembro, 92 sala 1203-centro-Rio de Janeiro-RJ-Brasil
TELEFONE: (021) 242-0348-fax (021)242-4760
INTERNET:http://www.ibase.org/~nemesistur

ARTIGO

O Modo de Transferência Assíncrona Parte I

Por: Marcello Praça Gomes da Silva

O que é o ATM? tire suas dúvidas,como foi desenvolvido , qual sua verdadeira função , como é orientado, enfim , nessa matéria você encontra suas principais características.

1. COMUTANDO CIRCUITOS, MENSAGENS E PACOTES

A primeira forma de comutação originou-se no sistema de telefonia fixa (na primeira metade do século 20). Chamou-se comutação de circuitos (**circuit switching**) pois eram os próprios circuitos físicos, e somente eles, que estavam diretamente envolvidos.

A comutação de circuitos estabelece um caminho dedicado entre a origem da comunicação e o seu destino (um canal dedicado durante todo o período de conexão). Ela é transparente no que diz respeito à transferência de informação, tem um baixo retardo de propagação (**propagation delay**) e um razoável retardo de conexão (**connection delay**). Sua tarifação é feita pelo tempo total de conexão e não pelo volume transmitido durante esse mesmo tempo.

Na comutação de mensagens (**message switching**) somente um único canal é usado de cada vez, as mensagens são armazenadas enquanto esperam a sua vez de serem transmitidas. Existe um cabeçalho (**header**) para direcionar o encaminhamento das mensagens e não existe "ligação" própria entre a origem e o destino. O retardo das mensagens é um parâmetro aleatório (randômico).

A comutação de pacotes (**packet switching**) apresenta um melhor aproveitamento da faixa passante no nível de rede (entre centrais). Em relação ao usuário, a faixa disponibilizada permanece fixa até se completar a ligação. Os caminhos são compartilhados. As mensagens são divididas em pacotes com cabeçalho para o seu encaminhamento.

Existe um protocolo de acesso (ausência de transparência) e o atraso costuma ser elevado. A interface padrão para a comutação de pacotes é a **X-25** (onde a operadora tarifa o canal em termos do volume transmitido).

Como uma forma de se agrupar as vantagens de cada modo de comutação surgiu a chamada comutação de células **(cell switching**) onde se sobressai o**Modo de Transferência Assíncrona** (**MTA** ou , em inglês, **ATM - Asynchro-**

nous Transfer Mode). Ver a figura número 1.

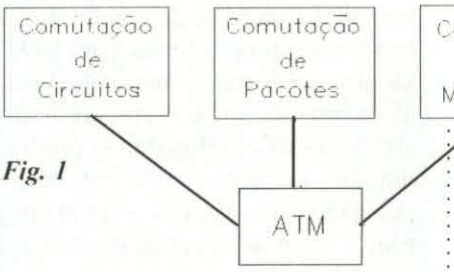


Fig. 1

2. O QUE É O ATM?

ATM é um tipo de comutação de pacotes bem mais sofisticada do que a comutação de pacotes tradicional. Ele combina a comutação de pacotes com o **TDM** (Multiplexação por Divisão no Tempo ou **Time Division Multiplexing**).

É orientado à conexão (**connection-oriented**) pois é necessário estabelecer um circuito virtual entre os dois extremos antes de se iniciar o processo de transmissão.

O **ATM** se aproveita da crescente tecnologia de comutação rápida (**fast-switching**). Ele pressupõe a existência de meios digitais de transmissão de confiabilidade muito elevada (essencialmente a fibra óptica e, com algumas restrições, os sistemas rádio digitais).

O **ATM** foi originalmente desenvolvido no âmbito do antigo **CCITT** (Comitê Consultivo Internacional de Telegrafia e Telefonia) atual **ITU-T** (órgão da União Internacional de Telecomunicações - **UIT**).

A comutação rápida é originária das centrais de comutação baseadas em **hardware** (já que o **hardware** é bem mais rápido do que o **software**). As centrais de comutação rápida (comutação por **hardware**) se aperfeiçoam à medida em que progride a microeletrônica (silício e arsenieto de gálio). O advento da chamada nanoeletrônica (ou eletrônica de dimensões nanométricas) e da eletrônica biológica (**biochips**) significará um grande avanço para todo o setor das telecomunicações (em particular esse da comutação rápida).

Finalmente ,com a implantação dos sistemas de chaveamento óptico(**FOSS- Fiber Optical Switching Systems**) será possível eliminar o chamado gargalo eletrônico (que é a "lentidão" com que a eletrônica processa os sinais em comparação com a rapidez do processamento de um sistema que fosse plenamente fotônico).

Para o **ATM,** o meio de transmissão por excelência é a fibra óptica (tipo monomodal ou, em alguns casos, a fibra óptica tipo multimodal). A fibra óptica multimodal é do mesmo tipo da usada nas redes locais do tipo **FDDI** (**Fiber-Distributed Data Interface**). Entretanto, podemos também considerar o satélite, os cabos coaxiais banda larga (**broadband**), as microondas digitais (rádios digitais) e os cabos de pares trançados (blindados ou não) como possíveis suportes físicos alternativos de transmissão para o **ATM**.

A fibra óptica monomodal deverá ser o meio de transmissão empregado nos enlaces para a interligação de **WANs** (**Wide-Area Networks**) e nas espinhas dorsais (**backbones**) das redes digitais em razão das baixas atenuações, da imunidade às interferências eletromagnéticas, da isolação elétrica entre os terminais, da enorme banda passante (o produto **Largura de Banda versus Distância** é muito alto) e de um grande número de outros fatores.

O **ATM** consegue responder às necessidades mais atuais dos usuários de grande porte. Ele admite um tratamento igualitário para a totalidade dos serviços e é também menos restritivo do que o **TDM** no que diz respeito ao problema de sincronização de rede. Ver a figura número 2.

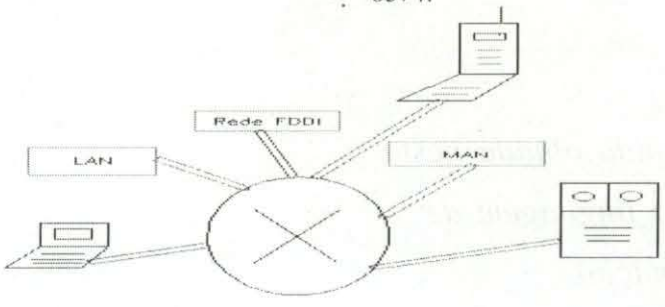


Fig. 2

Atualmente muitos consideram que o **ATM** seja o estágio final no processo evolutivo da tecnologia de redes. Ainda não é possível se vislumbrar qual será o próximo passo de **networking** após o **ATM** (nem mesmo se porventura haverá algum próximo passo).

Muitas outras tecnologias competem acirradamente com o **ATM,** dentre as quais podemos citar o **frame relay**, a **fast Ethernet**, a **gigabit Ethernet**, o **fiber channel**, etc. As duas derivadas da velha **Ethernet** de 10 Mbit/s (10 megabits por segundo) vêm ganhando um espaço muito grande e, seguramente, ameaçam a supremacia esperada do **ATM**.

O resultado desta batalha é uma incógnita. Só nos resta esperar e, enquanto isso, veremos algo mais sobre o **ATM** na próxima edição.

Marcello Praça Gomes da Silva *é engenheiro de sistemas eletrônicos. Pode ser contactado pelo telefone(021) 238-3562 ou pelo* ***TELETRIM*** *531 6374.*

ARTIGO

Visual Basic

Por: Marcelo Elias Del Valle

Uma rápida olhada nesta moderna linguagem de programação

O Visual Basic é uma linguagem de programação desenvolvida pela Microsoft para o desenvolvimento de aplicativos para ambientes Windows, e tem este nome devido a partilhar uma série de comandos da sua antecessora: o Basic, para DOS. Desde a primeira versão, o VB (Visual Basic) tem sofrido diversas modificações, mas a filosofia de programação, no entanto, manteve-se constante: possibilitar o RAD (Rápido Desenvolvimento de Aplicações). O VB, e posteriormente o Delphi, tornaram mais fácil a programação para Windows por permitirem o uso do mouse para a construção da interface de um aplicativo (as telas do programa).

- A Programação para DOS e para Windows:

A programação em linguagens para DOS, como o Clipper, é bastante diferente da programação para linguagens para Windows, como o VB. Em um programa DOS, as ações são tomadas em seqüência, enquanto que em um programa Windows as ações podem ser tomadas em qualquer ordem. O que faz com que a programação do Windows tome ações em ordem aleatória são os eventos.

Os eventos são mensagens que o Windows manda para um aplicativo sempre que o usuário toma alguma atitude, por exemplo: se um usuário dá um clique com o mouse na janela de um programa, o Windows manda uma mensagem ao programa dizendo que um clique foi efetuado nas coordenadas (X,Y) da janela, usando-se determinado botão do mouse (direito ou esquerdo). Em linguagens mais complicadas como o C++, a manipulação dessas mensagens é difícil e trabalhosa, enquanto que com o VB, fica muito mais fácil fazer com que o seu programa responda a essas mensagens.

- Como Desenvolver Programas:

O desenvolvimento de programas em VB basicamente consiste em dois passos:

1. Construção da interface;

A construção da interface é a parte mais fácil, consiste em colocar objetos (Botões de comando, caixas de texto, imagens, etc.) em formulários (janelas), através da barra de ferramentas. Clique em um botão da barra de ferramentas para selecionar um objeto, e então arraste o mouse sobre o formulário para criá-lo. Na figura 1, você vê criado um botão de comando.

Cada objeto tem propriedades, como cor, texto, Altura, Largura, etc. Essas propriedades são ajustadas pela janela Properties, também vista na figura 1.

Fig. 1

Para ajustar a legenda de um botão de comando, por exemplo, você seleciona o botão (clique nele para selecioná-lo), clica na parte da janela Properties onde está escrito Caption e digita o texto da legenda. As outras propriedades são ajustadas de forma semelhante.

2. Programação do código para responder aos eventos necessários.

A escrita do código é a parte da programação para Windows que mais mete medo nos programadores habituados com o DOS. No Visual Basic, você escreve o código para eventos que um objeto ou janela recebe do Windows. Dando um duplo-clique sobre um objeto ou formulário, aparece a janela de edição de código, vista na figura 2. Nela, você pode escolher o objeto que receberá o evento (Cada objeto tem um nome, ajustado pela propriedade name),e o evento que este receberá.

Você pode criar um botão de comando, por exemplo,e mandar o programa exibir uma mensagem na tela sempre que o usuário clicar sobre ele. Para tanto ,você precisa :

1. Dar um duplo clique em qualquer objeto ou apertar F7 para exibir a janela de edição de código;

2. Selecionar o botão que irá receber a mensagem de evento na caixa de combinação esquerda;

3. Selecionar o evento, no caso, CLICK, para receber um clique do mouse.

4. Escrever o código.

Os passos 1, 2 e 3 podem ser resumidos em um se você dar um duplo-clique no botão de comando desejado. A janela de edição de código será exibida e o respectivo botão estará selecionado, e como o evento CLICK é o mais comum, estará selecionado também.

O comando para exibir uma mensagem na tela é: **MsgBox** "Mensagem". *onde* "Mensagem" *é o texto que será exibido (Veja a figura 2).*

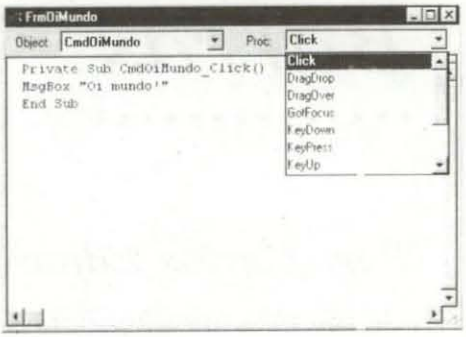


Fig. 2

***MARCELO ELIAS DEL VALLE** é estudante e programa em Visual Basic, C, e C++.*

ARTIGO

Visual Basic 3 Código de Barras

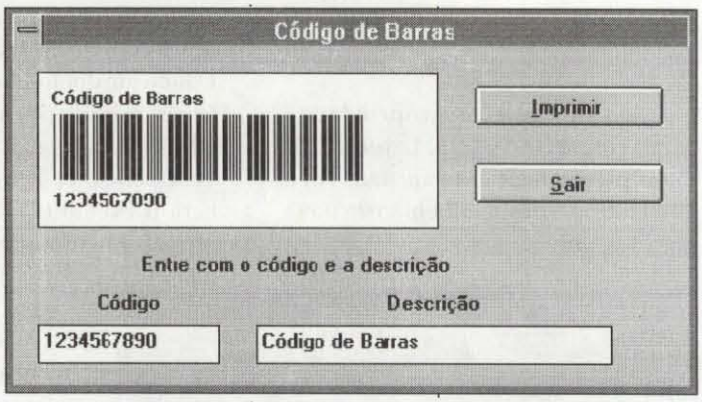
Por: Carlos Eduardo Ambrosi

Não é preciso lembrar da importância do código de barras atualmente, os maiores exemplos estão na utilização em supermercados na identificação dos produtos e nas empresas na identificação e controle de horários dos funcionários. Este programa de código de barras está bem simplificado apenas cria o código e imprime, mas pode ser implementado de acordo com as necessidades.

O programa de código de barras em Visual Basic compõem-se de dois arquivos:

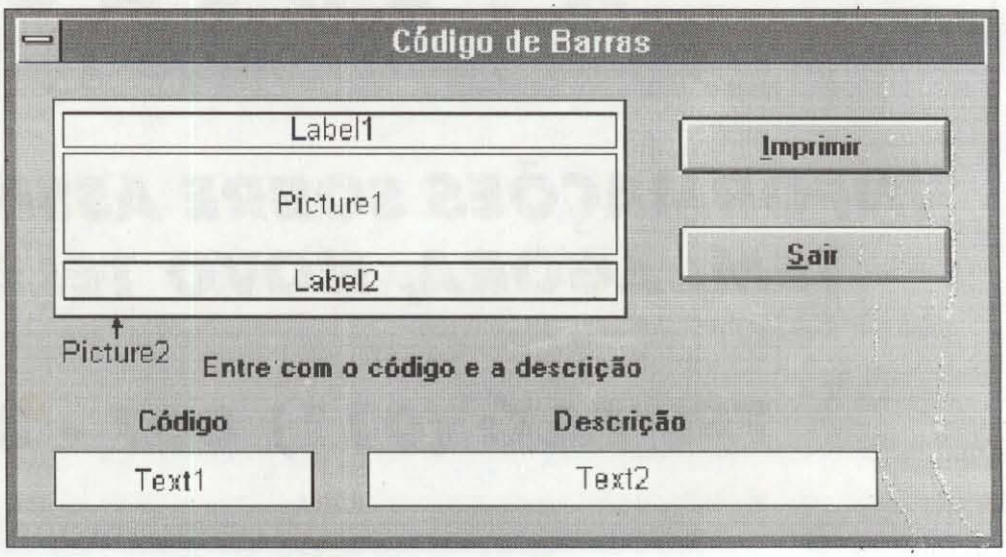
- Um com o formulário (tela) que chamaremos de CODBAR.FRM.

- E outro com as definições e funções que chamaremos de CODBAR.BAS.



Iniciando o Projeto

Menu File - New Project
Encontre Form1, crie os objetos e altere suas propriedades conforme figura e textos abaixo.



```
VERSION 2.00
Begin Form Form1
    AutoRedraw      =    -1  'True
    BackColor       =    &H00C0C0C0&
    BorderStyle     =    3  'Fixed Double
    Caption         =    "Código de Barras"
    ClientHeight    =    3150
    ClientLeft      =    1170
    ClientTop       =    1635
    ClientWidth     =    6315
    FontBold        =    -1  'True
    FontItalic      =    0   'False
    FontName        =    "Arial"
    FontSize        =    8.25
    FontStrikethru  =    0   'False
    FontUnderline   =    0   'False
    Height          =    3555
    Left            =    1110
    LinkMode        =    1  'Source
    LinkTopic       =    "Form1"
    MaxButton       =    0   'False
    MinButton       =    0   'False
    ScaleHeight     =    3150
    ScaleWidth      =    6315
    Top             =    1290
    Width           =    6435
    Begin CommandButton Sair
       Caption      =    "&Sair"
       Height       =    375
       Left         =    4320
       TabIndex     =    1
       Top          =    1080
       Width        =    1755
    End
    Begin TextBox Text2
       Height       =    375
       Left         =    2280
       TabIndex     =    3
       Top          =    2580
       Width        =    3315
    End
    Begin TextBox Text1
       Height       =    375
       Left         =    240
       TabIndex     =    2
       Top          =    2580
       Width        =    1695
    End
    Begin CommandButton Imprimir
       Caption      =    "&Imprimir"
       Height       =    375
       Left         =    4320
       TabIndex     =    0
       Top          =    360
       Width        =    1755
    End
    Begin PictureBox Picture2
       Height       =    1455
       Left         =    240
       ScaleHeight  =    1425
       ScaleWidth   =    3705
       TabIndex     =    4
       Top          =    240
       Width        =    3735
       Begin PictureBox Picture1
          AutoRedraw    =    -1  'True
          BackColor     =    &H00FFFFFF&
          BorderStyle   =    0   'None
          DragMode      =    1   'Automatic
          ForeColor     =    &H00000000&
          Height        =    735
          Left          =    120
          ScaleHeight   =    735
          ScaleWidth    =    3495
          TabIndex      =    5
          TabStop       =    0   'False
          Top           =    360
          Width         =    3495
       End
       Begin Label Label2
          BackColor     =    &H00FFFFFF&
          DragMode      =    1   'Automatic
          ForeColor     =    &H00000000&
          Height        =    255
          Left          =    120
          TabIndex      =    9
          Top           =    1080
          Width         =    3495
       End
       Begin Label Label1
          BackColor     =    &H00FFFFFF&
          DragMode      =    1   'Automatic
          FontBold      =    -1  'True
          FontItalic    =    0   'False
          FontName      =    "Arial"
          FontSize      =    8.25
          FontStrikethru =   0   'False
          FontUnderline =    0   'False
          Height        =    195
          Left          =    120
          TabIndex      =    10
          Top           =    120
          Width         =    3435
       End
    End
    Begin Label Label3
       Alignment      =    2  'Center
       BackColor      =    &H00C0C0C0&
       Caption        =    "Descrição"
       Height         =    195
       Left           =    2280
       TabIndex       =    8
       Top            =    2280
       Width          =    3315
    End
    Begin Label Label4
       Alignment      =    2  'Center
       BackColor      =    &H00C0C0C0&
       Caption        =    "Código"
       Height         =    195
       Left           =    240
       TabIndex       =    7
       Top            =    2280
       Width          =    1695
    End
    Begin Label Label5
       Alignment      =    2  'Center
       BackColor      =    &H00C0C0C0&
       Caption        =    "Entre com o
código e a descrição"
       Height         =    195
       Left           =    1080
       TabIndex       =    6
       Top            =    1920
       Width          =    3135
    End
End
```

Salve o projeto corrente como CODBAR.FRM para o formulário e CODBAR.MAK para o projeto.

Agora atribua a cada objeto do formulário sua função, ou seja, as ações que cada objeto deve executar quando acionado.

Botão Imprimir

Esta função será responsável pela impressão do código de barras portanto ela deve definir a impressora, posições e tamanhos bem como a leitura e interpretação do código.

```
Sub Imprimir_Click ()
SCREEN.MousePointer = 11
AlturaBarra$ = ".5"
Imprime$ = "SIM"
PRINTER.Print " "
PRINTER.Print
LABEL1.Caption
Call CABECARIO
CodBarras$ =
LABEL2.Caption
For A = 1 To
Len(CodBarras$)
DIGITO$ = Mid$(CodBarras$,
A, 1)
Select Case DIGITO$
  Case "0": Call ZERO
  Case "1": Call UM
  Case "2": Call DOIS
  Case "3": Call TRES
  Case "4": Call QUATRO
  Case "5": Call CINCO
  Case "6": Call SEIS
  Case "7": Call SETE
  Case "8": Call OITO
  Case "9": Call NOVE
  Case "A", "a": Call LETRA_A
  Case "B", "b": Call LETRA_B
  Case "C", "c": Call LETRA_C
  Case "D", "d": Call LETRA_D
  Case "E", "e": Call LETRA_E
  Case "F", "f": Call LETRA_F
  Case "G", "g": Call LETRA_G
  Case "H", "h": Call LETRA_H
  Case "I", "i": Call LETRA_I
  Case "J", "j": Call LETRA_J
  Case "K", "k": Call LETRA_K
  Case "L", "l": Call LETRA_L
  Case "M", "m": Call LETRA_M
  Case "N", "n": Call LETRA_N
  Case "O", "o": Call LETRA_O
  Case "P", "p": Call LETRA_P
  Case "Q", "q": Call LETRA_Q
  Case "R", "r": Call LETRA_R
  Case "S", "s": Call LETRA_S
  Case "T", "t": Call LETRA_T
  Case "U", "u": Call LETRA_U
  Case "V", "v": Call LETRA_V
  Case "W", "w": Call LETRA_W
  Case "X", "x": Call LETRA_X
  Case "Y", "y": Call LETRA_Y
  Case "Z", "z": Call LETRA_Z
End Select
ES
Next A
Call RODAPE
PRINTER.CurrentX = 0
PRINTER.CurrentY =
PRINTER.CurrentY +
Val(AlturaBarra$)
PRINTER.Print
LABEL2.Caption
Imprime$ = "NAO"
PRINTER.EndDoc
SCREEN.MousePointer = 0
End Sub
```

Botão Sair

Apenas encerra o programa

```
Sub Sair_Click ()
End
End Sub
```

Form_Load

Esta função é carregada ao iniciar o programa neste caso ela será responsável pelo posicionamento e abertura do formulário na tela de acordo com a resolução.

```
Sub Form_Load ()
LEFT = (SCREEN.Width -
Width) / 2
TOP = (SCREEN.Width -
Height) / 4
FORM1.Show
FORM1.Text1.SetFocus
End Sub
```

Text1 e Text2

Estas funções apenas controlam a digitação do código e descrição.

```
Sub Text1_KeyPress
(keyascii As Integer)
If keyascii = 13 Then
keyascii = 0:
FORM1.Text2.SetFocus :
NOVABARRA
End Sub

Sub Text1_LostFocus ()
NOVABARRA
End Sub

Sub Text2_KeyPress
(keyascii As Integer)
If keyascii = 13 Then
keyascii = 0:
FORM1.Text1.SetFocus :
NOVADESCRICAO
End Sub

Sub Text2_LostFocus ()
NOVADESCRICAO
End Sub
```

Concluído o formulário crie o arquivo de definições e funções.

Menu File - New Module

Novamente salve o projeto atribuindo o nome de CODBAR.BAS para o arquivo de definições e funções.

O arquivo CODBAR.BAS será responsável pela definição das barras, definição dos dígitos em barras e representação do código e descrição.

O arquivo CODBAR.BAS

Comece definindo duas variáveis que serão usadas pelo projeto.

```
Global Imprime$
Global AlturaBarra$
```

As funções a seguir definem os espaços e linhas das barras, sendo elas:

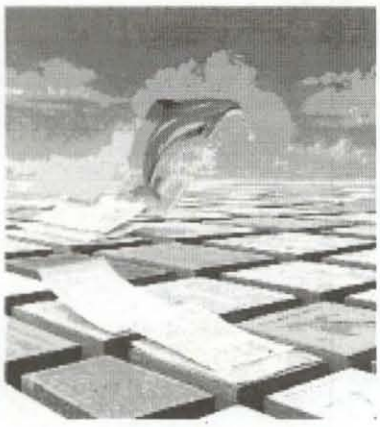
```
ED - espaço duplo
ES - espaço simples
LD - linha dupla
LS - linha simples
```

ONDE VOCÊ ARMAZENA SUAS HOME PAGES PREFERIDAS?

Se você costuma armazenar seus textos em um editor de textos e depois não consegue lembrar do nome dos arquivos...

Se você necessita trabalhar com informações estruturadas em campos e com informações desestruturadas ao mesmo tempo...

Então você precisa do banco de dados textual e hipertexto



askSam

forWindows

askSam é marca registrada da askSam Systems Inc.

Que tal um software que, além de buscar informações por qualquer palavra ou frase do texto, permite que você organize as Home Pages e as suas mensagens do correio eletrônico em bases de dados para posterior recuperação?

Nas versões Monousuário, Rede, Electronic Publisher e Web Publisher.

Consulte-nos:

EGÉRIA

Gerenciamento da Informação

Tel/Fax: (021)571-5372 E-mail: srodas@prolink.com.br

http://www.egeria.com.br

```
Sub ED ()
If Imprime$ = "SIM" Then GoTo Imprimir1
FORM1.Picture1.FillColor = QBColor(15)
FORM1.Picture1.Line (FORM1.Picture1.CurrentX, FORM1.Picture1.CurrentY)-
(FORM1.Picture1.CurrentX + 3, FORM1.Picture1.CurrentY + 40), QBColor(15), BF
FORM1.Picture1.CurrentY = FORM1.Picture1.CurrentY - 40
FORM1.Picture1.FillColor = QBColor(0)
GoTo SAIR_ED
Imprimir1:
PRINTER.ScaleMode = 5
PRINTER.FillColor = QBColor(15)
PRINTER.Line (PRINTER.CurrentX, PRINTER.CurrentY)-(PRINTER.CurrentX + .0375,
PRINTER.CurrentY + Val(AlturaBarra$)), QBColor(15), BF
PRINTER.CurrentY = PRINTER.CurrentY - Val(AlturaBarra$)
PRINTER.FillColor = QBColor(0)
SAIR_ED:
End Sub

Sub ES ()
If Imprime$ = "SIM" Then GoTo Imprimir2
FORM1.Picture1.FillColor = QBColor(15)
FORM1.Picture1.Line (FORM1.Picture1.CurrentX, FORM1.Picture1.CurrentY)-
(FORM1.Picture1.CurrentX + 1, FORM1.Picture1.CurrentY + 40), QBColor(15), BF
FORM1.Picture1.CurrentY = FORM1.Picture1.CurrentY - 40
FORM1.Picture1.FillColor = QBColor(0)
GoTo SAIR_ES
Imprimir2:
PRINTER.ScaleMode = 5
PRINTER.FillColor = QBColor(15)
PRINTER.Line (PRINTER.CurrentX, PRINTER.CurrentY)-(PRINTER.CurrentX + .0125,
PRINTER.CurrentY + Val(AlturaBarra$)), QBColor(15), BF
PRINTER.CurrentY = PRINTER.CurrentY - Val(AlturaBarra$)
PRINTER.FillColor = QBColor(0)
SAIR_ES:
End Sub

Sub LD ()
If Imprime$ = "SIM" Then GoTo Imprimir4
FORM1.Picture1.Line (FORM1.Picture1.CurrentX, FORM1.Picture1.CurrentY)-
(FORM1.Picture1.CurrentX + 3, FORM1.Picture1.CurrentY + 40), ; BF
FORM1.Picture1.CurrentY = FORM1.Picture1.CurrentY - 40
GoTo SAIR_LD
Imprimir4:
PRINTER.FillColor = QBColor(0)
PRINTER.ScaleMode = 5
PRINTER.Line (PRINTER.CurrentX, PRINTER.CurrentY)-(PRINTER.CurrentX + .0375,
PRINTER.CurrentY + Val(AlturaBarra$)), QBColor(0), BF
PRINTER.CurrentY = PRINTER.CurrentY - Val(AlturaBarra$)
SAIR_LD:
End Sub

Sub LS ()
If Imprime$ = "SIM" Then GoTo Imprimir3
FORM1.Picture1.Line (FORM1.Picture1.CurrentX, FORM1.Picture1.CurrentY)-
(FORM1.Picture1.CurrentX + 1, FORM1.Picture1.CurrentY + 40), , BF
FORM1.Picture1.CurrentY = FORM1.Picture1.CurrentY - 40
GoTo SAIR_LS
Imprimir3:
PRINTER.FillColor = QBColor(0)
PRINTER.ScaleMode = 5
PRINTER.Line (PRINTER.CurrentX, PRINTER.CurrentY)-(PRINTER.CurrentX + .0125,
PRINTER.CurrentY + Val(AlturaBarra$)), QBColor(0), BF
PRINTER.CurrentY = PRINTER.CurrentY - Val(AlturaBarra$)
SAIR_LS:
End Sub
```

O código de barras é a representação de caracteres através de linhas e espaços, portanto cada caracter possui sua combinação, as funções a seguir definem estas combinações.

```
Sub CABECARIO ()
LS  'linha simples
ED  'espaco duplo
LS  'linha simples
ES  'espaco simples
LD  'linha dupla
ES  'espaco simples
LD  'linha dupla
ES  'espaco simples
LS  'linha simples
ES  'espaco simples
End Sub

Sub RODAPE ()
LS
ED
LS
ES
LD
ES
LD
ES
LS
ES
End Sub

Sub ZERO ()
LS
ES
LS
ED
LD
ES
LD
ES
LS
End Sub

Sub UM ()
LD  'linha dupla
ES  'espaco simples
LS  'linha simples
ED  'espaco duplo
LS  'linha simples
ES  'espaco simples
LS  'linha simples
ES  'espaco simples
LD  'linha dupla
End Sub
```

```
Sub DOIS ()
LS  'linha simples
ES  'espaco simples
LD  'linha dupla
ED  'espaco duplo
LS  'linha simples
ES  'espaco simples
LS  'linha simples
ES  'espaco simples
LD  'linha dupla
End Sub

Sub TRES ()
LD
ES
LD
ED
LS
ES
LS
ES
LS
End Sub

Sub QUATRO ()
LS
ES
LS
ES
ED
LD
ES
LS
ES
LD
End Sub

Sub CINCO ()
LD
ES
LS
ED
LD
ES
LS
ES
LS
End Sub

Sub SEIS ()
LS
ES
LD
ED
LD
ES
LS
ES
LD
End Sub

Sub SETE ()
LS
ES
LS
ED
LS
ES
LD
ES
LD
End Sub

Sub OITO ()
LD
ES
LS
ED
LS
ES
LD
ES
LS
End Sub

Sub NOVE ()
LS
ES
LD
ED
LS
ES
LD
ES
LS
End Sub


Sub LETRA_A ()
LD
ES
LS
ES
LS
ED
LS
ES
LD
End Sub

Sub LETRA_B ()
LS
ES
LD
ES
LS
ED
LS
ES
LD
End Sub

Sub LETRA_C ()
LD
ES
LD
ES
LS
ED
LS
ES
LS
End Sub

Sub LETRA_D ()
LS
ES
LS
ES
LD
ED
LS
ES
LD
End Sub

Sub LETRA_E ()
LD
ES
LS
ES
LD
ED
LS
ES
LS
End Sub

Sub LETRA_F ()
LS
ES
LD
ES
LD
ED
```

```
LS
ES
LS
End Sub

Sub LETRA_G ()
LS
ES
LS
ES
LS
ED
LD
ES
LD
End Sub

Sub LETRA_H ()
LD
ES
LS
ES
LS
ED
LD
ES
LS
End Sub

Sub LETRA_I ()
LS
ES
LD
ES
LS
ED
LD
ES
LS
End Sub

Sub LETRA_J ()
LS
ES
LS
ES
LD
ED
LD
ES
LS
End Sub

Sub LETRA_K ()
LD
```

```
ES
LS
ES
LS
ES
LS
ED
LD
End Sub

Sub LETRA_L ()
LS
ES
LD
ES
LS
ES
LS
ED
LD
End Sub

Sub LETRA_M ()
LD
ES
LD
ES
LS
ES
LS
ED
LS
End Sub

Sub LETRA_N ()
LS
ES
LS
ES
LD
ES
LS
ED
LD
End Sub

Sub LETRA_O ()
LD
ES
LS
ES
LD
ES
LS
ED
```

```
LS
End Sub

Sub LETRA_P ()
LS
ES
LD
ES
LD
ES
LS
ED
LS
End Sub

Sub LETRA_Q ()
LS
ES
LS
ES
LS
ES
LD
ED
LD
End Sub

Sub LETRA_R ()
LD
ES
LS
ES
LS
ES
LD
ED
LS
End Sub

Sub LETRA_S ()
LS
ES
LD
ES
LS
ES
LD
ED
LS
End Sub

Sub LETRA_T ()
LS
ES
LS
```

```
ES
LD
ES
LD
ED
LS
End Sub

Sub LETRA_U ()
LD
ED
LS
ES
LS
ES
LS
ES
LD
End Sub

Sub LETRA_V ()
LS
ED
LD
ES
LS
ES
LS
ES
LD
End Sub

Sub LETRA_W ()
LD
ED
LD
ES
LS
ES
LS
ES
LS
End Sub

Sub LETRA_X ()
LS
ED
LS
ES
LD
ES
LS
ES
LD
End Sub
```

```
Sub LETRA_Y ()
LD
ED
LS
ES
LD
ES
LS
ES
LS
End Sub

Sub LETRA_Z ()
LS
ED
LD
ES
LD
ES
LS
ES
LS
End Sub
```

As funções a seguir definem a representação do código de barras e a descrição na tela, ou seja, cada vez que você digitar um código e descrição sua representação será automaticamente atualizada.

```
Sub NOVABARRA ()
PrintFlAG = "NAO"
FORM1.Picture1.Cls
FORM1.Picture1.ScaleMode = 3
FORM1.Picture1.CurrentX = 1
FORM1.Picture1.CurrentY = 1
CodBarras$ =
FORM1.Text1.Text
FORM1.Label2.Caption =
CodBarras$
If Len(CodBarras$) = 0
Then GoTo SEMCODIGO1
SCREEN.MousePointer = 11
CABECARIO
For A = 1 To
Len(CodBarras$)
DIGITO$ = Mid$(CodBarras$, A, 1)
Select Case DIGITO$
   Case "0": Call ZERO
   Case "1": Call UM
   Case "2": Call DOIS
   Case "3": Call TRES
   Case "4": Call QUATRO
   Case "5": Call CINCO
   Case "6": Call SEIS
   Case "7": Call SETE
   Case "8": Call OITO
   Case "9": Call NOVE
   Case "A", "a": Call LETRA_A
   Case "B", "b": Call LETRA_B
   Case "C", "c": Call LETRA_C
   Case "D", "d": Call LETRA_D
   Case "E", "e": Call LETRA_E
   Case "F", "f": Call LETRA_F
   Case "G", "g": Call LETRA_G
   Case "H", "h": Call LETRA_H
   Case "I", "i": Call LETRA_I
   Case "J", "j": Call LETRA_J
   Case "K", "k": Call LETRA_K
   Case "L", "l": Call LETRA_L
   Case "M", "m": Call LETRA_M
   Case "N", "n": Call LETRA_N
   Case "O", "o": Call LETRA_O
   Case "P", "p": Call LETRA_P
   Case "Q", "q": Call LETRA_Q
   Case "R", "r": Call LETRA_R
   Case "S", "s": Call LETRA_S
   Case "T", "t": Call LETRA_T
   Case "U", "u": Call LETRA_U
   Case "V", "v": Call LETRA_V
   Case "W", "w": Call LETRA_W
   Case "X", "x": Call LETRA_X
   Case "Y", "y": Call LETRA_Y
   Case "Z", "z": Call LETRA_Z
End Select
ES
Next A
RODAPE
SEMCODIGO1:
SCREEN.MousePointer = 0
End Sub

Sub NOVADESCRICAO ()
FORM1.Label1.Caption =
FORM1.Text2.Text
End Sub
```

O projeto esta concluído, salve-o e prossiga para fase de testes digite o número 238493 e compare com o código abaixo.

teste do programa

238493

Ocorrendo tudo bem crie o arquivo executável.

Menu File - Make EXE File...

Carlos Eduardo Ambrosi *é estudante de engenharia na Unisinos em São Leopoldo - RS e trabalha a aproximadamente a um ano com Visual Basic.*

ARTIGO

Aterramento e Interferência Via rede um assunto muito sério

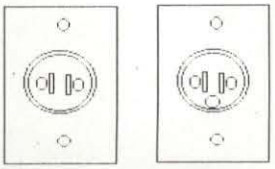
Por: Dálcio Crozero Momesso

Nesta matéria , Dálcio Crozero Momesso,fala das precauções que devemos tomar em relação aos fenômenos de interferência que ocorrem em nosso microcomputador.

ATERRAMENTO

Quando compramos um micro, a gente não vê a hora de instalá-lo e usá-lo o mais rápido possível. E de repente a tomada de energia elétrica não é igual a que vem no cabo de forca que liga o micro.

Fig.1



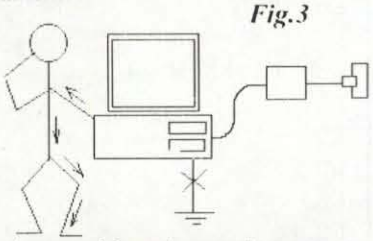
Tomada comum Tomada com plug p/ terra

E aí o que fazer?, muitos nem pensam e com a ajuda de um alicate arrancam o pino central do cabo de forca, que e justamente o plug de aterramento, ou seja, ele esta ligado diretamente a carcaça (parte metálica do micro) que evita eventuais descargas elétricas (choque) que por ventura venha a ocorrer. Tanto pode ser um raio por exemplo ou o simples fato de colocar a mão no micro e encostar o pé no chão, executando assim a descarga elétrica pelo nosso corpo no lugar que era para ser feito pelo plug de aterramento.

Fig.2

Plug do terra

Fig.3



Quando nao existe o aterramento,a pessoa corre o risco de descarga eletrica

Parece ser uma coisa simples, que não exige grandes cuidados ,onde a maior parte do micro são de plástico ou de material nao transmissor de corrente elétrica, mas o que acontece com o circuito eletrônico ? Aí é que está o problema, não tendo o fio terra (aterramento) o circuito ficará sobrecarregado e pode ocorrer a queima de memórias, processadores e até a inutilização do HD.

Muitos acham que o simples fato de se ter um estabilizador e um filtro de linha já estão imune deste fenômeno, mas

não, pois o nosso próprio corpo armazena energia elétrica, e esta mesma energia acumulada em nosso corpo, pode causar a queima do micro.

Cuidado especial devem ter quem montam os micros,onde evitam ao máximo manusear com as mãos os pentes de memórias,o processador e placas. Pois a descarga elétrica pode ocorrer através destes e causar a sua queima. Notamos que a queima de um chip de processador e irreversível , portanto queimou tem que trocar.

Portanto nunca retire o plug central que é usado para aterramento.

Cuidado especial deve se ter também com rede de 220V onde não temos o fio neutro.

E como fazer para ter o fio de aterramento? É muito simples, basta ter um pedaço de ferro (cobre e o melhor que tem) de cerca de mais ou menos de um ou um metro e meio e enfiçá-lo nochão, na terra mesmo, será preciso também comprar uma tomada apropriada que irá encaixar direitinho no mesmo lugar da tomada atual. Conecte um fio (1 1/5) nele e parafuse no pino central da tomada, mantendo os outros dois fios em sua posição inicial, sendo um neutro e outro com fase.Muitos eletricistas usam o neutro como terra, o que de certo ponto e válido pois o mesmo está conectado diretamente ao aterramento do relógio de força de sua residência, mas ao mesmo tempo e inviável, pois podem ocorrer problemas na rede de sua casa (um curto por exemplo) e a fase circular pelo neutro causando assim a queima irreversível do micro.É claro que isso é difícil de acontecer mas se isso ocorrer o gasto de um pedaço de metal e de poucos metros de fio pode sair muito caro, o que se for ver não cobre o valor de seu microcomputador.

Fig.4

E como fazer para quem mora em apartamentos,onde o chão fica longe? Geralmente em prédios mais novos o fio terra já vem instalado nas tomadas (ele tem obrigatoriamente a cor verde) o que deve ser pesquisado com o síndico do prédio.No caso de falta de informação ou certeza o mais prático e confiável seria quebrar + ou - meio metro da parede e chumbar o pedaço de cobre nela, deixando o fio para fora e ligar no pino central da tomada.

Acho que até este ponto você deve estar pensando que sou louco, mas é uma necessidade que pode salvar seu micromicrocomputador ou até mesmo você das descargas elétricas que podem ocorrer em nosso ambiente de trabalho e lazer.

INTERFERÊNCIA

Em nossa residências sempre existem muitos aparelhos elétricos ligados, Tvs, vídeo, Ar condicionado,etc.... mas também existem aqueles aparelhos que possuem motores.

É ai que mora o perigo. Estes aparelhos tipo secador de cabelo , liquidificador , acendedor de fogões, ventiladores, lâmpadas fluorecentes,etc... enfim,todos os aparelhos que tenha motor, pode causar interferências no micro.

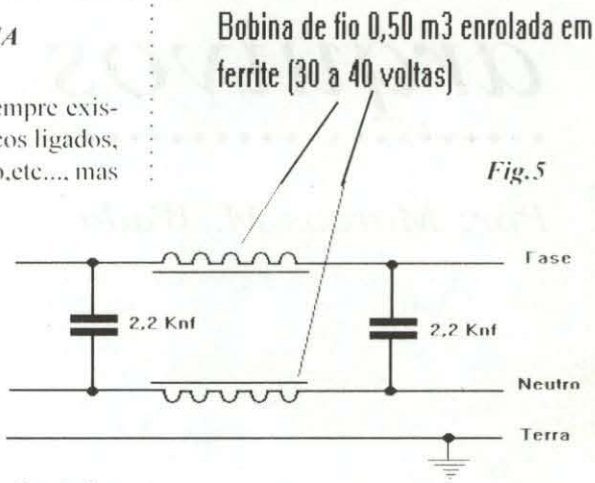
Por que isso ocorre?, é que quando o motor e ligado ele possui imãs que podem produzir um tipo de energia impura devido ao atrito de escovas com o enrolamento do motor, ou quando uma lâmpada fluorecente e acessa, o starter vibra para dar a partida de alta tensão p/ que acenda a luz.Este fenômeno é percebido quando a gente liga um secador de cabelo por exemplo perto de uma TV ligada,surge na tela chuviscos e faixas que são as impuresas causadas pelo imã do motor.

Em se tratando de micro onde os circuitos integrados são super sensíveis, imagine essa impureza atuando sobre eles. Este fato está ligado diretamente a queima do micro ou a detonação do HD, geralmente quando está sendo formatado ou se gravando ou lendo algum dado nele existente. Na maioria dos casos se perde apenas o arquivo que estava sendo utilizado na hora, mas o HD pode sofrer danos.

Um meio prático para se eliminar essa interferência é usar um filtro de linha e ter o aterramento, este que por sua vez joga as impuresas da rede para a terra.

Veja na figura 5. um esquema prático e que pode até ser montado por você, caso tenha aptidão em mexer com eletrônica. Não é difícil de montá-lo e de preço bem baixo.

Fig.5



ESTABILIZADOR

Este é outro item importante, pois o mesmo estabiliza a voltagem que será mandada para o micro. Sua função é de através de reles e de um transformador de isolamento manter a energia normal, evitando sobrecargas que podem queimar a fonte do micro.

***Dálcio Crozera Momesso** é Técnico de eletrônica e eletricidade, programa de informática..*

ARTIGO

Arquivos Secretos. Um método simples de Criptografia de arquivos

Por: Marcos M. Wada

Como o própio leitor pode observar, Marcos M. Wada , expõe um método de criptografia para o uso em QuickBasic , utilizando uma técnica deDaveWhitman, com algumas modificações feitas por ele mesmo.

Apresento um método de criptografia utilizando uma técnica de Dave Whitman adaptada e um pouco modificada por mim para uso em QuickBasic. Ele funciona bem no Qbasic que vem com o MS-DOS mas, recomendo o uso de um compilador devido à sua velocidade.

O método não é muito complicado:
- Cada caracter é lido do arquivo.
- O caracter é substituído de acordo com o código digitado.
- É regravado em um arquivo temporário
- O arquivo original é substituído pelo arquivo criptografado.

O processo da criptografia funciona da seguinte forma: observe a tabela verdade do operador XOR (ou exclusivo).

A	B	S
0	0	0
0	1	1
1	0	1
1	1	0

Agora, observe o que ocorre quando pegamos um caracter, como, por exemplo, M,

M

consultamos seu código na tabela ASCII,

77

convertemos para o binário,

01001101

e mascarando com um caracter qualquer (já em binário, por exemplo, 10101010):

01001101
10101010 XOR
11100111

Obtivemos um caracter especial (11100111 binário, 231 decimal ou E7 hexa). Repetindo-se o processo, usando a mesma máscara, obteremos de volta o caracter original, não necessitando assim a criação de dois programas diferentes para criptografar e descriptografar. Observe:

11100111
10101010 XOR
01001101

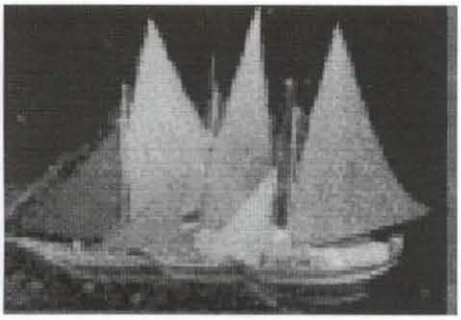
Recuperamos o caracter original 'M'!!!

A seguir a listagem do programa, assim, você poderá estudá-lo com mais cal-

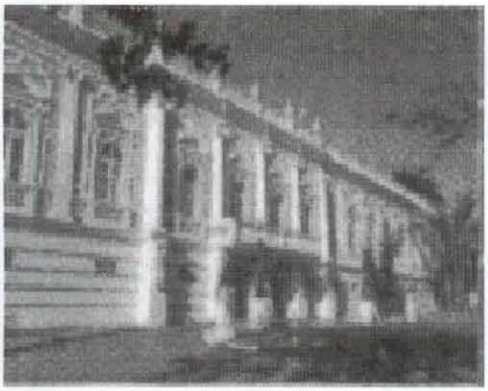
AQUI

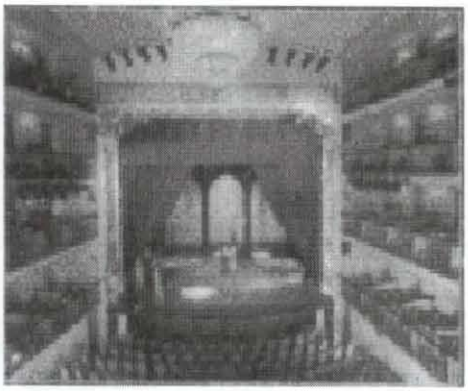
PORTUGUÊS E INGLÊS

CD-ROM



Ilha de São Luís





FOLCLORE * PRAIAS * CULTURA
ECONOMIA * TURISMO * CULINÁRIA

Leve já o seu!

Um super presente para você e sua familia.
Conheça São Luis, seus encantos, suas lendas, sua beleza histórica através da tela de seu computador.

SISTEMA MIRANTE DE COMUNICAÇÃO

RAUL FERNANDES
FONE FAX (011) 867 - 8377
FONE (011) 212 - 3620 R. 20
BIP. (011) 253 - 4545 COD. 72370

MÍDIA INFORMÁTICA

ma e fazer suas alterações e adaptações conforme a necessidade.

Para usá-lo, apenas digite um nome de arquivo e sua senha.O programa identificará arquivos criptografados com um cabeçalho que ele colocará, e para a descriptografar repita o processo.

=== Listagem ===

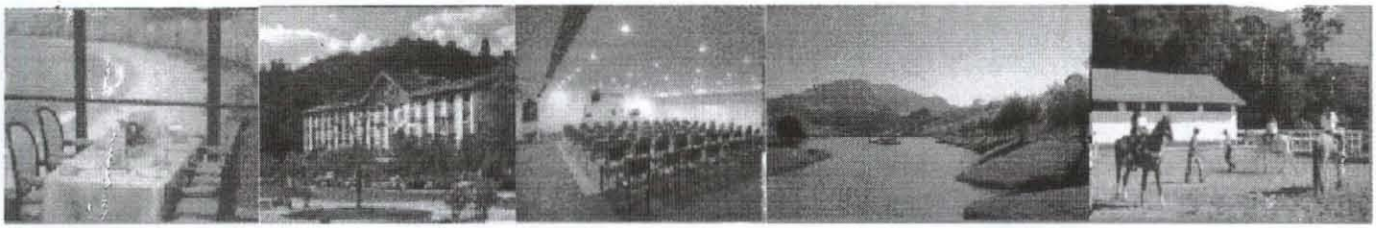
```
DECLARE SUB Box (l1, c1, l2, c2)
DECLARE SUB Tela ()
ON ERROR GOTO ErrorHandler
DIM Mask(1 TO 20) AS INTEGER
Inicio:
CLS
WIDTH 80, 25
Tela
Head$ = "Encrypt!!!" + CHR$(26)
LOCATE 7, 10: PRINT "Digite o nome do arquivo: "; : LINE
INPUT "", file$
REM LOCATE 7, 35: PRINT SPACE$(20)
IF file$ = "" THEN GOTO fim
OPEN file$ FOR INPUT AS #1
CLOSE
OPEN file$ FOR BINARY AS #1
OPEN "encryptd.dat" FOR OUTPUT AS #2
IF LOF(1) < 31 THEN headsize = 0: GOTO code
testhead$ = INPUT$(31, 1)
IF MID$(testhead$, 1, 11) = head$ THEN
  headsize = 31
  testcode$ = RIGHT$(testhead$, 20)
ELSE
  headsize = 0
  SEEK #1, 1
END IF

code:
LOCATE 9, 10: PRINT "Digite o codigo para a (des)criptografia:
: INPUT "", code$
LOCATE 9, 51: PRINT SPACE$(27);
IF code$ = "" THEN GOTO fim
IF LEN(code$) < 5 OR LEN(code$) > 20 THEN GOTO code
code$ = code$ + STRING$(20 - LEN(code$), " ")
msize = 20   'LEN(LTRIM$(RTRIM$(code$)))
FOR x = 1 TO msize: mask(x) = ASC(MID$(code$, x, 1)) XOR
((10 * x) + 5): NEXT
code$ = ""
FOR x = 1 TO msize: code$ = code$ + CHR$(mask(x)): NEXT
IF headsize = 0 THEN
  PRINT #2, head$ + code$;
ELSE
  IF testcode$ <> code$ THEN GOTO code
END IF
mcounter = 1
fs = LOF(1)
FOR x = 1 TO LOF(1) - headsize
  a = ASC(INPUT$(1, 1)) XOR mask(mcounter)
  mcounter = mcounter + 1
  IF mcounter = msize + 1 THEN mcounter = 1
  PRINT #2, CHR$(a);
NEXT
CLOSE
KILL file$
NAME "encryptd.dat" AS file$
LOCATE 23, 3: PRINT "Terminado. Pressione qualquer tecla
para continuar.";
LOCATE 24, 24: PRINT "ou ESC para terminar."
k$ = ""
WHILE k$ = "": k$ = INKEY$: WEND
IF k$ = CHR$(27) THEN GOTO fim
GOTO Inicio

fim:
CLOSE
CLS
SYSTEM


ErrorHandler:
  LOCATE 23, 3
  SELECT CASE ERR

CASE 53: PRINT "Arquivo nao encontrado";          ' File not
found
CASE 64: PRINT "Nome de arquivo invalido";
' Bad filename

 CASE 71: PRINT "Unidade de disco nao preparada";  ' Drive
not ready

CASE 76: PRINT "Caminho nao encontrado";          ' Path not
found
  CASE ELSE: PRINT "Anote este numero e chame pelo
responsavel pelo programa:"; ERR;
  END SELECT
  WHILE INKEY$ = "": WEND
RESUME Inicio

SUB Box (l1, c1, l2, c2)
  FOR l = l1 TO l2
    LOCATE l, c1: PRINT chr$(179);
    LOCATE l, c2: PRINT chr$(179);

  NEXT
  c = c2 - c1
  LOCATE l1, c1: PRINT STRING$(c, 196);
  LOCATE l2, c1: PRINT STRING$(c, 196);
  LOCATE l1, c1: PRINT chr$(218);
  LOCATE l2, c1: PRINT chr$(192);
  LOCATE l1, c2: PRINT chr$(191);
  LOCATE l2, c2: PRINT chr$(217);

END SUB
SUB Tela
  Box 1, 1, 25, 80
  LOCATE 3, 2: PRINT STRING$(78, 196);
  LOCATE 3, 1: PRINT CHR$(195);
  LOCATE 3, 80: PRINT CHR$(180);
  LOCATE 2, 15: PRINT "Criptografador de Arquivos 1.0 por
Marcos M. Wada"

LOCATE 9, 10: PRINT "Digite o codigo para a (des)criptografia:
";

LOCATE 12, 10: PRINT "Tecle [ENTER] sem digitar nome de
arquivo para sair."

LOCATE 14, 10: PRINT "O Codigo de criptografia  nao deve
ser esquecido, pois"

LOCATE 15, 7: PRINT "sem ele voce nao podera recuperar seu
arquivo original."

END SUB
```

NOVA EDIÇÃO

HOTEL ROM

O ÚNICO GUIA DE HOTÉIS EM CD-ROM

NADA MELHOR DO QUE VER PARA ESCOLHER
Com este guia em CD-ROM você pode:



VER: 4.000 fotos "full-screen" de piscina, fachadas, restaurantes, apartamentos, salas de conveção, cenário.

ESCOLHER: Os melhores 2.000 hotéis do Brasil:
hotéis, flats, chalés, pousada, hotel de selva e pantanal, hotel fazenda e outros.

PESQUISAR: Caracteristicas e serviços oferecidos: nome do hotel, localização, cidade, estado, região, tipo de estabelecimento, restaurante, piscina, apto para portadores de deficiência, faixa de preço.

O **HOTEL-ROM** é ferramenta ideal para:
- Executivos
- Turistas
- Coordenadores de eventos
- Operadoras de turismo
- Viagens de negócios e lazer

OS MELHORES 2.000 HOTÉIS E MAIS DE 4.000 FOTOS!

RAUL FERNANDES
FONE FAX (011) 867 - 8377
FONE (011) 212 - 3620 R. 20
BIP. (011) 253 - 4545 COD. 72370

ARTIGO

DRIVELOCK

Por: Adriano C. R. da Cunha
Ligia Camargo

Aprenda como proteger seus discos rígidos e flexíveis contra gravação,permitindo,ainda, que você tenha controle sobre ela.

Você teve aquela estranha sensação de que seu computador estava gritando por socorro e, ao chegar em casa, descobre que seu irmão mais novo apagou o diretório raiz do drive C pra instalar "aquele joguinho que copiou do amigo", enquanto que seu drive E tinha 200Mb de espaço livre?

Ou, ainda, naquela noite de insônia em que você resolve descobrir como fazer acesso a disco pelo BIOS, por um descuido você grava 500kb de lixo no seu HD principal, começando no setor de boot...

O DriveLock pode ajudar a evitar estes "pequenos" inconvenientes. Este programa tem como objetivo proteger seus discos rígidos e flexíveis contra gravação, permitindo, ainda, que você tenha controle sobre ela. É um TSR (terminate and stay resident) que ocupa 2k de memória, podendo ser carregado na parte alta.

O programa possui quatro modos de operação:

· todos os drives do seu PC são protegidos contra gravação;

· todos os drives do seu PC não são protegidos contra gravação;

· a operação de escrita em disco só é permitida se ambas as teclas **Shift** forem pressionadas durante o processo; em caso contrário você receberá um erro de disco protegido;

· quando uma operação de escrita ocorre, uma janela aparece na sua tela, pedindo-lhe para apertar a tecla **Shift** esquerda para continuar ou a tecla **Shift** direita para cancelar a operação;

Atenção quanto ao último modo! A referida janela está em modo texto. Logo, se você estiver em modo gráfico quando a dita cuja aparecer, não irá vê-la e ainda xingará seu computador, pensando que ele travou.

A sintaxe de uso do programa é simples. Basta digitar, no prompt:

C:\>drvlock /opção
Sendo que "/opção" pode ser:

/?, **/H** ou **?** mostra um resumo das opções do programa

/L *protege todos os drives contra gravação (*lock mode*)*

/D *desprotege todos os drives contra gravação (*disable mode*)*

/S permite a gravação somente se ambas as teclas **Shift** estiverem apertadas durante o processo (secret mode)

/V avisa o usuário da operação de gravação e espera permissão para *continuar (**Shift** esquerdo) ou não (**Shift** direito) (*verbose mode*)*

/U desinstala o programa da memória

Atenção: a opção **/D** desprotege todos os drives contra gravação, mas não desinstala o programa da memória. Isso só é feito pela opção **/U** e com a condição de que a interrupção de disco (INT 13h) não tenha sido modificada por outro programa. Em outras palavras: que você não tenha instalado outro programa relacionado à disco após o DriveLock, como o Norton DiskMonitor, TurboImage, etc.

Depois de instalado o programa você pode trocar entre as opções **/L**, **/D**, **/S** e **/V** a qualquer momento, apenas executando-o novamente com a opção desejada.

O DriveLock funciona perfeitamente no MS-DOS 6, MS-DOS 7 (hein?), IBM-DOS, OS/2 DOS Prompt, NDOS e 4DOS. Mas como nem tudo é perfeito, vamos às exceções:

a) No DOS Prompt do Ruindows3.11 o DriveLock funciona, mas tenha cuidado se você estiver em uma janela, e não em tela cheia, pois com a opçao **/V** a janela de aviso não aparece.

b) O DriveLock não funciona no DOS Prompt do Ruindows95 e do RuindowsNT.

c) Alguns programas podem travar se o DriveLock estiver operando com a opção **/V**. O EDIT do MS-DOS 6 é um exemplo.

A listagem 1 apresenta o gerador BASIC do programa DRVLOCK.EXE. Optei por esta forma por dois motivos: o tamanho do código a ser digitado é menor (você não cansa tanto seus dedinhos) e o programa pode ser usado por quem programa ou não em assembly (ou seja, não exige um assembler para a geração do código objeto). Além do mais, se você é programador assembly, pode muito bem usar o DEBUG no programa (não que isso seja melhor que o código fonte)...

Mas voltando à listagem 1, ela pode ser digitada em qualquer dialeto BASIC: QBasic, ASIC, GWBasic e, até mesmo, VisualBasic. O gerador inclui um checksum para verificar se não houve erro de digitação. Mas se você é masoquista, pode entrar os códigos hexadecimais diretamente, utilizando o DEBUG... Só lembre-se que 1735 é 6C7 em hexadecimal.

Uma última explicação: as mensagens do programa estão em inglês por motivos pessoais (nada contra a língua-mãe). Acho que ninguém se importa, nao é?

Até a próxima!

Adriano Camargo Rodrigues da Cunha
Cursa o terceiro ano de Engenharia de Computação na UNICAMP. É autodidata em Assembly, Pascal, C, C++, Forth, Clipper, Perl, VB, QB, Java, Delphi e qualquer outra linguagem que estiver na moda. Quando a faculdade permite, desenvolve programas para PC e (sim!) MSX.

Ligia Camargo
É geóloga formada e trabalha no DNPM do Rio de Janeiro. Por ser mais ocupada que o Adriano, não programa em tantas linguagens. Quando o trabalho permite, faz a mesma coisa que ele: programas pra PC e (sim!) MSX

PROCURA-SE TALENTOS

Dê uma chance ao sucesso liberando seus bytes

*Se você passou horas ou dias, ou por acaso descobriu aquela super-dica sobre uma rotina de uma linguagem de programação, ou sobre a Internet, a Multimídea, o Windows, o OS2 ou o DOS, ou sobre uma planilha, editor de texto, gerenciador de banco de dados, ou editor gráfico que você usa com freqüência, e portanto, **domina**, divulgue seu conhecimento pela Micro Sistemas.*

A Micrto Sistemas é a primeira Revista Brasileira (PRB) de informática e é a única publicação nacional que abre espaço para trabalhos feitos no Brasil, por leitores e para leitores. Esta é a pricipal razão do sucesso imbatível da revista. Afinal, grande parte dos profissionais com projeção neste mercado, foram ou ainda são colaboradores da Micro Sistemas.

O que você esta esperando para fazer parte deste time?

Para que seu trabalho (artigo, crônicas, curso, dicas, programas, rotinas, etc) seja avaliado por nossa equipe é imprescindível que:

- A matéria venha impressa (texto e figuras) e também gravada em disquete;
- As figuras sejam gravadas separadamente;
- Os programas e as rotinas venham com uma versão compilada com exemplos;
- O disquete seja padrão IBM PC (disco 3 1/2");
- O texto seja em formato Word for Windows .

Se aprovada por nossa equipe,a matéria só será publicada se acompanhada de autorização do autor. A autorização é dada quando o autor define o texto de fechamento da matéria, ou seja, nome, endereço, e reduzido histórico de suas atividades. Sempre indicar se deseja a publicação de dados pessoais, tais como, telefone, caixa postal, nome da empresa, e-mail, etc.

O material enviado para a revista não será devolvido.
Os autores das matérias publicadas, além de estarem divulgando conhecimento para milhares de leitores, receberão inteiramente grátis uma assinatura anual de Micro Sistemas.

Enviar o material para:
REVISTA MICRO SISTEMAS
Caixa Postal 18347, CEP: 20722-970,RJ - RJ
e-mail: microsis@nutecnet.com.br
TEL/FAX: (021)261-7841 (DAS 9,00 ÀS 17,00 HS)

ARTIGO

Como fazer um bom Design de Home Pages Pessoais e Corporativos

Por: Renato F. G. Amaral

Este artigo dá algumas dicas sobre como fazer um bom design de home pages na internet,mostra também cuidados básicos que devemos ter ao fazer web sites para médicos,empresários,advogados,etc.

Discutiremos, nesse artigo, os aspectos relevantes da confecção de uma boa home page na Internet. As dicas a seguir refletem aquilo que aprendi a duras penas na prática e talvez servirão para evitar futuras dores-de-cabeça para os leitores da revista.

Um aspecto fundamental do bom design de home pages é o fundo (background): ele deve ser simples, de bom gosto e não atrapalhar a leitura das frases contidas no texto. Cores berrantes, exóticas são péssimas. Sobriedade é a palavra de ordem aqui.

A música MIDI anexa à home page deve ser suave na maioria dos casos. Admitem-se músicas mais "arrojadas" em home pages que transmitam uma idéia de dinamismo, de desembaraço e de excelência profissional. Mas, fora desses casos, a música tranqüila é uma boa pedida.

O texto usado na home page deve primar pela linguagem simples, até mesmo coloquial. Qualquer um deverá entender o que está escrito. Não devemos, entretanto, descuidar da boa gramática, da concordância verbal, da regência nominal, do uso correto dos pronomes, etc. Ser simples não significa ser também inculto.

O bom humor poderá estar presente na Internet, mas cumpre sempre lembrar que algumas pessoas são sérias e não apreciam muito as brincadeiras. A única maneira de averigüar isso é mostrar o trabalho e ver se há uma boa receptividade. Se a pessoa olhar torto, é melhor mudar o design para algo mais sério...

Antes de fazer uma home page, o designer deve ter em mente aquilo que está sendo proposto pelo cliente. Devemos manter esse último sempre em vista, pois a home page é para ele ou sua empresa. Às vezes, o freguês não possui uma nítida idéia daquilo que ele deseja. Nesse caso, é preciso ter paciência e refazer várias vezes o serviço até acertar, obtendo a concordância final do "customer". Certas pessoas somente entendem o que querem ao ver aquilo que não querem. Isso é absoluta-

...iente normal. É difícil acertar logo da primeira vez.

Conselho final: a estrutura da home page deve ser simples, contendo informação bem dosada. Se uma home page ficar muito longa, ocupando mais de duas telas, divida-a em seções ou, até mesmo, crie uma segunda page. A home page principal deve sempre conter um índice que possibilite o acesso às outras pages do site. Essa modularização facilita muito a atualização, de tal modo que não é preciso alterar tudo, bastando editar apenas uma ou duas pages.

Um providencial link de retorno à home page principal (o já mencionado índice) é uma real necessidade para uma boa navegação do site. O uso excessivo de animações e de figuras retardam o download do material via Internet, além de provocar cansaço visual nos internautas que acessam a home page. Uma boa home page deve carregar em cerca de trinta ou quarenta segundos no máximo.

Um contador de acessos pode ser uma boa idéia. Esse dispositivo conta quantos webmaníacos acessaram a sua home page. Geralmente, basta conversar com o provedor de acesso que ele fornecerá um código HTML a ser inserido no final da home page principal (aquela que recomendavelmente deve conter um índice geral).

Se o titular da home page possuir amigos e contatos no estrangeiro, é altamente recomendável redigir uma page em inglês que seja chamada com o seguinte "jump" na home page principal: "To see a version of this home page written in English, please click here."

Quando redigir em inglês, tome muito cuidado para não cometer gafes causadoras de má impressão. Redija com o dicionário ao lado e consulte-o sempre que preciso. Se o seu inglês estiver "estropiado" como o meu, peça ajuda a um amigo que entenda do assunto.

As home pages corporativas devem estar afinadas com os interesses da empresa. Não deve, por exemplo, refletir os pensamentos de apenas um dos sócios, por exemplo . É preciso definir desde o princípio para que irá servir o site empresarial . É apenas para fincar bandeira na Internet ou para , de fato, contribuir para a missão da empresa?

Essas dicas foram empregadas no meu site e no do meu irmão, que é médico. Caso você, caro leitor, queira nos visitar, eis aqui os endereços virtuais:

Renato F. G. Amaral:
http://www.dedalus.net/~renato
Leonardo G. Amaral (meu irmão):
http://www.d edalus.net/~leoamaral

O jeito de fazer uma boa home page consiste em seguir as minhas humildes dicas e experimentar, experimentar, experimentar, experimentar até ficar bom. Seja perseverante que os resultados compensam...Faça bons designs... Torne-se um web publisher de mão cheia...

Caso queira trocar algumas idéias, mande-me um e-mail. O meu correio eletrônico é renato@dedalus.net

Renato F. G. Amaral *é analista de sistemas, escritor e web publisher. Possui quinze anos de experiência de programação pesada. Estuda no Curso Superior de Processamento de Dados da FATEC-SP (Faculdade de Tecnologia de São Paulo).*


Há três anos nos Estados Unidos assessorando empresas e empresários.
Importação • Exportação • Documentação.
Pesquisa de compra nos Estados Unidos, América Latina, Europa e Asia.
Armazenagem, transporte e desembaraço alfandegário.
Computadores- Acessórios tudo em informática- Os melhores preços.
Floppys- fax modems-notebooks-motherboards-multimedia kits-svga monitors.
Equipe altamente qualificada e versátil, à disposição 24 horas por dia para um atendimento rápido e preciso.
Aceitamos todos os cartões de crédito.
E-mail: mgroup2886@aol.com
MF GROUP CORP.
ESTADOS UNIDOS
1001 Brickel Bay Drive, Suite 1714
Miami, Flórida 33131
Tel.: (305) 377-1991
BRASIL (Raul Fernandes)
FONE/FAX: (011) 867-8377
FONE (011) 212-3620 R.20
BIP: (011) 253-4545 cod. 72370

Seu Próximo Destino para Fazer Bons Negócios

EXECUTIVE TRAVEL 98

Feira de Turismo e Serviços para Executivos

12 a 14 AGOSTO de 1998

vão livre do MASP - SP

Apoios:

MASP

SET
SECRETARIA DE ESPORTES E TURISMO DO ESTADO DE SÃO PAULO
COORDENADORIA DE TURISMO

SÃO PAULO
CONVENTION & VISITORS
B U R E A U

RW
PROJETOS & DECORAÇÕES

Informações: **PERSPECTIVA EVENTOS**
TEL/FAX (011) 3115-5386