Spooler

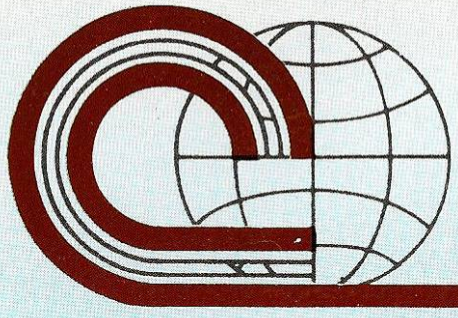
MAGAZINE DOS UTILIZADORES PC

N.º 10 JULHO/SETEMBRO 1990 PREÇO: 400$00

FRACTAIS

PROGRAMAR EM PASCAL (Parte I)

- TOTOLOTO
- UM CONFORTÁVEL ACENTO
- O CARDÁPIO DA DISQUETE
- ADA - UMA LINGUAGEM BÉLICA



Consuldata

SERVIÇOS DE INFORMÁTICA, LDA.

— DISKETTES DE 3" — 3,5" — 5,25" E 8"
BAIXA E ALTA DENSIDADE

— DATA CARTRIDGES (STREAMTAPES)
DE 10 — 20 — 45 — 60 E 150 MEGABYTES

— DISK CARTRIDGES E DISK PACKS
PARA GRANDES SISTEMAS

— TAPE CARTRIDGES PARA IBM 34800
INCLUINDO OS CARROS DE TRANSPORTE
E MÓVEIS ESPECIAIS DE ARQUIVO

— FILTROS ANTI-MAGNÉTICOS E ANTI-ESTÁTICOS
PARA TODOS OS ECRANS DE COMPUTADOR

— PASTAS PARA ARQUIVO DE FORMULÁRIOS
DE 9,5X12 — 12X9,5 — 12X15,5 E 15,5X12
E RESPECTIVOS SEPARADORES

— MOBILIÁRIO PRÓPRIO PARA CENTROS
DE INFORMÁTICA

Consuldata

R. da Quinta do Almargem, 7-A 7-B — 1300 LISBOA - PORTUGAL
Telefs.: 64 48 54 - 64 51 38 - 64 83 06/7/8
Telex: 63755 ULDATA P Telefax: 64 91 08

Nº 10 — Julho/Setembro de 1990

Spooler
MAGAZINE DOS UTILIZADORES PC

MISCELÂNEA DE FIM DE VERÃO

a) A disquete Spooler

Como os leitores se devem ter apercebido, a disquete **Spooler,** a partir do n.º 9, sofreu algumas alterações. Em primeiro lugar beneficiou de um novo desempenho, onde o logótipo da **Spooler** representa o principal papel. **Cor, Som e Movimento** poderia ser o título do*show* com que a disquete nos presenteia, quando, uma vez no palco, chamamos pelo seu nome. O àutor do *script* e dos cenários não se poupou a esforços e conseguiu oferecer-nos um bom espectáculo. Constitui, simultaneamente, um magnífico exemplo daquilo que se pode fazer em *assembly* e o Franco Gomes vai dar-nos o código-fonte para que todos fiquemos a saber como conseguiu aqueles efeitos. Veja o conteúdo da disquete.

E uma ''vedeta'' deste calibre não podia continuar anónima. Agora tem nome e número de contribuinte. Bem-vinda à legalidade!

b) Os "bruxedos" da Spooler

Graças a algumas ''mezinhas'' oportunamente aplicadas, conseguiu-se erradicar grande parte da ''magia'' que embruxava a revista. O número **9** saiu dentro de um prazo muito razoável, mas o número **10** acabou por ser vítima duma moléstia muito própria da época estival — as **férias**! Não que tenhamos algo contra as benditas — muito pelo contrário! Mas a verdade é que as férias ''dos outros'' contribuíram para este novo atraso e contra estas nada pudemos fazer. Esperamos que, pelo menos no que respeita aos nossos actuais e futuros colaboradores, esta época permita que nos ''inundem'' com material para futuras edições.

c) Os Artigos deste número da Spooler

Não nos compete comentar a qualidade do material publicado na revista (e na disquete). Os nossos leitores fazem-no constantemente. Mas não devemos deixar de chamar a atenção para dois dos artigos que enriquecem este número: **Fractais** e **Programar em Pascal**. O primeiro pela sua actualidade e elevado tecnicismo; o segundo pela qualidade a que o seu autor já nos habituou e pelo interesse que despertará aos que se querem iniciar nesta linguagem. Referir estes artigos (e respectivos programas) significará qualquer juízo de valor relativamente aos restantes? De maneira nenhuma. Cabe ao leitor pronunciar-se.

d) A BBspooler

Os trabalhos de instalação correram a bom ritmo. Desde 1 de Setembro que esta se encontra acessível através do telefone **793 87 69** das 18.30 às 10.00 H. No próximo número publicaremos o respectivo regulamento e condições de inscrição.

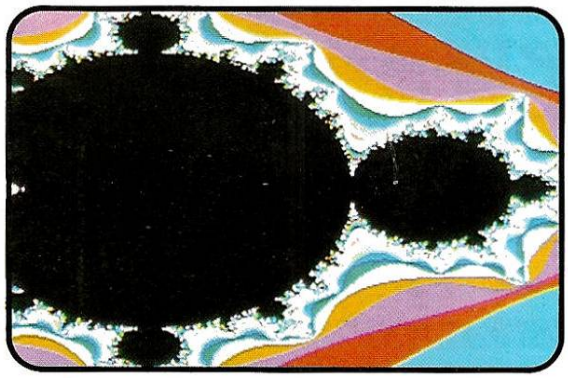
e) O passatempo O melhor Jogo

Já começámos a receber jogos para esta segunda eliminatória. Será altura de definir um prazo para a entrega final: **30 de Setembro.** Os concorrentes que não descurem os aspectos que qualificam um bom jogo. Falámos sobre o assunto no n.º 9 da **Spooler.** Já agora, uma recomendação: não esqueçam as ''fontes'' e não gastem dinheiro no envio das respectivas listagens e outros ''escritos'' — metam tudo na disquete.

Renato Casquilho

Spooler

- **Propriedade:** Graficria, Publicidade e Artes Gráficas, Lda.
- **Director de Edição:** Franco Gomes ■ **Director**: Renato Casquilho
- **Redacção:** João Cardoso, Luís Alho, Paulo Bernardo, Paulo Pinheiro, Vicente Gracias e Eng.º Vítor Guerreiro
- **Colaboradores:** António Bule, Armando Santos, Carlos Albuquerque, Júlio Anjos, Eng. Maia Nogueira, Eng. Miguel Vitorino, Pedro Ferreira e Vítor Hugo Marques
- **Ilustrações:** Franco e Leonor Gomes
- **Relações Públicas e Publicidade:** Jorge Parreira
- **Assinaturas e encomendas:** Pedro Silva
- **Redacção, Publicidade e Serviços Administrativos:** Rua Alfredo Roque Gameiro, 21-1.º Dt.º, 1600 LISBOA — Tel. 76 27 32
- **Produção Gráfica:** Graficria, Publicidade e Artes Gráficas, Lda.
- **Impressão:** Empresa Litográfica do Sul, S.A. ■ **Distribuição:** Electroliber
- **Revista mensal** ■ **Preço:** 400$00 ■ **SRIP N.º** 213900
- **Tiragem:** 12000 exemplares ■ **Dep. Legal N.º:** 28587/89



INFORMÁTICA EM NOTÍCIA

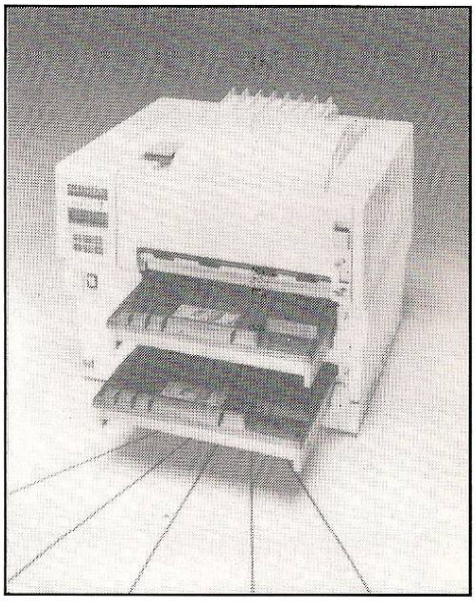
CONSUMÍVEIS ORIGINAIS IBM NA CONSULDATA

A Consuldata foi nomeada, pela IBM, a partir de Outubro do ano passado, concessionário autorizado para a comercialização de todos os consumíveis originais daquela marca.
Encontra-se, assim, disponível um vasto leque de produtos, entre os quais: fitas para máquinas de escrever e impressoras, suportes magnéticos tais como disquetes, bandas, *tape cartridges*, cadeias de impressão (*print belts*), esferas e margaridas nos diferentes tipos de letra para máquinas de escrever. Estes produtos poderão, agora, ser encontrados também na Consuldata.

ASI INFORMATIZA GASODATA

A Gasodata adquiriu recentemente equipamento informático à Araújo & Sobrinho Informática (ASI). A solução adquirida, que está já em funcionamento, engloba as componentes de software ASI e de hardware da marca Olivetti, e permitirá à Gasodata uma gestão financeira, comercial e administrativa integrada. A Gasodata pretendeu, assim, optimizar os seus serviços, recorrendo às novas tecnologias de informação para poder responder com maior rapidez e eficiência às necessidades do mercado.
Como representante exclusiva em Portugal da marca de gasolina "Dresser Wayne", a Gasodata conta com grande prestígio no mercado português fornecendo neste momento algumas das maiores empresas do sector de combustíveis.
A Araújo & Sobrinho Informática é uma empresa de informática do grupo Araújo & Sobrinho que comemorou 160 anos de existência no ano transacto. A empresa dispõe de uma solução de software próprio que corre em Unix, Xenix e MS-DOS, se destina a responder às necessidades das diferentes áreas da gestão empresarial e está devidamente articulada com uma linha completa de hardware das marcas Olivetti e Altos, suas representadas.
Os produtos Altos serão complementados brevemente com a linha de produtos Acer, resultante do recente negócio entre as duas empresas em que a Acer comprou 50% das acções da Altos, prevendo o acordo a comercialização de produtos Acer com a marca Altos.

PROLASER 12



A ProLaser 12 da Citizen

A Prolaser 12 é a primeira impressora laser com emulação PostScript lançada pela Citizen. Destinada ao utilizador mais exigente, esta impressora imprime 11 páginas por minuto utilizando um mecanismo Xerox, o primeiro produto derivado de um acordo OEM entre a Rank Xerox e a Citizen Europe.
Garantindo a compatibildiade com uma vasta gama de aplicações, a Prolaser 12 emula HP Laserjet II, IBM Proprinter II, HPGL 7475A, Epson FX 80 e Diablo 630 ECS. A emulação PostScript é opcional através de uma *cartridge*.
A Prolaser 12 possui 11 fontes residentes compatíveis HP (6 em HP Laserjet II e 5 em HP'F) e permite ainda um leque de fontes opcionais através de cartas IC. Ao projectar o mostra

SOLUÇÃO INFORMÁTICA «IOM/TRANSPORTES»

A IOM – Informática, Organização e Métodos, Lda. disponibilizaou uma solução informática para gestão de frotas de empresas transportadoras, baseada num profundo estudo deste sector. Esta solução assenta num mecanismo de controlo de serviços intimamente ligado com os recursos de manutenção, permitindo às empresas leituras aprofundadas dos resultados do exercício em cada momento da actividade.
Dispondo de uma completa colecção de mapas, o gestor pode controlar desde o nível de *stocks* até aos custos oficinais, passando pelos consumos de combustíveis, *stock* de pneumáticos, gestão de serviços por camião e motorista, etc.
Instalada inicialmente em duas empresas do sector, o *package* IOM/Transportes encontra-se pronto a ser comercializado a nível nacional, estando preparado para trabalhos em multiposto, permitindo o uso de multi-sistemas para gestão de frota nacional e internacional.
Este *package* vem tornar mais rico o panorama do software produzido em Portugal para a gama Atari ST, linha de software profissional quase toda da responsabilidade da IOM.

DISPONÍVEL VERSÃO 6.06 DO SAS SYSTEM

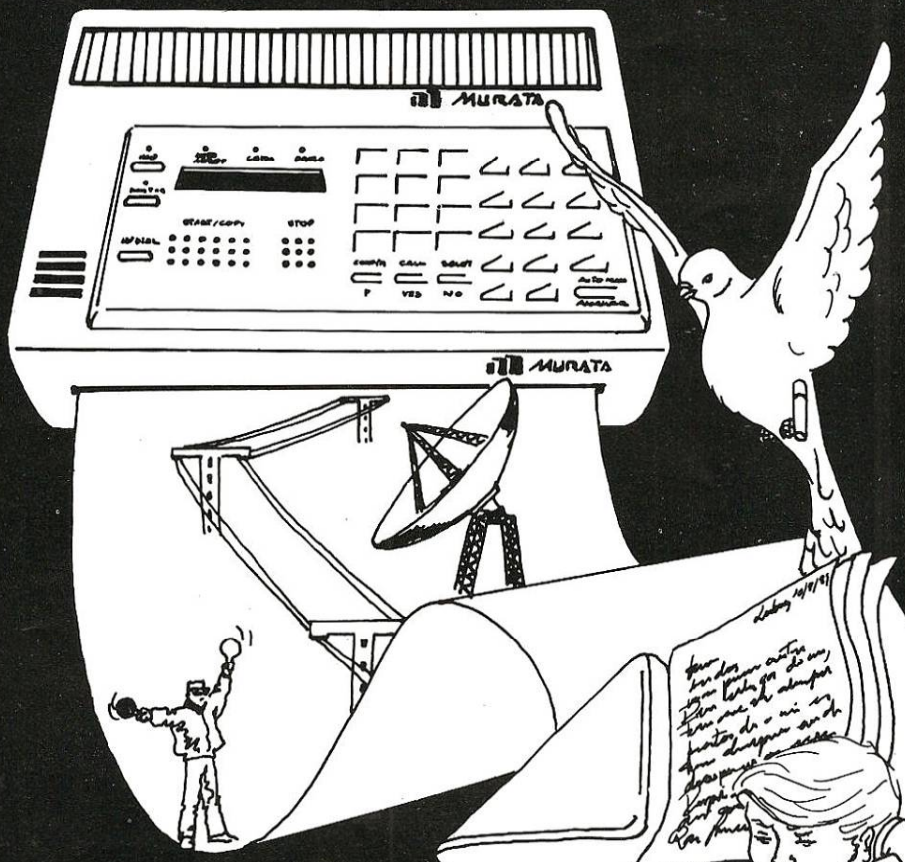
A CICS – Consultoria Informática e Comercialização de Sistemas, Lda. anuncia a disponibilidade no mercado português da versão 6.06 do SAS System, recentemente apresentada na conferência SEUGI.
A conferência SEUGI é o encontro anual dos utilizadores do SAS System na Europa, que este ano teve lugar no Palácio dos Congressos em Paris e reuniu mais de 700 representantes de 21 países, tendo sido apresentadas 90 comunicações, descrevendo as mais diversificadas experiências com a utilização do SAS System, bem como as novas facilidades da versão apresentada, que permitem disponibilizar qualquer aplicação para qualquer utilizador de mainframe, minicomputador, estação de trabalho ou PC.
Também foi anunciado na conferência SEUGI que o SAS Institute e a IBM fizeram um acordo tendo em vista o desenvolvimento comum em várias áreas, bem como a diponibilização do SAS System para ambiente AIX, OS/2.
O SAS System é um sistema modular que permite combinações diversificadas em função do tipo de aplicação que se pretende: desenvolvimento de aplicações; manipulação e gestão de conjuntos de dados; geração de relatórios e gráficos; análise estatística e matemática; planeamento, previsão e apoio à decisão; investigação operacional e gestão de projectos; controlo de qualidade e concepção de protótipos.
Este sistema está disponível para correr em variadíssimos ambientes.

PHILIPS E AIRC INFORMATIZAM CÂMARA DE CASCAIS

Na sequência do Concurso Público realizado no início deste ano pela CM Cascais com o intuito de informatizar todas as suas áreas administrativas e técnicas, a Philips Portuguesa, S.A. foi a empresa que, no entender daquela edilidade, reuniu melhores condições para levar a efeito o projecto de informatização posto a concurso.
O sistema informático a fornecer pela Philips é baseado no processador Motorola 68030 a 50MHz com capacidade para se integrar numa rede local e aceder, através da rede Telepac, a outros sistemas que a câmara venha a instalar noutros serviços remotos da autarquia. Este sistema, o primeiro da sua categoria a ser instalado em Portugal, coloca deste modo a CM Cascais numa posição privilegiada em relação aos outros municípios. Ao sistema terão acesso, numa fase inicial, cerca de 30 utilizadores, podendo o mesmo vir a ultrapassar os 100 utilizadores.
Por acordo entre as três entidades, Philips, AIRC e CM Cascais, a Philips fornecerá todos os meios técnicos necessários à AIRC para esta proceder ao suporte e manutenção do seu software.
Uma das áreas críticas que este projecto pretende atacar é a do atendimento do público que, com este sistema, se espera melhorar significativamente, utilizando para isso, além dos meios informáticos virados para o utilizador da Câmara, os chamados terminais de atendimento, onde o público poderá ter acesso a um vasto conjunto de informações camarárias.

comunicar é FAXil

SELNOE



Murata
TELECOPIADORES

REPRESENTANTE EXCLUSIVO

TELFAX
Equipamentos para Escritório, Lda.

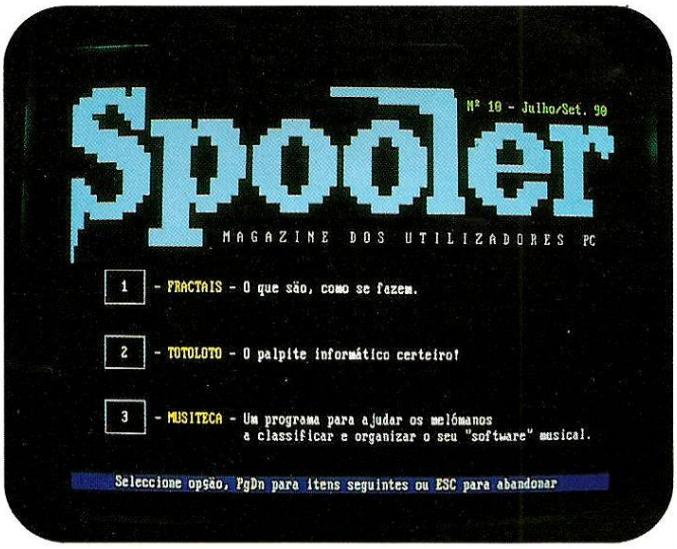
Ligados a todo o mundo funcionamos a todas as horas. Transmitimos com rapidez, economia e qualidade os seus documentos e imagens. Muito mais que úteis somos indispensáveis.

LISBOA - (01) 410 71 13 • PORTO - (02) 31 73 59

Av. Gen. Norton de Matos, 69 - S/L - E — MIRAFLORES — 1495 LISBOA — TELEF. 410 71 13/4 — TELEFAX. 410 70 37
Rua Gonçalo Cristóvão, 348 - 4.º D — 4000 PORTO — TELEFONES: 31 73 59/31 74 99 — TELEFAX. 31 72 73

O Conteúdo da Disquete

O material da disquete Spooler **n.º 10 vai, contamos, encantar os nossos leitores. Vejamos o seu conteúdo por directórios:**



FRACTAIS

O programa executável, a fonte e os *drivers* para as diferentes placas gráficas estão compactados com o nome de FRACTAL.EXE. Para os que contactam pela primeira vez com programas compactados, lembramos que deverão copiá-los para uma disquete (ou disco rígido) e digitar o seu nome: no caso presente será FRACTAL. Terminada a descompactação, podem apagar o FRACTAL.EXE para poupar espaço no disco. O artigo contém todas as explicações necessárias à execução do programa. Mas seja paciente e leia o artigo com atenção antes de tentar correr o programa FRACTAIS: só assim poderá desfrutar de toda a beleza das imagens criadas pelo seu computador.

PASCAL

Conforme o autor nos diz no seu artigo — não deixe de o ler atentamente — JOGO01.EXE é a versão base de um jogo que vai evoluindo à medida que "formos avançando na matéria". Os diferentes programas e "fontes" encontram-se compactados com o título de PASCAL.EXE. O leitor terá de proceder à descompactação como já foi explicado.

INPUT

Neste directório encontra-se a pequena rotina em GWBASIC referida no **Correio dos Leitores.**

ACENTOS

Esta é mais uma abordagem ao problema da acentuação dos textos escritos com teclados feitos para anglo-saxónicos. Para executar o programa ACENTOS.EXE presente neste directório, leia primeiro o artigo; caso contrário, arrisca-se a algumas surpresas...

BANCO

A pedido de várias famílias e, sobretudo, para completar a resposta dada a um dos estimados leitores, aqui vai — finalmente — a fonte em GWBASIC do programa BANCO.EXE apresentado no Nº 8 da **Spooler.**

TOTOLOTO

Neste directório encontra-se o programa TOTO49.EXE e a fonte TOTO49.BAS, compactados para o ficheiro TOTO.EXE. Trata-se dum artigo incluído na rubrica **Colaboração dos leitores,** que mereceu publicação pela sua qualidade.

MUSICA

Os diferentes ficheiros que compõem este programa encontram-se compactados com o nome MUSICA.EXE. Depois da descompactação — a fazer conforme já tem vindo a ser explicado — leia o artigo que, para além das explicações para a sua instalação, contém matéria algo diferente da habitual verborreia informática...

MENU

Conforme referimos no **Editorial**, o autor do novo "figurino" da disquete **Spooler** vai revelar-nos os seus segredos. Neste directório encontram-se as fontes em *assembly* do *menú* da disquete n.º 10. Não deixe de ler o artigo, através do qual certamente aprenderá uns quantos "truques" para embelezar os seus próprios programas.

GANHE UM PSION

Mais um **PSION** foi sorteado por entre todos os assinantes da **SPOOLER**, inscritos até à data.

Foi contemplado o nosso assinante n.º 341, **Vitorino Lobato Dias** de **Matosinhos.**

O **PSION** vai já a caminho, pelo correio.

VIDEOTEX COM SOFTWARE SOFTNET

A Associação de Comerciantes de Materiais de Construção acaba de dar forma a mais uma componente da sua estratégia que aponta no sentido de fornecer, aos associados, informação profissional com a qualidade e acuidade que a agressividade do tecido comercial exige.

Após exaustiva consulta ao mercado fornecedor de soluções videotex, a responsabilidade de montar o seu centro servidor foi confiada a APIX – Desenvolvimento e Comercialização de Produtos Informáticos, Lda., parceiro comercial da softNET, S.A. a actuar no Norte do País, com instalações no Porto.

O software de videotex da softNET, S.A:, complementado por *interfaces* de ligação a bases de dados, permitirá que todos os associados aderentes ao serviço tenham acesso, através de um terminal videotex, a informação técnica e profissional.

LOJAS SINGER COM SOFTWARE BLAST

A Singer escolheu o software de comunicações Blast, representado em Portugal e Espanha pela softNET, S.A., para a implementação de uma rede de comunicações nas suas 44 novas lojas.

Tendo optado por um sistema Nixdorf Unix 386 para as suas novas lojas, a Singer terá disponível o software de comunicações Blast, que possibilitará chamadas automáticas do sistema central para cada loja, a determinadas horas pré-programadas, transmitindo toda a informação relativamente a vendas/encomendas.

O Blast possibilitará, sem qualquer intervenção do operador, conectar-se a cada uma das lojas durante a noite, informando no dia seguinte quais as lojas às quais foi possível a ligação, número de chamadas para a mesma loja, lojas às quais não se conseguiu ligar, ficheiros transmitidos, etc.

O Blast encontra-se disponível para variadíssimos sistemas. A Singer, com 120 lojas distribuídas por todo o País, possui uma das nossas maiores redes de *retail* .

PROGRAMA DE CASE PARA SISTEMAS ABERTOS

A APIE – Software House anuncia o produto Open Case, cuja fin alidade é oferecer um ambiente completo de desenvolvimento de aplicações distribuídas para sistemas abertos utilizando tecnologia CASE – Computer-Aided Software Engineering. A Informix assinou um contrato com a Systematica Limited, uma empresa inglesa, líder em engenharia de software, para colaborar com a Informix no desenvolvimento de produtos OpenCase que irão satisfazer as necessidades americanas e europeias.

O programa OpenCase da Informix aumentará a produtividade dos programadores e a qualidade das aplicações para sistemas abertos através da disponibilização de um ambiente integrado e flexível que suportará todo o ciclo de vida do software – desde as especificações da aplicação até à respectiva manutenção. O OpenCase é desenhado especificamente para satisfazer as necessidades de desenvolvimento de software, do mercado crescente de sistemas abertos.

A Informix planeia ainda oferecer serviços de consultoria e customização dos seus ambientes de desenvolvimento para satisfazer as necessidades dos clientes.

A APIE – Apoio Informático às Empresas, Lda. é uma empresa comercial que tem como actividades principais o desenvolvimento e comercialização de software e a consultoria e formação em tecnologias de informação.

A Informix Software, Inc. é a líder no fornecimento de software de gestão de informação, fornecendo soluções no campo de bases de dados nas mais variadas plataformas.

ACESSO

O acesso ao Ensino Superior tornou-se, nos últimos anos, um processo altamente complexo que exige dos alunos, professores e encarregados de educação um conhecimento detalhado das diferentes opções e cursos disponíveis no país.

O livro publicado pelo Ministério da Educação (Guia de Acesso ao Ensino Superior, 1990) tem como finalidade proporcionar essa informação. Contudo, é notória a grande dificuldade da sua utilização. As consequências deste facto traduzem-se num deficiente conhecimento, por parte dos alunos, das opções que lhes estão abertas e dos pré-requisitos inerentes a essas opções. Em muitos casos, esta ignorância tem tido implicações altamente gravosas e irreversíveis tais como o não acesso ao curso pretendido ou, em casos extremos, não entrada na universidade.

No âmbito de uma disciplina da licenciatura em Engenharia de Sistemas e Informática da Universidade do Minho, um grupo de alunos, sob a orientação do prof. Altamiro Machado procurou analisar o problema tendo em vista a elaboração de um programa que, fazendo uso de técnicas avançadas de desenvolvimento de sistemas de informação, possa auxiliar na consulta da informação disponível sobre os vários cursos superiores e, também, na determinação dos conjuntos de cursos em que um determinado aluno tem maiores probabilidades de acesso. O resultado desse trabalho é um sistema de apoio ao candidato no acesso ao Ensino Superior designado por Acesso e que permite:

a) a consulta inteligente da informação respeitante aos cursos que constam da publicação *Guia de Acesso ao Ensino Superior, 1990,* da responsabilidade da Direcção-Geral do Ensino Superior do Ministério da Educação;

b) consulta a informação variada (tipos de provas específicas, de pré-requisitos, legislação e calendário de provas);

c) introdução das classificações pessoais do utilizador;

d) elaboração de várias simulações de condições de acesso do utilizador em função das suas classificações pessoais.

Desde a produção do programa foram tomadas várias iniciativas que culminaram na sua comercialização, tendo-se iniciado a venda nos finais de Junho passado. Montou-se também uma rede de venda directa junto aos gabinetes de Acesso ao Ensino Superior, às secretarias das universidades e aos locais de realização das provas específicas.

A escolha de um preço para a comercialização deste produto na ordem de magnitude de 10 vezes o preço do suporte físico da informação (isto é, da disquete) e a escolha de canais de comercialização de grande divulgação pode ser a solução para o êxito da comercialização deste produto.

Resta acrescentar o óptimo acolhimento que o ACESSO tem tido por parte das entidades oficiais e o facto de neste momento já estarem em preparação outros produtos para o mercado da educação.

INFORMATIZAÇÃO DA ASSEMBLEIA DA REPÚBLICA

No passado mês de Março deslocaram-se à Assembleia da República representantes da Datinfor (distribuidor exclusivo, para Portugal, da Wang Laboratories, Inc.) que procederam à entrega de microcomputadores portáteis Wang destinados aos gabinetes do presidente da Assembleia, do secretário-geral, do ministro dos Assuntos Parlamentares e dos grupos parlamentares. Estes microcomputadores foram oferecidos pela Datinfor à AR com a finalidade de contribuir para uma maior autonomia destes gabinetes e um mais fácil acesso à informação disponível no sistema de informação instalado naquele Órgão de Soberania.

Na reunião foi referido pelo presidente da AR o seu desempenho na prossecução do projecto, a sua satisfação pelo modo como o mesmo tem decorrido e a sua confiança na equipa técnica dos gabinetes de informática da AR, bem como a confiança depositada no fornecedor.

Como tem sido divulgado, o Projecto de Informatização da AR envolve a instalação de um sistema central Wang VS, onde reside a Base de Dados Parlamentar, bem como uma dezena de outros sistemas Wang VS de menor porte, conectados entre si e ao sistema central através da rede de banda larga "Wangnet". Este projecto, que tem decorrido dentro do planeado e se encontra agora na sua fase final de instalação, tem suscitado uma elevada receptividade por parte dos diversos utilizadores, e é considerado um projecto de vanguarda em instituições similares da Europa.

Correio dos Leitores

POR RENATO CASQUILHO

AS CARTAS E OS TELEFONEMAS

Para além dos muitos telefonemas recebidos na nossa redacção e aos quais procurámos dar a melhor resposta que nos foi possível, desta vez temos, também, muito correio dos nossos leitores. Aproveitamos para pedir que as questões de ordem técnica nos sejam colocadas por escrito pois, não só nos permite um estudo mais aprofundado dos assuntos, como a resposta pode servir a outros leitores. Escrevam sempre.
Mas antes, um pequeno apelo: Pede-se ao nosso leitor Rui Miguel Dias Anastácio, o favor de contactar a nossa revista, para assunto de seu interesse.
Vamos ao correio:

..segui as instruções do seu artigo – fiquei desiludido. Estive a lançar dados, a perder tempo, pois acabou por me apagar tudo... dizendo que o DOS é incorrecto... ao pedir-lhe para ordenar por DATA..... Mas considero grave o facto de o "run" depender da versão do DOS (3.21).... Ainda tentei substituir o MS-DOS, IO e COMMAND mas foi em vão... ugeria mais duas ou três coisas:
1. Não faça depender da impressão a correcção de dados...
2. O mesmo a respeito do fecho de contas. Por quê ter que imprimir?...
3. Faz muita falta uma tecla de função que permita... voltar ao menu principal, sem ter de estar a preencher tudo.

Olindo Pinto Marques
Monte da Caparica

Tenho de reconhecer que a falta foi minha. O seu problema é o mesmo que o dos outros leitores que também se queixam de que o computador entra em quebra quando se faz a ordenação por data.
Não se trata de qualquer incompatibilidade com a versão do DOS instalada. Como já tive ocasião de explicar no n.º 9 da **Spooler, a ordenação das datas é feita através de um "shell" ao DOS para o comando SORT.EXE.** Se o leitor não tem disco rígido, terá de copiar este comando (que se encontra na disquete "Sistema" do seu computador) para a disquete onde instalou o BANCO. Se tem disco rígido, basta modificar ou acrescentar o *path* do AUTOEXEC.BAT, de forma a que a máquina saiba onde procurar o comando SORT.EXE. Admitindo que os seus ficheiros do DOS estão num directório chanado SYSTEM, o *path* deverá incluir: \system.
A falta foi minha, pois deveria ter dado esta explicação no artigo em que apresentei o BANCO. Mas, se virmos bem as coisas, esta "falta" também teve os seus lados positivos, pois contribuiu para um grande diálogo com os estimados leitores e proporcionou-lhes certos conhecimentos que, "se a papinha estivesse toda feita", possivelmente não teriam oportunidade de obter.
Passemos à segunda questão. Na verdade, a solução encontrada foi de compromisso. Como o leitor deve ter reparado, o ecrã dos extractos está completamente cheio, não tendo espaço para qualquer outra coluna. Tive de escolher: ou diminuía o comprimento dos campos "referência", "designação" ou "valores" – ou não incluía o número do registo. Optei por este última solução, pois, sem impressão dos extractos, então sim, o programa seria "um aborto". Mas nada impede que o leitor modifique a dimensão dos campos do ficheiro BANCO.FIC. Na disquete deste número vai a fonte em Basic do programa BANCO (que não pudemos incluir na n.º 8 por absoluta falta de espaço) para que o leitor faça as alterações que entender.
Quanto ao fecho de contas: na verdade não depende da impressão – o leitor pode confirmar o seu desejo de efectuar o fecho de contas sem, previamente, ter impresso o extrato total ou do mês de Dezembro. A mensagem que aparece no ecrã é um conselho – não uma ordem! Só que, se não tem impressora ou não tomou nota, "à mão", dos valores dos extractos, tudo se perderá pois, como é habitual neste tipo de programas, o saldo é transportado para o ano seguinte e o ficheiro é limpo para poder começar a receber os dados do novo ano.
Quanto à tecla de função para voltar ao menu principal, sinceramente não percebo o seu problema. *Sem ter de estar a preencher tudo...*?! Preencher o quê, a ficha de lançamentos? Mas quando se pede "2 – Lançamentos a D/C" é precisamente para preencher toda a ficha. Não há mais nada a preencher, salvo os "Dados Iniciais", mas estes só se preenchem uma vez...quando se abre uma conta. Se efectivamente a sua observação tiver razão de ser, volte a escrever-nos.

...Possuo um Schneider PC/AT ...MS-DOS 3.30A (V1.1), cuja linguagem basic é a GW-BASIC 3.22 ... Como a linguagem BASIC disponível no mercado.... é incompatível com a utilizada pelo computador referido...solicito... se dispõem de material de apoio à referida linguagem...

Victor Manuel dos Santos Coelho
Mem Martins

Em princípio não conhecemos incompatibilidades entre as sucessivas versões do GW-BASIC da Microsoft. O que pode estar a acontecer é um problema de gravação dos ficheiros .BAS. Como o leitor sabe, podemos gravar os programas basic sob diferentes formatos, a saber:
SAVE «programa», a: grava o programa em caracteres ASCII, isto é, pode ser aberto e lido em qualquer processador de texto.
SAVE «programa», b: grava o programa em formato binário codificado, pelo que não pode ser lido ou editado.
SAVE «programa»: grava o programa no formato normal binário comprimido, que só pode ser lido e editado pelo editor do GW-BASIC.
Sugerimos que o leitor faça uma gravação em ASCII e tente lê-la no seu computador. Quanto ao material de apoio, sugerimos que consulte a lista de livros da disquete n.º 9 da **Spooler.**

Estou tentado a fazer uma assinatura conjunta da vossa revista com a CATS-BBS... Gostaria também que me esclarecessem se, em caso de assinatura, garantem que a disquete que acompanha a revista chegue nas devidas condições.

Artur Jorge de Oliveira Marçal
Torres Novas

O n.º 4 da **Spooler** ainda não está esgotado, mas quase... se o leitor quiser podemos enviá-lo à cobrança. Mas a nosa sugestão é que compre o último número (o 9) que tem um artigo na pág. 6 sobre a CAT-BBS. Lembramos, no entanto, que a **Spooler** já lançou a sua própria BBS – referimo-nos ao assunto neste número.

...Pode utilizar-se um monitor VGA para fazer correr programas destinados a monitores de menor resolução (EGA, MDA, Hercules, etc.)?... Gostaria que voltassem a incluir na Montra Spooler discos rígidos... os descontos a estudantes em alguns artigos da Montra Spooler de Novembro foi uma óptima ideia. Por que não voltar a fazê-lo?... É possível utilizar disquetes de 5 1/4 e 360K numa drive de 5 1/4 de 1,2 MB? Se sim como é feito? ...Existe diferença entre disco duro, disco rígido e disco fixo?

João Paulo da Cunha Tomás
Lisboa

Com efeito pode correr programas preparados para cartas de menor resolução, num monitor com carta VGA. O inverso é que nem sempre é possível.
A **Spooler** deixou de incluir discos rígidos na sua montra, pois, normalmente, um comprador destes equipamentos não o faz por esta via. No entanto, se quiser, podemos enviá-lo à cobrança, desde que nos dê a referência e características respectivas. Sinceramente, não o aconselhamos a entrar neste circuito, a menos que tenha os necessários conhecimentos para a montagem.
Os descontos para estudantes eram uma promoção dos representantes dos equipamentos. Não está na nossa mão essa iniciativa.
Com efeito, podem ler-se disquetes de baixa densidade numa drive de alta densidade, isto é, as de 360K podem ser lidas numa drive de 1,2Mb e as de 730K numa de 1,4Mb. O inverso é que não é assim tão linear. Vejamos: se tiver uma drive de 1,2Mb e se quiser mandar uma disquete de 5 1/4 a um amigo, terá de utilizar uma disquete de baixa densidade, formatá-la a 360K e poderá escrever nela o que quiser. Não tente formatar uma disquete de alta densidade a 360K e lê-la numa drive de baixa densidade quase de certeza terá problemas.
É interessante a questão que coloca sobre as muitas designações que normalmente vemos serem dadas aos discos rígidos (como nós, aqui na **Spooler**, lhes chamamos). Para começar, mesmo em inglês encontramos duas designações: *hard disk* e *fix disk*. Ambas se referem a discos rígidos (em oposição a *flopy disk* – disco flexível ou disquete) mas, enquanto que a primeira só significa disco rígido, a segunda diz-nos que esse disco é "fixo", isto é, que não é amovível

como as disquetes. Dirá o leitor – "mas isso não é novidade nenhuma, os discos rígidos não são para se tirarem e porem! Acontece é que existem discos rígidos amovíveis que se podem retirar e guardar num cofre para protecção de dados, por exemplo. Portanto, serão rígidos e não fixos. Penso que o leitor terá ficado esclarecido.

...possuo uma placa CGA. Gostava de saber se existe algum programa para emular a EGA na CGA...

Heitor Filipe Alhinha Coelho
Braga

Infelizmente, como regra, não se pode emular uma carta de resolução mais baixa para correr programas feitos para trabalharem com placas de maior resolução. O inverso já é possível, tal como foi respondido ao nosso leitor anterior.

...comecei a programar em Quick Basic V 4.5, mas surgiram uns problemas... por exemplo, num campo numérico introduzindo-se uma letra ou uma vírgula, em vez de dar "redo from start", queria que desse uma mensagem de erro ou que voltasse ao início... limitar o campo do cursor... como fazer para ele ter um campo demarcado e teclando ENTER mude para o campo seguinte...

José Manuel Pires
Riba-de-Ave

Em relação ao assunto das mensagens de erro no Basic, poderá fazer uma rotina que detecte os possíveis erros do seu programa e actue de acordo com eles. A instrução **ON ERROR GOTO** (número da 1.ª linha da rotina de detecção de erros), colocada no início do seu programa, provoca um salto para a rotina de detecção de erros sempre que um erro é detectado pelo Basic. A rotina de detecção pode utilizar as variáveis **ERR** (código do erro) e **ERL** (nº de linha onde ocorreu o erro) para proceder às necessárias correcções. Pode também usar a instrução **RESUME (0)** – continua a execução do programa a partir da linha que causou o erro; **RESUME NEXT** – continua a execução do programa a partir da instrução seguinte à que causou o erro ou **RESUME [nº de linha]** – neste caso a execução do programa far-se-á a partir do n.º de linha especificado. Também tem a possibilidade de definir ou simular os seus próprios erros através da instrução **ERROR n**. Em relação a isto aconselhamo-o a ler o respectivo manual do Basic.
Poder limitar o campo do cursor em **INPUT**, não é possível, tanto quanto sei. Pode, no entanto, construir a sua própria rotina de *input* utilizando a função **INKEY$**. Dou a seguir um pequeno exemplo de como poderá fazer isso. Supondo que quer escrever na linha 12, começando no valor de coluna 10, terminando em 50 e depois arquivar a *string* assim obtida na variável **NOME$**, deve chamar a rotina que começa na linha 60 da maneira que exemplifico na linha 10. Teclando **ENTER** a rotina passa para a linha 20 e assim por diante. O exemplo que forneço também possibilita utilizar a tecla **BACKSPACE** para apagar um carácter atrás da posição do cursor. O leitor poderá melhorar a rotina incluindo, por exemplo, as teclas cursoras direita-esquerda, **HOME** e **END**, etc., etc.. A rotina chama-se **INPUT.BAS** e encontra-se no directório **INPUT** da disquete **Spooler** deste número. (Franco Gomes foi o autor desta resposta e da rotina).

*...como ainda não possuo um computador, gostaria que me aconselhasse algum(uns) de boa qualidade... de um razoável preço... gostaria de aceder aos serviços da CATS-BBS... existe alguma diferença entre discos duros, discos rígidos ou discos fixos... será que posso trocar as disquetes de 5 1/4 por 3 1/2... porque na **Spooler** 9 puseram o mesmo artigo sobre a CATS-BBS que na número 8... números esgotados...*

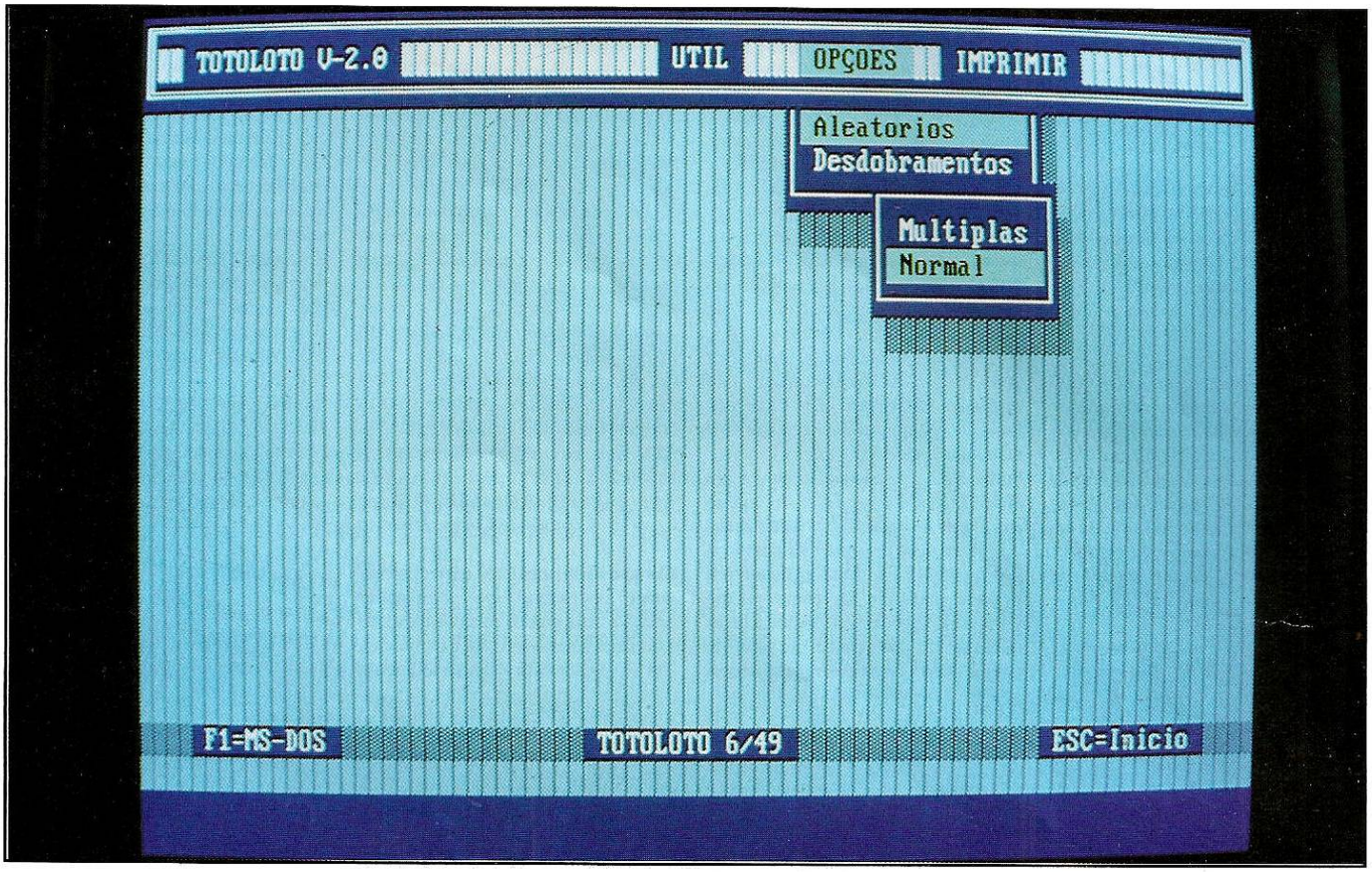
António Gualdino F. Jesus
Câmara de Lobos, Madeira

Como o estimado leitor deve compreender, aconselhar um computador envolve uma grande responsabilidade e também questões de ética comercial. Tal actividade está totalmente fora do âmbito da nossa revista. No entanto, conforme já anunciámos na **Spooler** n.º 9, vamos publicar brevemente uma separata com informações sobre diferentes marcas e modelos de computadores, dentro dum determinado limite de preço. Isto ajudá-lo-á a resolver o seu problema.
Quanto à CATS-BBS, tem duas hipóteses: ou adquire a **Spooler** n.º 4 (se a não tem), ou escreve para lá — Rua Carlos José Barreiro, 6 c/v, 1000 LISBOA — Telf. 524027.
A resposta sobre as diferentes designações dadas aos discos rígidos está dada mais acima.
Claro que pode trocar o tipo da disquete que recebe com a sua assinatura da Spooler. Basta que nos escreva nesse sentido.
O "artigo" sobre a CATS-BBS não é propriamente um artigo, mas publicidade redigida pela própria organização, pelo que a **Spooler** não tem qualquer responsabilidade sobre o seu conteúdo.
Infelizmente o n.º 3 também já está esgotado. Ainda não fizemos a reedição dos números esgotados, mas contamos fazê-lo em breve — os nossos leitores serão avisados atempadamente.

Renato Casquilho

COLABORAÇÃO DO LEITOR

TOTOLOTO



O nosso leitor Carlos Jacinto enviou-nos o seu programa TOTOLOTO V-2.0 que, como o seu nome indica, se destina a auxiliar os apostadores na sempre ingrata tarefa que é a escolha dos números que "irão" sair na "roleta". O programa, pela sua qualidade, mereceu ser publicado, para benefício da comunidade Spooler. **Passamos a reproduzir o texto que acompanhava:**

Sendo comprador da Revista **Spooler** desde o primeiro número, quero, antes de mais, dar os parabéns por esta revista, cujos conteúdo e apresentação são de louvar. Desta forma contribuo para ela com o **TOTO49**, um "Programa ou Jogo, conforme o critério" chamado **Totoloto V-2.0 "6/49"**, que tem como objectivo fazer desdobramentos, múltiplas, ou números aleatórios de 1 a 49, conforme a opção escolhida.

Este programa foi feito na linguagem TURBO BASIC V-1.0, que é uma linguagem Basic que não precisa de linhas para se programar. Tento desta forma aumentar os conhecimentos dos programadores interessados e, com a fonte de origem e algumas explicações que irei dar, espero que fiquem a ter um conhecimento bastante razoável do programa.

Para começar temos vários menus: **UTIL, OPÇÕES, IMPRIMIR.**

1.º MENU – UTIL – "UTILITÁRIOS DO TOTOLOTO"

Neste menu pode-se Chamar números anteriormente gravados, Gravar apostas (depois de se ter ido ao segundo menu), Rever, fazer o Shell, ou voltar ao MS-DOS.

2.º MENU – OPÇÕES

Dá-nos a escolher as várias opções à nossa disposição: Apostas Aleatórias do Computador, Aleatórias com Múltiplas, Desdobramento de Números introduzidos "com opção de múltiplas...". Depois de os números serem vistos no monitor, pode tornar a ver-se os mesmos números bastando ir ao 1.º Menu e escolher a opção REVER.

Se pretender imprimir basta ir ao 3.º Menu.

3.º MENU – IMPRIMIR

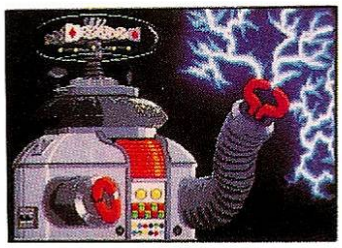
Esta opção dá-nos a hipótese de imprimir as apostas requeridas para a impressora, bastando que, para isso, se tenha ido primeiro ao 2.º Menu, ou ido ao 1.º Menu, como já referi.

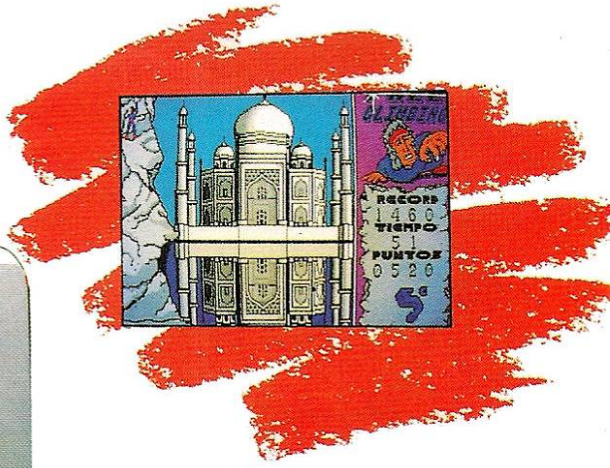
Boas apostas e grandes prémios. Boa sorte!

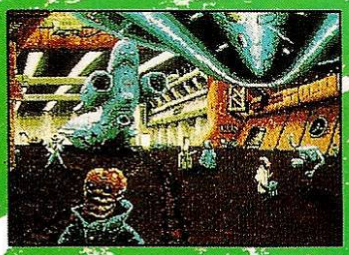
Carlos Jacinto
Mirandela

A MAIOR AVENTURA DO SÉCULO

ATARI 1040 STE







PROGRAMAS INCUÍDOS:

- **1ST Word** (Tratamento de texto)
- **NeoChrome** (Programa de gráficos)
- **ST Basic** (Linguagem de programação)
- **1 Jogo à escolha** (mais de 450)

REPRESENTANTE LOCAL

ATARI®

Só CARTÃO JOVEM
ou **MELO CARTÃO**
ENTRADA: 35.000$00
+12 CH. 15.750$00

MELO INFORMÁTICA

LOJA MELO: Rua Gonçalves Crespo, 18-C – 1100 LISBOA (junto ao Arquivo de Identificação)
Tels.: 54 99 04/52 56 69 FAX: 52 46 37

UM CONFORTÁVEL ACENTO

> **O programa** Acentos **encarrega-se de acentuar os seus textos, através de uma substituição de caracteres**

Quantas vezes não ansiou já por uma forma prática e rápida de pôr acentos nos seus textos? Pois bem, através do programa aqui apresentado, isso torna-se extremamente fácil. Não terá, pois, de voltar a decorar os códigos ASCII dos caracteres, nem de fazer malabarismos de dedos na tecla Alt...

O programa **Acentos** encarrega-se de acentuar os seus textos, através de uma substituição de caracteres. Assim, a sequência de caracteres ^e é substituída pelo carácter ê. O mesmo se passa com os outros acentos e com a cedilha.

Na verdade, o programa é configurável e o utilizador pode modificar os caracteres que representam os acentos. De qualquer das formas, existe uma configuração por defeito que assume, por omissão, os caracteres :

~ para o til;
‘ para o acento agudo;
^ para o acento circunflexo;
‘ para o acento grave;
, para a cedilha.

Por vezes, temos interesse em que não haja sempre substituição (é o meu caso ao escrever este artigo...). Deveremos então preceder o acento do carácter \ Se quisermos que no nosso texto figure o carácter \, deveremos então escrever dois caracteres \ seguidos.

Por forma a tornar bem clara a utilização deste programa vou, então, apresentar um pequeno exemplo. Se tivermos o texto inicial:

{‘E dif’icil arranjar um exemplo com muitas palavras acentuadas? ,c \,c,d’e `a^i\\ Nem por isso ...

o **Acentos** encarregar-se-á de o passar para:
É difícil arranjar um exemplo com muitas palavras acentuadas? ç ,c,dé àÔ\ Nem por isso ...

UMA MARCA INICIAL

Tem, por vezes, utilidade que o **Acentos** só comece a actuar a partir de um determinado ponto no texto. Para obviar esta necessidade, o **Acentos** reconhece a marca {, Ou seja, o **Acentos** salta todo o texto inicial, sem o reproduzir de forma alguma, até atingir o sinal {.

CONFIGURAÇÃO DO ACENTOS

No caso de preferir utilizar outros caracteres para definir os acentos, não há problema. Bastará que, ao invocar o programa, o faça com, no máximo, cinco argumentos – um por cada sinal de acentuação. O programa deverá ser, pois, ser invocado da seguinte forma:

ACENTOS til agudo circunflexo grave cedilha

Vejamos alguns exemplos:

ACENTOS # @

Neste caso, o carácter # passa a ser interpretado como o til, e o @ como o carácter agudo. Quanto aos restantes acentos, não é alterada a configuração por defeito.

ACENTOS ! @ # $ %

Desta forma, o carácter ! é interpretado como til, o @ como sendo o acento agudo, o # como o acento circunflexo, o $ como o acento grave, e o % como sendo a cedilha.

ACENTOS

É aceite a configuração por defeito, já referida acima.

UMA PEQUENA AJUDA

No caso de necessitar utilizar este precioso utilitário, e já se ter esquecido de como fazê-lo, não desespere. Dirija-se à tabacaria mais próxima e compre um novo exemplar desta magnífica revista. No caso de isso ser completamente impossível, poderá então recorrer a um pequeno help instalado no **Acentos**. Para invocá-lo não tem mais que chamar **Acentos** com o argumento h:

ACENTOS h

COMO UTILIZAR O ACENTOS

O programa funciona como a maioria dos pequenos utilitários do sistema UNIX. Ou seja, lê do *standard input* e escreve no *standard output*. Para passar um ficheiro inteiro pelo **Acentos** deverá, pois, fazer:

ACENTOS < ficIn > ficOut

Os sinais < e > são interpretados pelo Command.com, o interpretador de comandos que naturalmente mais utilizará, como sendo, respectivamente, a redirecção do *standard input* e do *standard output*. Assim, o que o comando acima faz não é mais que dar o ficheiro ficIn a ler ao **Acentos**, e escrever o resultado no ficheiro ficOut.

Imaginemos, pois, que tínhamos acabado de escrever o nosso ficheiro de texto "artigo", e que queríamos produzir o ficheiro devidamente acentuado "final". Contudo, queríamos que o **Acentos** interpretasse como sendo o til, não o carácter ~, mas sim o !. Não teríamos mais que digitar o comando:

ACENTOS ! < artigo > final

Tenha sempre o cuidado de não dar como *standard output* um ficheiro que lhe seja precioso. Caso contrário, estará irremediavelmente perdido...

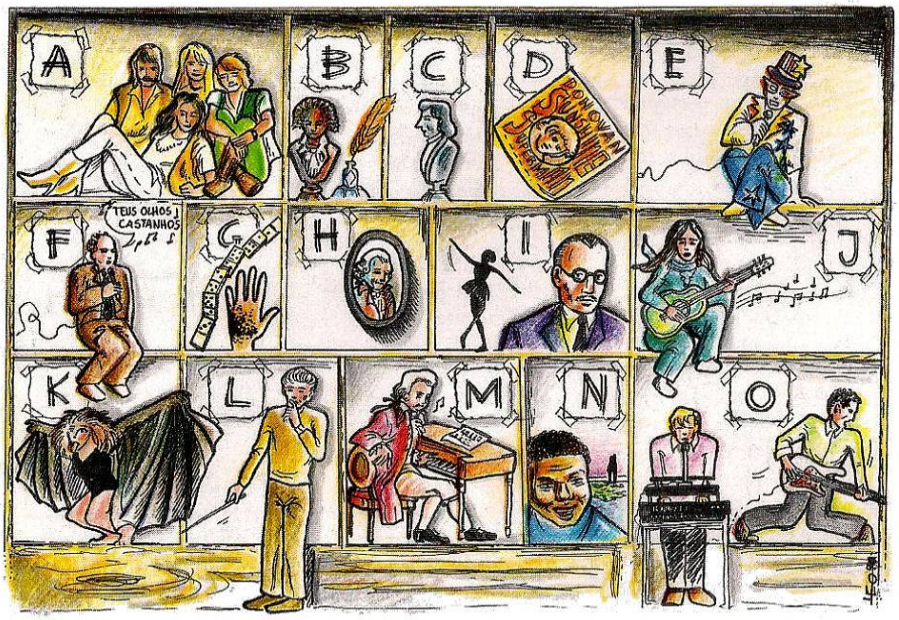
NOTAS DE IMPLEMENTAÇÃO

O código do programa – feito em C – está, como já é habitual, acessível na disquete distribuída juntamente com a revista, bem como o programa executável.

A implementação é absolutamente banal. Utiliza-se uma tabela pré-preenchida com os valores por defeito, e que pode ser alterada mediante a leitura do argumento passado ao main. É a função *translate* que faz todo o trabalho, recorrendo a *ungetc* no caso de ter avançado excessivamente. Isto acontece no caso de aparecer um sinal sem que, contudo, se lhe siga uma letra que possa ser acentuada com esse mesmo sinal.

João Cardoso

MUSITECA



HOMENAGEM AOS COLECCIONADORES

“...coleccionar (a sério) não deve limitar-se ao «juntar», ao «acumular», ao «amontoar» de coisas, objectos ou assuntos”

Um dos passatempos que mais admiro pela potencialidade de evasão e de distracção que contém, é o de ''coleccionar''. Coleccionar, dos selos aos automóveis antigos, da música às moedas, das caixas de fósforos aos relógios, dos livros aos diapositivos. Tudo se pode coleccionar e tudo se colecciona. Não tenho dúvidas que, por esse mundo, há ''maduros'' capazes de coleccionar coisas das mais incríveis que não passam pela cabeça ao comum dos mortais.

Também creio que, em maior ou menor grau, todos temos uma certa ''veia'' de coleccionador. Que o digam os entusiastas da informática, que juntam ''montes'' de disquetes com programas para todos os usos e fins; o tio daquele meu amigo que guarda todos os frascos e frasquinhos; a vizinha do meu primo que, desde pequenina, acumula caixas e caixinhas de todos os tamanhos e feitios...

Todos me merecem o maior respeito e admiração. Mas a minha atenção recai – até porque me incluo nas suas ''hostes'' — naqueles coleccionadores mais convictos, que levam a sério o seu passatempo e que, por isso, inevitavelmente, devem ter problemas. Porque coleccionar (a sério) não deve limitar-se ao ''juntar'', ao ''acumular'', ao ''amontoar'' de coisas, objectos ou assuntos.

Receio, para nosso mal (ou bem?), estar a atirar pedras aos telhados dos vizinhos, quando o meu é de vidro. Admitamos que é para nosso bem e continuemos a teorizar. Mas, afinal, o que entendo eu por ''coleccionar''?

Em primeiro lugar trata-se de reunir um número suficiente de ''artigos'' da mesma natureza, que justifique o nome de ''colecção''. Em seguida temos o mais difícil (se bem que o mais interessante) : ''classificar'' e ordenar os artigos. Depois há que ''arrumá-los''. Finalmente o mais gostoso: ver, admirar, contemplar, exibir a nossa colecção. Todavia, para que este último estádio resulte, as etapas anteriores têm de ser executadas correcta e metodicamente. Já imaginou, o leitor que é um fervoroso filatelista, o que seria ter todos os seus valiosos selos amontoados num único grande envelope, os de Portugal misturados com os da China? E, quando lhe perguntassem se possuía o famoso selo da série tal, do ano tal, do país tal, nem sequer soubesse se o tinha (e até tinha!) e onde? Que vergonha!

E o leitor que é um dedicado melómano e que foi juntando discos, ou gravações em cassete ou bobina — algumas verdadeiras peças de museu, raras e preciosas, ainda gravadas em 78 rotações — e não soubesse se tinha a obra tal, do compositor tal e onde? Que lástima! Que desperdício!

Podia acrescentar mais exemplos lúgubres. Mas chega de impressionar os sãos de espírito.

Vamos analisar uma situação concreta. Por sinal, aquela que melhor conheço e que está na origem deste artigo e deste programa. A música. A bem-dita música que nos ajuda a passar as boas e as más horas da vida. Pois bem, se realmente somos verdadeiros amadores e coleccionadores de música (melómanos, como diriam os bem-falantes), certamente teríamos constituído o clássico ficheiro organizado por compositores e por ordem alfabética dos seus apelidos, composto por uma série daquelas fichas de cartolina que se vendem em qualquer papelaria. E à medida que aumentássemos a nossa colecção, iríamos acrescentando novas obras à ficha do compositor em causa, ou novas fichas, quando a última já estivesse completamente preenchida. Até aqui tudo bem.

Mas admitamos que o leitor tinha uma predilecção especial pela música de **Mozart**. Para ilustrar devidamente a situação com que se pode deparar, vou correr o risco de ser fastidioso para alguns dos nossos leitores mais ocupados ou menos admiradores da música chamada de ''clássica''. Vamos para a frente! Acontece que **Wolfgang Amadeus Mozart** (1756-1791) compôs (salvo erro ou omissão):

19	Missas	36	Peças Litúrgicas
18	Sonatas de Igreja	12	Peças Maçónicas
26	Óperas	72	Árias, Melodias e "Lieder"
34	Dansas	23	Cassações e Divertimentos
13	Serenatas	71	Sinfonias e Aberturas
13	Concertos para Violino	24	Concertos para Piano
13	Concertos para instrumentos de sopro	9	Quintetos de cordas
33	Quartetos	13	Trios
61	Duos	35	Sonatas e Fantasias para Piano
15	Variações para Piano	19	Peças breves para Piano
4	Peças para Orgão	23	Trabalhos sobre obras de outros Compositores

Sinceramente, não estou a ver o leitor (nem eu próprio) a constituir subficheiros para cada uma destas 23 classificações das obras de **Mozart.** Sendo assim, as **586** obras desta lista ocupariam cerca de **146** fichas (admitindo que cada ficha levaria 4 obras, contemplando o nome, o seu número de catálogo, o número de ordem dentro da classificação, o tom, a interpretação e o código de arquivo ou de arrumação). E era nessas **156** fichas que o leitor teria de procurar a obra desejada — o que não seria tarefa nada fácil. Vejamos um exemplo:

O leitor queria saber se tinha e onde guardava, digamos, o*Trio nr.7 em Mi bemol maior K498 para clarinete, piano e viola.* Como as entradas nas fichas vão sendo feitas à medida que se adquirem as obras, o mais provável é que a obra procurada estivesse perdida pelo meio das sinfonias ou das sonatas e, muito possivelmente, lá para o fim do ficheiro.

E **Franz Joseph Haydn** (1732-1809), com as suas 104 Sinfonias, 14 Concertos para piano e orquestra, 5 Concertos para violino, 83 Quartetos, 97 Trios, etc.? E **Ludwig van Beethoven** (1770-1827), com as suas 9 Sinfonias, 6 Aberturas, 5 Concertos para piano e orquestra, 1 Concerto para violino, mais de 130 peças de Música de Câmara, etc.? E **Johann Sebastian Bach** (1685-1750), autor de 200 Cantatas, 260 Corais e inúmeros Concertos, Sonatas, Sinfonias, Partitas, Árias, Tocatas, Fugas, Suites, etc.?

Todos estes exemplos se destinam a ilustrar as dificuldades que se deparam a um coleccionador quando o ''objecto'' da sua colecção atinge proporções inesperadas, que se traduzem, inevitavelmente, por dois grandes inconvenientes: o primeiro está na dificuldade de pesquisa e de localização; o segundo é a manifesta desorganização do ficheiro. São estes inconvenientes que me proponho eliminar com o auxílio do nosso fiel amigo computador e do programa MUSITECA.

Antes de prosseguir, devo esclarecer que os exemplos escolhidos sobre a chamada ''música clássica'', não pressupõem qualquer menosprezo pela outra música, também conhecida por ''ligeira''. Muito pelo contrário, o que me interessa na música é sobretudo a qualidade e esta pode existir (ou não) em todos os géneros de música. Aliás, a minha ''musiteca'' encontra-se igualmente beneficiada por muitas obras ligeiras, nomeadamente da chamada música popular — canções, musiquinhas, bons fados, etc.

O PROGRAMA MUSITECA

Talvez os filólogos se arrepiem com o termo que inventei para nomear o programa mas, felizmente, a informática admite certas ''liberdades'' de linguagem. Passemos adiante.

Como o nome sugere, destina-se a gerir uma ''biblioteca'' de música. Foi construído em dBase III Plus, pelo que só funcionará nesse ambiente. Desta vez fiz umas ''habilidades'' para tornar o programa mais rápido e atraente: tem duas opções: para monitores a cores ou monocromáticos e os ecrãs do menu principal e das opções de impressão são ficheiros .COM chamados directa e automaticamente pelo comando !RUN do dBase. Veja os programas MUSITECA.SRC e MUSPRINT.SRC para se aperceber do seu funcionamento. E, já agora, chame os programas MUSICACR.COM e IMPRIMCR.COM (se o seu monitor for a cores) ou MUSICAPB.COM e IMPRIMPB.COM (se o seu monitor for monocromático) directamente a partir do DOS e verá como os ecrãs aparecem instantaneamente — sem funcionarem, claro!

Figura 1 – Aspecto do ecrã inicial de Musiteca.

CONSTITUIÇÃO DO PROGRAMA

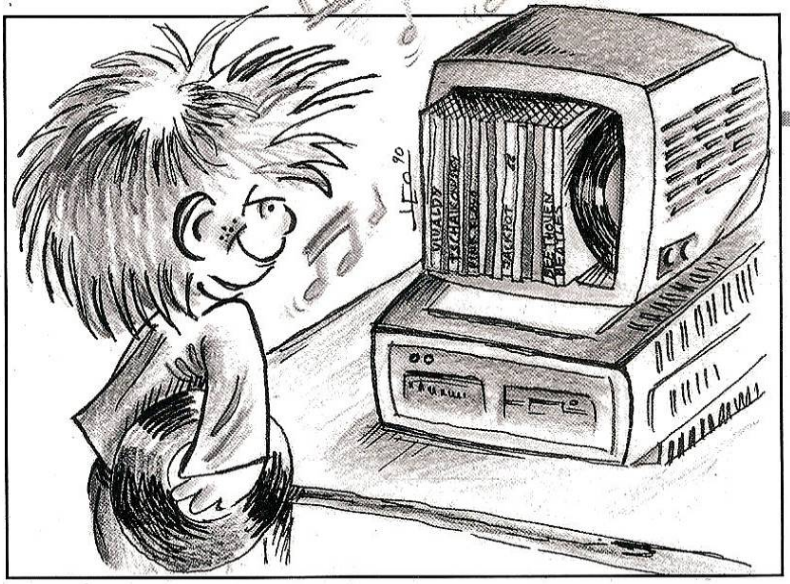
O programa é constituído pelos seguintes ficheiros:

MUSICA.PRG	=	O programa ''pseudocompilado'' e ''linkado''
MUSITECA.SRC	=	fonte do menu principal
MUSICACR.COM	=	ecrã do menu principal para cores
MUSICAPB.COM	=	ecrã do menu principal para preto e branco
MUSENTRA.SRC	=	fonte da entrada de dados
MUSCOMP.SRC	=	fonte da pesquisa por compositores
MUSOBRA.SRC	=	fonte da pesquisa por obras
MUSOCOMP.SRC	=	fonte da pesquisa por obras/compositores
MUSCOD.SRC	=	fonte da pesquisa pelo código
MUSALTER.SRC	=	fonte de corrigir registos
MUSAPAGA.SRC	=	fonte de eliminar registos
MUSPRINT.SRC	=	fonte das opções de impressão
MUSPRN1.SRC	=	fonte da impressão por compositor
MUSPRN2.SRC	=	fonte da impressão por obra
MUSPRN3.SRC	=	fonte da impressão por código
IMPRIMCR.COM	=	ecrã das opções de impressão para cores
IMPRIMPB.COM	=	ecrã das opções de impressão para preto e branco
MUSICA.DBF	=	ficheiro de dados
MUSCOMP.NDX	=	tabela de indexação por compositor + obra
MUSOBRA.NDX	=	tabela de indexação por obra
MUSCASS.NDX	=	tabela de indexação por código

Nota: Os ficheiros impressos a **cheio** são os indispensáveis para o funcionamento do programa. Os restantes são as fontes em dBase III Plus para o leitor analisar no editor do dBase ou noutro processador que admita texto ASCII.

Falei em ''pseudocompilado'' e ''linkado''. Com efeito, o programa não se encontra verdadeiramente compilado, pelo que não corre directamente a partir do DOS — precisa sempre do interpretador do dBase. No entanto, esta ''pseudo compilação'' e a subsequente

“Talvez os filólogos se arrepiem com o termo que inventei para nomear o programa mas, felizmente, a informática admite certas «liberdades» de linguagem”



"linkagem" possuem duas grandes vantagens:

— os programas .PRG ficam com cerca de 50% do comprimento original e correm com maior rapidez;

— a "linkagem" (união) "une" todos os .PRG num único programa com a extensão .PRG.

Estas operações são feitas recorrendo aos dois utilitários DBC.COM e DBL.COM que acompanham a *package* do dBase. A extensão .SRC é para ser reconhecida pelo DBC.COM e significa *source* — fonte.

INSTALAÇÃO DO PROGRAMA

> **"O programa possui um dispositivo de detecção automática do tipo de monitor instalado"**

O programa encontra-se no directório MUSICA da disquete **Spooler 10**, compactado com o nome de MUSICA.EXE, pelo que, em primeiro lugar, tem de ser descompactado para o disco rígido ou para uma disquete. Vejamos as duas alternativas:

1. Instalação em disco rígido

Se o leitor leu, apreciou e instalou o programa CATALOGO do nr.º 9 da **Spooler**, já sabe que tem de criar um subdirectório chamado **MUSICA** no seu directório DBASE. Comece por copiar para esse subdirectório o programa MUSICA.EXE; depois coloque-se no subdirectório MUSICA e execute o programa digitando MUSICA: o ficheiro é descompactado. Faça DIR e verá todos os ficheiros que constituem o programa. Não se esqueça de apagar o MUSICA.EXE para poupar espaço no disco.

Ainda dentro do subdirectório MUSICA, copie os quatro ficheiros MUSICACR.COM, IMPRIMCR.COM, MUSICAPB.COM e IMPRIMPB.COM para o directório DBASE: **esta operação é indispensável para que o programa funcione correctamente** a partir do CATALOGO; depois pode apagar esses ficheiros .COM do subdirectório MUSICA, pois já aí não são necessários e poupa espaço no disco.

A última etapa consiste em acrescentar o novo programa ao CATALOGO. Para o efeito, coloque-se no directório DBASE, chame o dBase e quando aparecer o CATALOGO escolha a opção "Aumentar Programas" do programa CATSET; depois é só preencher o registo com o número do novo programa (que é o número do registo), com o nome (que é MUSICA) e a data (que será a data em que o leitor fizer a instalação). Regresse ao CATALOGO premindo "0", o que lhe facultará a escolha do programa que vai ver – claro que será **MUSICA!**

Tudo deverá funcionar impecavelmente desde que o leitor atento tenha seguido as instruções do último número da **Spooler** para a instalação do CATALOGO e, agora, estas últimas indicações.

2. Instalação em disquete

Vamos partir do princípio que o leitor utiliza disquetes de 3,5" de baixa densidade (730K). Normalmente terá uma disquete de arranque do dBase e uma segunda disquete com os ficheiros DBASE.OVL e COMMAND.COM. Na primeira deverá instalar o ficheiro CONFIG.DB, cuja última linha deverá ser:

command = do musica

Na segunda disquete instalará o programa MUSICA.EXE a fim de ser descompactado e dar origem aos diferentes programas que o constituem. Para isso, copie esse ficheiro que se encontra no directório MUSICA da disquete **Spooler 10** para a sua disquete n.º 2 e digite MUSICA – a descompactação efectuar-se-á automaticamente. Em seguida deve apagar o ficheiro MUSICA.EXE que passou a ser desnecessário, bem como todos os ficheiros "fonte" com a terminação .SRC (depois de os ter copiado para uma outra disquete, a fim de poder analisá-los quando entender).

Coloque a disquete dBase III Plus na drive e chame o programa digitando DBASE; quando lhe for pedida a introdução da disquete n.º 2, faça-o e prima ENTER. O programa MUSICA arrancará automaticamente.

FUNCIONAMENTO DO PROGRAMA

O programa possui um dispositivo de detecção automática do tipo de monitor instalado. Se o seu sistema for a cores, o programa chama o ecrã MUSICACR.COM; se for monocromático, chamará o ecrã MUSICAPB.COM. O mesmo se passará com os ecrãs das opções de impressão.

O arranque do programa dará origem ao ecrã representado na figura 1.

A primeira operação será, naturalmente, a **«Entrada de dados»**. Prima "1" e terá à sua disposição um ecrã especialmente concebido para o preenchimento do ficheiro MUSICA.DBF. No topo do ecrã aparece o último registo entrado — no caso presente estará em branco — e que é muito útil quando se interrompem as entradas e nos esquecemos do ponto onde tínhamos ficado. Mais abaixo, terá a zona destinada ao preenchimento dos registos.

O ficheiro MUSICA.DBF tem somente 3 campos:

Campo	**COMPOSITOR** com	15 caracteres
Campo	**OBRA** com	30 caracteres
Campo	**CODIGO** com	7 caracteres
	Total =	**53** caracteres

De notar que tive a preocupação de limitar ao mínimo o número e a dimensão dos campos pois, como é fácil de calcular, uma colecção de discos ou cassetes já com um certo vulto, facilmente dará origem a um volumoso ficheiro base e a três quase tão volumosas tabelas de indexação.

O leitor perguntará: "mas um ficheiro que se preze não deverá conter outras informações, tais como a interpretação (orquestra, maestro, coros, solistas, etc.), a data da gravação, etc.?" Sem dúvida que a resposta é afirmativa. Mas, se só com estes três campos o ficheiro pode tomar proporções inesperadas e incomportáveis, o que não seria se lhe acrescentássemos mais campos para prever todos os items? A solução (a minha solução) é colocar estes items no rótulo das cassetas ou das bobinas. Claro, como é óbvio, trata-se de uma opção pessoal, que os caros leitores podem reprovar. Por isso lhes enviamos as fontes a fim de poderem ser modificadas.

Ora vejamos o que uma colecção de 1000 obras daria (aproximadamente) em termos de ocupação de disco:

1.000 obras x 53 = 53 000 bytes
Tabela MUSCOMP.NDX = cerca de 65 500 bytes
Tabela MUSOBRA.NDX = creca de 47 000 bytes
Tabela MUSCASS.NDX = cerca de 19 000 bytes
Total de memória de massa ocupada = **cerca de 184 500 bytes**

NOTA: a tabela MUSCOMP.NDX terá uma dimensão superior à do ficheiro base, pois ordena por compositor e, dentro de cada compositor, por obra.

Após ter dado entrada a um certo número de obras, o leitor está em condições de ensaiar as restantes opções do menu principal. As opções 2, 3, 4 e 5 permitem a pesquisa, respectivamente, por COMPOSITOR, OBRA, OBRAS de um determinado COMPOSITOR e por CÓDIGO. Em qualquer das opções não é necessário escrever o nome completo, salvo em casos de COMPOSITORES ou OBRAS que tenham as primeiras letras em comum. Por exemplo, se na pesquisa por COMPOSITORES digitar «B», a listagem apresentará todas as OBRAS dos COMPOSITORES cujos nomes comecem por «B»:

BACH
BEETHOVEN
BELLINI
BERLIOZ
BIZET
BOMTEMPO
BORODINE
etc...

Se digitar «BE», serão apresentadas as obras de:

BEETHOVEN
BELLINI
BERLIOZ
etc...

Se quiser ver exclusivamente as obras de **Beethoven**, poderá digitar somente «BEE», pois não é provável ter outro compositor cujo nome comece por estas três letras.

No caso da pesquisa por OBRAS, verifica-se a mesma situação. Assim, se digitar «S», a listagem apresentará, pelo menos:

SONATA
SONATINA
SUITE

Aconselhamos o leitor a escolher judiciosamente as abreviaturas que usará para as suas obras (terá de abreviá-las, caso contrário, não lhe chegam os 30 caracteres), a fim de poder pedir selectivamente as obras desejadas.

As restantes opções de pesquisa obedecem ao mesmo critério.

— A opção **"Corrigir registos"** começa por pedir o número do registo a corrigir — esse número é retirado de qualquer uma das listagens de pesquisa. A correcção faz-se levando o cursor à palavra ou letra errada e digitando a necessária alteração.
— A opção **"Eliminar registos"**, depois de pedir o número do registo a eliminar, apresenta-o no ecrã e pede a sua confirmação. Premindo-se «1» o registo é eliminado. Premindo «2» volta-se ao ecrã principal.

NOTA: As opções "Entrada de dados", "Corrigir registos" e "Eliminar registos" são obrigatoriamente seguidas por um processamento das tabelas de indexação, cujo tempo de execução depende do número de registos do ficheiro MUSICA.DBF.

— A opção **"Opções de Impressão"** dá origem ao ecrã da Figura 2.

Qualquer das suas opções obedece aos mesmos condicionalismos das pesquisas para ecrã, isto é, há que ter em atenção o número de letras que se digita para um determinado compositor ou obra.

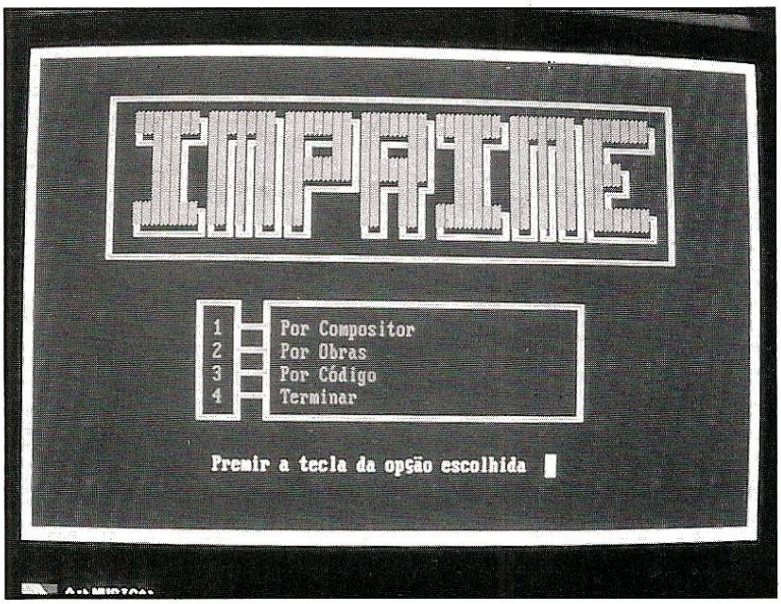


Figura 2 – O ecrã das «Opções de Impressão»

O programa está configurado para trabalhar com folhas soltas de papel A4 e o tipo de letra é reduzido de forma a imprimir 132 caracteres por linha. Contudo tem uma "habilidade" destinada a poupar papel. Quando a impressão atinge a linha 57, aparece uma mensagem no ecrã avisando para se alimentar a impressora com papel e voltar a colocá-la *on line*. Volte a meter a mesma folha e, depois de premir uma tecla qualquer; uma nova mensagem pede-lhe que indique a margem de impressao (7 ou 75); como já tem metade da folha impressa com a margem a «7», indique «75» para que a nova impressão comece nessa coluna. Terminadas as novas 57 linhas, volta a aparecer a primeira mensagem e o processo repete-se, mas desta vez introduzindo uma folha nova e indicando «7» para a margem.

Finalmente, a opção **"Terminar Programa"** começa por perguntar se foram feitas alterações, isto é, se o ficheiro MUSICA.DBF foi alterado por introdução de novos dados, por correcções ou por eliminação de registos — isto destina-se à execução de cópias de segurança do ficheiro base e das suas tabelas indexadas. Na afirmativa prima "S", bastando-lhe seguir as indicações que o programa vai dando; no final do *backup* regressa-se ao CATALOGO. Na negativa, o regresso ao CATALOGO é imediato.

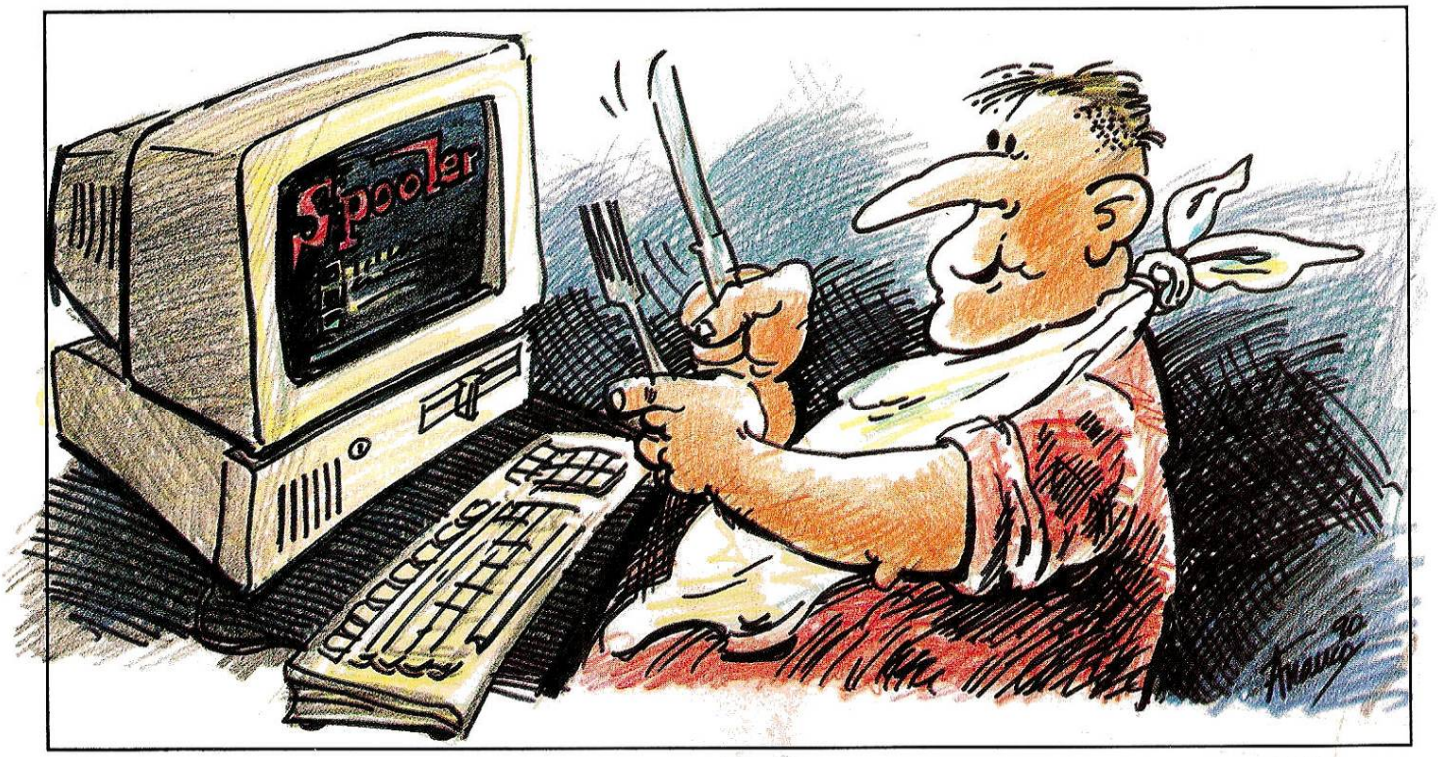
Resta-me acrecentar que este será o primeiro de uma série de utilitários dedicados aos coleccionadores. O próximo será — por que não? — para os "coleccionadores" de programas, jogos, utilitários, etc., pois penso que, tal como me acontece, já têm tanta coisa que nem se lembram do que têm e, sobretudo, onde têm. Depois será a vez dos filatelistas, numismatas, coleccionadores de *slides*, etc... Mas aqui preciso da colaboração dos leitores em geral e dos coleccionadores em particular. Primeiro para que me digam o que mais lhes interessa. Depois — e aqui apelo para os especialistas — para me dizerem, dentro de cada especialidade, quais os campos a contemplar no ficheiro, a sua dimensão, o que pesquisar, etc. Conto com a vossa colaboração.

A conversa já vai longa e o leitor deve estar ansioso por transformar o seu "antiquado" ficheiro manual num moderno e eficiente ficheiro "electrónico". Escusado será dizer que estamos à sua inteira disposição para esclarecer quaisquer dúvidas ou dificuldades. Os meus votos para uma gloriosa e bem organizada MUSITECA.

Renato Casquilho

"O programa está configurado para trabalhar com folhas soltas de papel A4 e o tipo de letra é reduzido de forma a imprimir 132 caracteres por linha"

O CARDÁPIO DA DISQUETE SPOOLER



> **Dedico este artigo a todos os leitores que ainda navegam, como eu, ou pretendem navegar nos primórdios do *Assembly*.**

Foi com uma certa relutância que acedi a escrever este artigo sobre o pequeno programa que, desde o número 9, apresenta ao leitor a lista de "pratos" que a disquete lhe propõe. Com relutância, porque não sou um especialista na matéria, apenas um curioso que gosta, como o leitor, destas máquinas infernais, que tantas vezes nos fazem ficar acordados pela noite dentro, a olhar fixamente para os pequenos monitores e a martelar raivosamente o teclado, esperando sucesso daquela rotina que teima em não resultar. Algumas vezes chegamos até a insultar a inocente máquina quando esta resolve resistir, deliberada e continuadamente, contra a profunda lógica dos nossos programas.

Na qualidade de responsável pelo grafismo da revista, sempre me desgostou o fraco aspecto que a disquete, em termos de imagem, apresentava e, como quem quer vai, quem não quer manda, resolvi fazer qualquer coisa nesse sentido. O *menu* da disquete do n.º 8 já saíu com alguma cor e som, gerados por um pequeno programa em GWBASIC compilado em QBASIC. Mas o inconveniente destes compiladores é sabido: os executáveis ficam mais rápidos, mas enormes. E o espaço da nossa disquete é demasiado precioso para alojar as manias de grandeza dos compiladores das linguagens de alto nível.

Foi por isso que (à semelhança do que acontece em algumas mesas redondas na TV) decidi baixar o nível da linguagem. Não! Não desatei aos palavrões: desatei, isso sim, a estudar a linguagem *assembly*. Peguei em algumas ferramentas (o "assemblador" A86 de Eric Isaacson, e no *Norton Guides* de Peter Norton) e lancei mãos à obra.

Dedico este artigo a todos os leitores que ainda navegam, como eu, ou pretendem navegar nos primórdios do *Assembly*. O que vai seguir-se não tem quaisquer pretensões didácticas, é somente o relato da minha experiência e "descobertas" pessoais sobre o tema e que tenho muito gosto em compartilhar com o leitor, seguindo o espírito que norteia esta revista. Estou, portanto, receptivo ao intercâmbio sobre *assembly* e, se a participação do leitor assim o justificar, prometo, desde já, uma secção regular onde se tratará do assunto.

Os meus conhecimentos de *assembly* remontam do tempo do código máquina para o *Spectrum* e mesmos esses já estavam bastante esquecidos. Mas com alguns serões de estudo, muito café, teimosia e centenas de "bloqueios" do processador, consegui juntar os conhecimentos básicos para começar a realizar o programa de apresentação do conteúdo da disquete que não ocupasse quase todo o espaço da mesma. Era necessário dominar minimamente duas áreas: a manipulação do ecrã e a produção de efeitos sonoros.

A IMAGEM EM MODO TEXTO

Em relação à imagem, as rotinas prefabricadas do DOS e BIOS não me atraíram; o que eu queria era uma única rotina que, sendo o mais versátil possível, permitisse imprimir qualquer coisa, em qualquer lugar, em qualquer cor e num número qualquer de vezes. Creio ter conse-

guido esse objectivo com a rotina **PRINTV** que o leitor pode encontrar (laboriosamente comentada) no ficheiro **SPOOL_A.ASM** na disquete. Essa rotina, que funciona exclusivamente em modo texto, baseia-se na cópia de dados directamente do nosso programa para o *buffer* do ecrã. Este *buffer* não é mais que uma determinada zona na memória que guarda os dados que ''refrescam'' a imagem do ecrã. Qualquer byte que seja aí ''pokado'' aparece de imediato na imagem. Por exemplo, se o leitor tiver a palavra **SPOOLER** em cor amarela sobre fundo preto escrita no canto superior esquerdo do seu monitor, a seguinte sequência de bytes encontrar-se-á a partir do endereço **0** do segmento do *buffer* do ecrã:

53 0E 50 0E 4F 0E 4F 0E 4C 0E 45 0E 52 0E ...

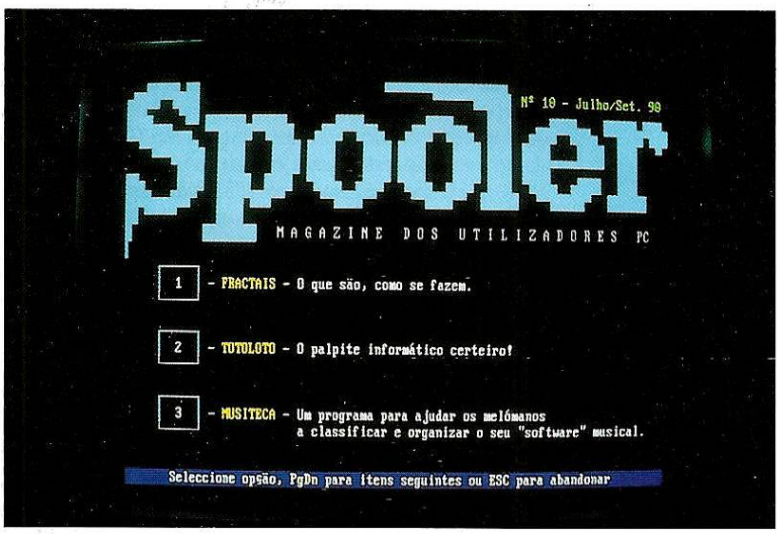
Mas se a palavra *Spooler* tem sete caracteres como é que aparecem catorze bytes em hexadecimal? Isso é devido à maneira como o ecrã está organizado – a cada posição de impressão segue-se o respectivo valor para o atributo de cor (**0E** é, em hexadecimal, o valor do atributo referente à cor amarela em fundo preto). Em modo texto (há também o modo gráfico que trata as coisas ao nível do *pixel* , mas isso é assunto para outro artigo) temos **80** posições na horizontal e **25** linhas na vertical. Multiplicando esses valores e dobrando o resultado, chegamos à conclusão que um ecrã completo ocupa **4 000** bytes.

Com estes elementos, e depois de saber qual o valor atribuído ao segmento do *buffer* do nosso ecrã, poderíamos escrever a seguinte rotina para imprimir a palavra *Spooler* em cor amarela sobre fundo preto:

```
PRINT_1:  PUSH ES          ; Guarda-se o registo que vai
                           ; ser afectado.
          MOV ES,[ECRAN]   ; No endereço "ECRAN" está o
                           ; valor para o segmento do
                           ; buffer do ecrã.
          MOV SI,DADOS     ; Em "DADOS" começa a
                           ; string com os dados copiar.
          MOV DI,0
          MOV CX,14        ; Número de bytes a copiar.
          CLD              ; Incrementa os registos DI e
                           ; SI depois de MOVSB.
L0:       MOVSB            ; Copia um byte de DS:SI
                           ; para ES:DI.
                           ; (O registo DS não foi
                           ; inicializado. Subentende-se
                           ; que ele tem o valor do
                           ; segmento corrente).
          LOOP L0          ; O ciclo é repetido até serem
                           ; copiados 14 bytes.
          POP ES
          RET

DADOS:    DB 53h, 0Eh, 50h, 0Eh, 4Fh, 0Eh, 4Fh, 0Eh, 4Ch,
          DB 0Eh,45h, 0Eh, 52h, 0Eh
ECRAN:    DW B800h         ; Se o seu ecrã é monocromático
; este valor poderá ser B000h (neste caso não reclame se a
; palavra Spooler não aparecer a amarelo).
```

Vou abrir aqui um parêntesis: (Como autodidata tenho algumas manias. Uma delas é a de não usar variáveis pré-declaradas e utilizar sempre posições endereçadas de memória para o efeito. Explicar por que faço isso é mais complicado. Creio que, com este processo, evito confusões para mim e para o ''assemblador''. Sendo assim, uma etiqueta dentro de parêntesis rectos é sempre o conteúdo de uma ou duas posições de memória, *byte* ou *word* conforme foram definidas; e uma etiqueta sem parêntesis rectos é sempre o próprio endereço. Uma outra mania é usar sempre que possível numeração decimal).



O valor do segmento do *buffer* do ecrã é normalmente **B800**h e **B000**h respectivamente para placas poli e monocromáticas (ou 45056 e 47104 em bons e velhos decimais). Este valor pode ser pedido à própria máquina como é o caso da rotina **VIDEO** que pode ver também em **SPOOL_A.ASM**.

Embora extremamente rápida, a rotina atrás descrita tem o inconveniente de ser muito trabalhosa, e algo idiota no que se refere à escrita dos dados a imprimir. De facto, se a cor se mantém igual em toda a palavra a imprimir, por que não arranjar um modo mais prático de fazer as coisas?

```
PRINT_2:  PUSH ES
          MOV ES,[ECRAN]
          MOV SI,DADOS
          MOV DI,0
          CLD
L0:       MOV AL,[SI]      ; AL espreita o byte a ser copiado
                           ; para o ecrã.
          CMP AL,1         ; Se o byte encontrado tiver o
                           ; valor 1 salta-se para L1.
          JZ >L1
          CMP AL,0         ; Se o valor encontrado for 0
                           ; salta-se para L2.
          JZ >L2
          MOVSB            ; Transfere um byte de "DADOS"
                           ; para o ecrã.
          PUSH SI          ; Guarda-se a posição
                           ; do próximo carácter.
          MOV SI,CORTM     ; Vai-se procurar o atributo
                           ; para a cor noutro sítio.
          MOVSB            ; Copia o byte da cor para o ecrã.
          POP SI           ; Recupera-se SI.
          JMP L0           ; Repete-se tudo até ser
                           ; encontrado na string um byte
                           ; com valor 0.
L1:       INC SI           ; Aponta-se para o endereço que
                           ; contém o atributo para a cor.
          MOV AL,[SI]      ; O registo AL "apanha" o valor
                           ; do atributo
          MOV [CORTM],AL   ; e guarda-o no endereço
                           ; "CORTM" (cor temporária).
          INC SI           ; Aponta-se SI para a posição
                           ; seguinte em "DADOS".
          JMP L0
L2:       POP ES
          RET

DADOS:    DB 1,14,'SPOOLER',0  ; O "assemblador" é que vai
; converter para os respectivos códigos de carácter tudo o que
; se encontra dentro das pelicas.

CORTM:    DB 0
ECRAN:    DW 47104         ; ou 45056 ou...
```

Esta rotina já apresenta algumas melhorias em relação à primeira: o byte com valor **1** nos dados indica à rotina que se segue o valor para o atributo. Esse valor, até nova ordem, é então usado para colorir todos os dados que se seguem. O zero é usado para indicar fim de dados e provoca o retorno da rotina. Para imprmir *Spooler Magazine* com *Spooler* a vermelho e *Magazine* a amarelo sobre fundo preto, a *string* de dados teria esta redacção:

```
DADOS: DB 1,4,'SPOOLER',1,14,'MAGAZINE',0
```

À rotina **PRINT_2** falta-lhe ainda muita coisa para ser verdadeiramente versátil. Vamos incluir-lhe uma maneira de definir a posição de impressão no ecrã relativamente a coordenadas do género **X** e **Y**. Os valores para a **coluna** (coordenada horizontal) deverão situar-se entre **0** e **79** e os valores para **linha** (coordenada vertical) entre **0** e **24.** Estes valores são correntes nas linguagens chamadas de alto nível.

```
PRINT_3:  PUSH ES
          MOV ES,[ECRAN]
          MOV SI,CX       ; Agora é CX que trás o endereço
                          ; dos dados a imprimir no ecrã.
          MOV DI,0
          CLD
L0:       MOV AL,[SI]
          CMP AL,1
          JZ >L1
          CMP AL,0
          JZ >L2
          CMP AL,5        ; Se o byte nos dados for 5 isso
                          ; significa que se seguem valores
                          ; para coordenadas linha – coluna.
          JZ >L3
          MOVSB
          PUSH SI
          MOV SI,CORTM
          MOVSB
          POP SI
          JMP L0
L1:       INC SI
          MOV AL,[SI]
          MOV [CORTM],AL
          INC SI
          JMP L0
L2:       POP ES
          RET
L3:       INC SI          ; Aponta-se para a posição que
                          ; contém o valor para a linha.
          MOV AL,[SI]     ; AL=linha.
          INC SI          ; Segue-se a posição com o valor
                          ; para a coluna.
          MOV BL,[SI]     ; BL=coluna.
          MOV BH,160      ; BH vai ser multiplicador.
          MUL BH          ; Multiplica-se AL por BH.
                          ; O resultado vem em AX.
          ADD BL,BL       ; Duplica-se o valor de coluna
          MOV BH,0        ; Isto é para BX ficar igual a BL
          ADD AX,BX       ; Adiciona-se AX a BX e está
                          ; encontrada a posição
          MOV DI,AX       ; no ecrã que se copia para DI.
          INC SI          ; A fórmula usada foi:
                          ; DI=(160*linha)+(2*coluna).
          JMP L0

DADOS:    DB 1,14,5,12,35,'SPOOLER',0
CORTM:    DB 0
ECRAN:    DW 47104        ; ou 45056 ou...
```

E pronto, desta vez imprimos *Spooler* bem no meio do ecrã. Uma versão mais elaborada desta rotina encontra-se na fonte **SPOOL_A.ASM** e, como já disse, chama-se **PRINTV**. Se o que aqui foi apresentado lhe abriu o apetite, então não deixe de ir à disquete dar-lhe uma vista de olhos, pois outras possibilidades são aí apresentadas tais como: repetição de bloco, incremento de linha, etc... (gastei algumas horas a escrever os comentários para si – portanto faça o favor de os ler!..)

Para testar a rotina **PRINT_3** basta chamá-la da seguinte maneira:

```
MOV CX,DADOS
CALL PRINT_3
INT 32          ; Esta interrupção manda-o
                ; de volta para o DOS.
```

Isto é quase tão simples como **LOCATE 12,36: COLOR 14,0: PRINT "SPOOLER": SYSTEM** e muito, muito mais rápido, uma vez que o processador não vai perder tempo a traduzir aquela linguagem que, para ele, é muito arrevesada. Depois, se compilar aquela linhazinha em *GWBasic* , verificará que o ficheiro **.EXE** ficará com **23979** bytes (compilado em QuickBasic) e que toda a rotina **PRINT_3** mais a rotina de chamada ocupam... **90** bytes.

Em *assembly* lidar com a imagem é um prazer. Imagine que pode copiar toda a imagem presente no ecrã para determinada posição na memória, imprimir sobre a imagem no ecrã, por exemplo, um quadro com instruções, e depois restaurar o ecrã com a imagem anterior que tínhamos armazenado, sem se notar qualquer oscilação no ecrã, como se o tal quadro com instruções aparecesse e desaparecesse como por artes mágicas. E isso é bem simples. E depois das rotinas-chave preparadas, basta fazer uns **CALL** para aqui e para acolá, sem nunca nos preocuparmos se o programa vai ficar lento. Às vezes é necessário até "travar" o *assembly* utilizando rotinas de pausa.

CORRER PROGRAMAS EXTERNOS

Para o programa do *menu* da disquete utilizei uma função do DOS que se serve do *loader* do **COMMAND.COM** para carregar em memória e correr programas externos. Quando, por exemplo, o leitor jogou com o **SPOOGAME** na disquete do n.º 9, foi essa rotina que foi chamada. Para os leitores que só dispunham da carta Hercules monocromática, essa rotina foi chamada por duas vezes: primeiro para correr o utilitário **CGA2.COM** para emular em CGA essas cartas e a seguir para executar o jogo. Confesso que tive, de início, algumas dificuldades em dominar essa rotina e em compreender o seu funcionamento. A sua estrutura é, basicamente, a seguinte:

```
SHELL:    MOV AH,75       ; Função 75 do DOS (EXEC).
          MOV AL,0        ; AL=0 para carregar e executar o
                          ; programa externo.
          MOV DX,NOME     ; NOME é o endereço onde
                          ; começa a string com o nome
                          ; do programa a executar.
          MOV BX,PARM     ; PARM é onde começa a zona de
                          ; parâmetros para o programa.
          INT 33
          RET

NOME:     DB 'C:\DIVERSOS\TESTE.COM',0   ; A string pode
          ; especificar drive\directório além do nome do
          ; programa a executar e deve terminar com um zero.
```

A zona de parâmetros, para melhor compreensão, pode

simplificar-se da seguinte maneira:

```
PARM:  DW 0    ; Segmento onde o programa externo
               ; vai alojar-se
       DW 0    ; Endereço da string de parâmetros (*)
       DW 0    ; Segmento da string de parâmetros
       DD 0    ; End. e segmento para o 1.º FCB (**)
       DD 0    ; Idem para o 2.º FCB
```

(*) – Quando se escreve no *Prompt* , por exemplo, **XCOPY A: C: /S /E**, os parâmetros do programa XCOPY.EXE são **A: C: /S /E** (o exemplo não é famoso). A *string* de parâmetros deve também terminar com um zero.

(**) – FCB – *File Control Block*. Explicarei isto noutra ocasião. Para já basta deixar lá os zeros.

Antes de chamar a rotina **SHELL** é necessário ''reservar'' memória suficiente para o programa que se pretende executar. Depois de vários falhanços, cheguei à conclusão que a melhor maneira de o fazer é invocando a função **74** do DOS (ver **RSMEM** em **SPOOL_A.ASM**). Desse modo o valor para os segmentos nos parâmetros é o do corrente segmento (**CS**).

Podem também chamar-se funções específicas do sistema operativo, tais como **DIR, COPY, TYPE**, etc., se o nome para o programa a executar for '**COMMAND.COM**', e a *string* de parâmetros começar por ' **/C**'. (Note-se no espaço antes da barra). Isto é necessário, pois o programa carrega outro COMMAND.COM em memória e a barra e o C ''descarregam-no'' quando já não é necessário. Segue-se um exemplo de como escrever as *strings* de modo a efectuar uma cópia do ficheiro **TESTE.COM** do *drive* **A** para o disco rígido:

```
NOME:  DB '\COMMAND.COM',0
PARM:  DB ' /C COPY A:\TESTE.COM C:',13,0 ; No caso do
       ; programa COMMAND.COM a string de parâmetros
       ; deve terminar com os valores 13 e 0.
```

OS SONS

Com respeito aos sons, o pouco que consegui descobrir deixou-me com vontade de querer saber muito mais. De facto é espantoso o que se consegue fazer em *assembly* com os sons. É possível, por exemplo, criar a ilusão de sons compostos, ou seja, escutar um acorde harmónico em vez de uma única nota melódica. Prometo dedicar algum do meu tempo (pouco) livre a estudar o assunto. Na disquete seguem vários efeitos sonoros, muito mais fruto da experimentação do que de uma verdadeira compreensão dos mesmos. O princípio utilizado resulta semelhante ao das funções **SOUND** de algumas linguagens. Quando fizer experiências com as rotinas tenha cuidado com os valores que atribui para a frequência de duração dos sons, pois pode produzir sons que duram horas...

Faça vários efeitos sonoros e através de uma rotina de detecção do teclado chame cada efeito a partir de determinada tecla. O resultado é surpreendente, vai ver.

Preparei as rotinas mais importantes de modo a que fossem o mais versáteis possível e de maneira a poderem ser utilizadas em qualquer outro programa. Depois de feitas e testadas podem ser esquecidas, embora continuem lá, algures na listagem, a prestar um bom serviço. Quando nos meus programas *assembly* escrevo **MOV CX,MOLDR**, **CALL PRINTV**, e quando o resultado desta instrução é a elegante e instantânea cercadura que aparece a envolver o texto quando seleccionado um item do *menu* da disquete, dou por bem empregues as horas que passei a iniciar-me nesta linguagem de baixo nível.

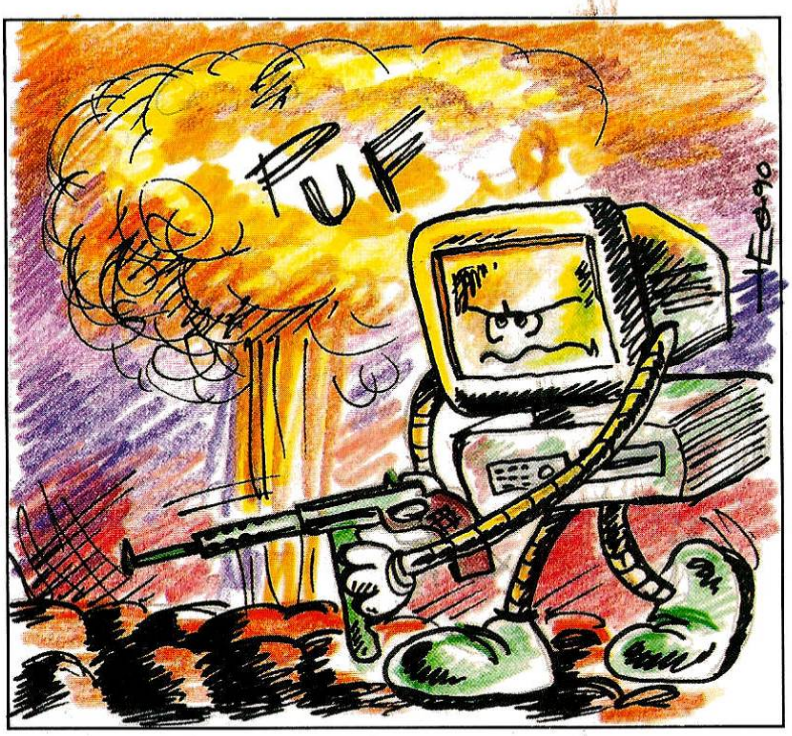
Outras rotinas fazem parte do cardápio da disquete: rotinas de *scroll* parcial do ecrã, rotina de detecção do teclado, posicionar e esconder o cursor, etc., etc., etc..

O ficheiro **SPOOL_B.ASM** contém exclusivamente dados para a rotina **PRINTV**.

Et voilá! O segredo do *menu* da disquete foi desvendado. Como, para além do *menu*, temos livro de reclamações, cá estou para o que der e vier.

Franco Gomes

ADA: UMA LINGUAGEM BÉLICA



> **O mais grave é que o custo do software não era, na altura, sequer linear com a sua extensão, mas quadrático**

"Bolas, não há dinheiro que pague isto". Isto e muito mais diriam, provavelmente, os coronéis e generais do Departamento de Defesa do Estados Unidos da América (DDEUA) em princípios dos anos setenta, a propósito dos custos do software.

E porquê? Bom, os números eram auto-explicativos – três biliões de dólares gastos anualmente em software para sistemas integrados. Enfim, um balúrdio. E tão mal aplicadinhos, que até fazia dó. Quantos jovens e promissores norte-americanos acabariam por ficar sem possibilidade de conhecer novas culturas e os exóticos países do Sudeste Asiático – como o Vietname, Cambodja e Laos? Era um preço que a Nação não se podia dar ao luxo de pagar!

E quando acabasse a guerra? Havia de ser a derrocada. O congresso – aqueles bananas – não teria pejo algum em nos cortar as nossas preciosas verbas. Se calhar, até propor que a guerra deixasse de ser subsidiada pelo Estado. E passasse a estar submetida às leis do mercado...

Não, não, havia que achar maneira de baixar o custo destes malditos sistemas integrados. Até porque, se não, seriam cada vez menos integrados. Sim, porque num sistema deste tipo temos um computador ligado a uma autêntica panóplia de sistemas de comunicação e armamento. E pelo andar da carruagem, para pagar o raio dos malditos programas para o computador, teríamos que começar a vender as preciosas bombitas que são, afinal, a nossa verdadeira matéria-prima. Sem elas, não há guerra...! Pois se já agora nos vai faltando a massa p'rós *napalms*...

QUATROCENTAS LINGUAGENS E DIALECTOS

O mais grave é que o custo do software não era, na altura, sequer linear com a sua extensão, mas quadrático. Consequentemente, projectos suficientemente grandes tornavam-se aparentemente infazíveis.

Apuradas as culpas de tamanha vaga inflacionista, chegou-se à conclusão que o problema estava na utilização de um número enorme de linguagens de programação e respectivos dialectos – que, ao que parece, ascendiam à espantosa marca de mais de quatrocentos – ainda por cima, todos eles obsoletos e inadequados.

O DDEUA decidiu pois criar um grupo de trabalho, o Higher Order Language Working Group (HOLWG), destinado a produzir as especificações de uma linguagem, que teria necessariamente que ser o *state-of-the-art* em linguagens de programação. Este grupo nasceu em 1975 e entrou imediatamente em funcionamento, tendo ainda nesse mesmo ano produzido uma especificação preliminar, o Strawman. Depois foram sucessivamente lavradas novas especificações, cada uma delas mais detalhada e rígida – como deixam, de resto, os sugestivos nomes adivinhar – que a anterior: Woodenman(75), Tinman(76), Ironman(78) e finalmente o Steelman(79). É curioso ainda notar que, logo em 1977, o HOLWG decidiu que nenhuma das vinte e seis linguagens mais sofisticadas da altura, satisfaziam o que era pretendido. Ou seja, nem sequer da classificação, já de si confrangedora, de folha-de-flandres eram merecedoras.

Mas o que se pretendia, então, desta linguagem ? O documento elaborado pelo HOLWG focava três pontos fundamentais: *information hiding*, verificação de programas e concorrência.

INFORMATION HIDING

Esta expressão banalizou-se completamente nos últimos anos. A explicação é fácil: não há actualmente linguagem alguma cujo autor não se gabe em alto e bom som, de ser essa uma das suas principais características.

As *releases* do Turbo Pascal e restantes congéneres encarregam-se do resto. São produtos de ampla divulgação – embora eu não conheça pessoalmente ninguém que os compre – e que povoam densamente os disco cá no burgo. Além disso, os manuais a contar os pormenores são facilmente acessíveis – a indústria da fotocópia deve muito à informática... – e contam os pormenores das maravilhas da information hiding.

No entanto, para percebermos as origens desta expressão, temos de recuar até 1971. Vivia-se nessa altura, como já vimos, a angústia do custo do software. Um dos métodos para combatê-lo parecia ser dividir os programas em módulos tão independentes quanto possível. Desta

forma, cada um dos módulos podia ser desenvolvido e depurado separadamente, diminuindo substancialmente a complexidade do programa. D.L. Parnas, da Universidade de Carnegie Mellon, desenvolveu, então, um famoso princípio de programação que consistia, afinal, em esconder dentro de um módulo as particularidades de implementação desse mesmo módulo. Mais especificamente, Parnas pretendia que este módulo fosse uma espécie de caixa preta com um folheto de instruções. O utilizador do módulo não deve, nem pode, preocupar-se com o que está dentro da caixa, mas unicamente com a sua funcionalidade. Por outro lado, o implementador não tem, de algum modo, de preocupar-se com qual será a futura utilização do módulo: interessa-lhe exclusivamente conseguir que este tenha a funcionalidade prometida.

UM CONCURSO

Como vimos, já em 1977, a HOLWG tinha concluído que nenhuma das linguagens existentes se revelava satisfatória. Era, pois, altura de arranjar a linguagem que satisfizesse os necessários requisitos. Fez-se, para isso, um concurso que envolveu diversos gigantes da indústria mundial. E foi assim que, após uma renhida luta, envolvendo inicialmente dezasseis projectos, se passou sucessivamente a quatro, dois, e finalmente um.

A vitória foi para uma equipa da CII-Honeywell-Bull liderada por Jean Ichbiah, um francês. Ou seja, a Europa acabou por sair vencedora do concurso lançado pelo todopoderoso DDEUA. A árdua tarefa de baptizar a linguagem coube, contudo, aos norte-americanos, tendo o HOLWG escolhido o nome em Maio de 79: Ada. Em honra a Augusta Ada, condessa de Lovelace e filha do famoso Lord Byron.

Diga-se de passagem que Lord Byron adorava Sintra, que consistia, afinal, a matéria-prima para os seus poemas mais sedutores. A conclusão a tirar não pode, pois, ser outra: sem Sintra não existiria, hoje, Ada. E é, afinal, deste modo inusitado, que Portugal acaba por ter o seu papel mais importante em toda a história da informática.

O leitor mais atento não terá deixado de se interrogar sobre o motivo que levou à escolha de Augusta Ada. Tratava-se de uma matemática e foi a programadora de Charles Babbage. Desta forma tornou-se, nem mais nem menos, na primeira programadora de sempre. Diga-se, de resto, que nomear uma linguagem em honra a um matemático não é propriamente original. Senão, vejam-se os exemplos das linguagens Pascal, Euler e Euclid.

PASCAL LIKE

Sintacticamente, a linguagem Ada é tipo Pascal. Trata-se de uma linguagem fortemente tipada, tendo-se entretanto resolvido diversos problemas que surgiam no Pascal dentro deste domínio, como sejam os registos com variante e a passagem de funções e procedimentos por parâmetro. Recordo que estas duas características permitiam furar o mecanismo de tipos daquela linguagem.

A linguagem Ada apresenta sensivelmente os mesmos construtores de tipos que o Pascal: enumerados, arrays, records e apontadores. Por forma a manter seguro o seu sistema de tipos, o Ada só permite variar a discriminante de um registo com variante, afectando de uma só vez um registo inteiramente novo. Desta forma, o discriminante reflecte sempre qual o campo do registo utilizável no momento.

A propósito do mecanismo de tipos, vimos ainda que um dos problemas que surgia no Pascal, era a impossibilidade de escrever uma função que actuasse sobre qualquer array, independentemente da sua dimensão. Isto acontecia porque o Pascal obriga a função a declarar os tipos de todos os seus argumentos, e ainda porque os limites extremos dos índices do array faziam parte do seu tipo.

```
generic

    SIZE: POSITIVE;
    type ELEMENT is private;

    package STACK is

    procedure PUSH ( E : in ELEMENT );
    function POP return ELEMENT;

    OVERFLOW, UNDERFLOW: exception;

end STACK;
```

Figura 1

Na linguagem Ada torneou-se este problema, permitindo especificar um tipo array sem explicitar quais os limites dos seus índices. Vejamos um exemplo:

```
type VECTOR is array ( INTEGER <> ) of FLOAT;
```

Ao declararmos variáveis deste tipo, temos agora de dizer qual a dimensão do array:

```
PESO: VECTOR( 1.. 20 );
COMPRIMENTO: VECTOR( 1.. 40 );
```

Podemos agora escrever uma função SOMA para qualquer variável do tipo VECTOR:

```
function SOMA ( VEC: VECTOR ) return FLOAT is
    TOTAL : FLOAT := 0.0;
begin
    for I in VEC'FIRST.. VEC'LAST loop
    TOTAL := TOTAL + VEC( I );
    end loop;
    return TOTAL;
end SOMA;
```

Falta explicar como aparecem VEC'FIRST e VEC'LAST a delimitar o ciclo for. É que, por cada array passado por parâmetro, a linguagem passa mais dois parâmetros implícitos: o primeiro e o último índice do array. Note-se que não se trata de uma mera questão de comodidade para o programador. Se este fosse obrigado a passar esses limites explicitamente, isso tornar-se-ia uma possível fonte de erros que, assim, é graciosamente eliminada.

Note-se ainda que a variável que comanda o ciclo *for* não é declarada em Ada. Resolve-se assim o eterno dilema dos programadores Pascal, que passam a vida a interrogar-se sobre se o valor dessa variável se mantém após a saída do ciclo (a resposta é que não se mantém obrigatoriamente, sendo pois o seu valor indefinido). Uma vez que em Ada a variável só existe dentro do ciclo, não sendo sequer possível nomeá-la fora deste, a questão não se chega sequer a pôr.

Fascinante no Ada, são sem dúvida estas soluções inteligentes que nos vão surgindo a par e passo.

PACOTES...

Como já vimos, uma das particularidades exigidas à linguagem Ada era a de possuir mecanismos de information hiding. Vimos já também que um processo de consegui-los é através da utilização de módulos.

Pois bem, na terminologia Ada os módulos designam-se por *packages*. Cada package possui uma especifica-

“Sintacticamente, a linguagem Ada é tipo Pascal. Trata-se de uma linguagem fortemente tipada, tendo-se entretanto resolvido diversos problemas que surgiam no Pascal”

```
package body STACK is

  ST : array ( 1 .. SIZE ) of ELEMENT;
  SP : POSITIVE;

function FULL return BOOLEAN is
begin
  return SP > SIZE;
end FULL;

function EMPTY return BOOLEAN is
begin
  return sp = 1;
end EMPTY;

procedure PUSH( E : in ELEMENT ) is
begin
  if FULL then raise OVERFLOW; end if;
  ST( SP ) := E;
  SP := SP + 1;
end PUSH;

function POP return ELEMENT is
begin
  if EMPTY then raise UNDERFLOW; end if;
  SP := SP - 1;
  return ST( SP );
end POP;

end STACK;
```

Figura 2

> **“A especificação ou definição do package funciona um pouco como um contrato que o implementador do mesmo se compromete a cumprir”**

ção, a interface, e um corpo. A especificação ou definição do package funciona um pouco como um contrato que o implementador do mesmo se compromete a cumprir. Explicita as funções, procedimentos, variáveis e excepções que o package exporta.

Vejamos então um exemplo de um package, patente nas figuras 1 e 2. Trata-se de uma implementação de *stacks*.

Quem tenha seguido esta série de artigos saberá já, sem dúvida, o que é um stack ou pilha. Trata-se de uma estrutura de dados facultando essencialmente duas operações: *push* (empilhar) e *pop* (desempilhar). Tal como numa pilha de pratos, o primeiro elemento a sair é sempre aquele que entrou por último.

Comecemos por analisar a definição do package (fig.1). Ignoremos por agora as três primeiras linhas. Vemos que cada package tem um nome, e que o nome deste é stack. Seguem-se as declarações do que é exportado pelo package: o procedimento push (para empilhar um elemento no stack) e a função pop (que devolve o elemento no topo do stack). Note-se que, em Ada, as funções podem devolver qualquer tipo de dados.

Vemos ainda que o package exporta duas excepções: overflow (stack cheio), e underflow (stack vazio). O mecanismo de excepções do Ada reflecte a perspectiva de que os erros ocorrendo no interior do package devem ser tratados por quem o utiliza, e não por quem o implementa. Assim, sempre que ocorre um erro, é levantada uma excepção – mais à frente veremos um exemplo disso – e será da responsabilidade do invocador tratá-la.

Na fig. 2, temos a implementação do package. Esta é conseguida à custa de um array, mas como é evidente, nada nos impediria de fazê-lo através de uma lista. O que pretendo, afinal realçar, é o facto de o utilizador do package, ao utilizá-lo, não precisar saber como foi ele implementado. Na verdade, e a menos que tenha acesso ao código da implementação, mesmo que queira sabê-lo não terá grandes hipóteses de o fazer. Para além das vantagens já apontadas, isto permite que a implementação de um package seja alterada, sem terem sequer que se recompilar os utilizadores desse package.

Como se pode verificar, o package define duas funções para uso interno: *full*, que determina se o stack está cheio e *empty* que detecta se este está vazio.

A implementação do package é trivial, restando contudo focar o papel das excepções. Estas são levantadas, respectivamente quando se pretende empilhar um elemento com o stack cheio, ou desempilhar um outro com o mesmo vazio.

O tratamento das excepções deverá ser definido pelo procedimento invocador, no fim do mesmo:

```
procedure INVOCADOR (.....) is
.....
      push (...);
.....
      pop (...);
.....
exception
      when overflow =>
..... código a executar quando ocorrer
      overflow.....
      when underflow =>
.....
end INVOCADOR;
```

Ao ser levantada a excepção overflow em push, por exemplo, é passada a este procedimento, sendo pois executado o código especificado. No caso de não existir tratamento para a excepção no procedimento invocador, o que não é, obviamente, o caso , esta seria passada ao procedimento ou função que o tivesse invocado. E assim sucessivamente, até a excepção poder ser tratada. Se ninguém tratasse a excepção, o programa acabaria por abortar.

POLIMORFISMO

Certas e determinadas operações deveriam ser completamente independentes do tipo de dados a que se aplicam. É o caso do nosso stack: estejamos nós a empilhar inteiros, ou registos complicadíssimos, a semântica da operação é sempre a mesma. Seria, pois, interessante dispor de um mecanismo que nos permitisse definir packages, por exemplo, independentes do tipo de dados a que se aplicam polimórficos.

O Ada dispõe realmente de um mecanismo deste tipo, o *generic*. E isso leva-nos de volta às três primeiras linhas da interface do nosso package (fig.1). Como pode verificar-se, a definição do tipo *element* fica adiada, bem como o tamanho da pilha. Na realidade, será o utilizador do package que se encarregará de especificá-los. Ao fazê-lo, dizemos que cria uma nova instância do package:

```
procedure INVOCADOR (.....) is
  package STACK_FLOAT is new STACK (1000, FLOAT);
begin
.....
  STACK_FLOAT.PUSH (10.32);
.....
end INVOCADOR;
```

No nosso exemplo, STACK_FLOAT passa, assim, a ser um novo package implementando um stack de floats.

OVERLOADING DE OPERADORES

Em Ada, um procedimento ou função não se designa unicamente pelo seu nome, mas ainda pelos tipos dos seus argumentos. Trata-se da sua assinatura. Isto permite que tenhamos duas funções diferentes com o mesmo nome:

```
function QUADRADO ( X : INTEGER ) return INTEGER is
begin
      return X * X;
end QUADRADO;

function QUADRADO ( X : FLOAT ) return FLOAT is
begin
      return X * X;
end QUADRADO;
```

Note-se que em Pascal, por exemplo, teríamos mesmo de ter dois nomes diferentes, o que cria por vezes uma certa artificialidade (IntQuad, FloatQuad,... ?).

Mas o Ada permite-nos ainda redefinir os próprios operadores. Assim, e relembrando a função SOMA, poderíamos utilizar o operador + para somar dois vectores:

```
function "+" ( X, Y : VECTOR ) return VECTOR is
begin
.....
end "+";
```

Como sabe o Ada que operador aplicar, afinal? A resposta é simples: através da respectiva assinatura. Assim, quando temos

```
X := 1+2
```

e uma vez que 1 e 2 são inteiros, o Ada utiliza a soma de inteiros predefinida. Quando temos

```
PESOS1, PESOS2, PESOS: VECTOR ( 1.. 500 );
.....
PESOS := PESOS1 + PESOS2;
```

O Ada sabe agora que a operação é entre vectores e chama a função por nós definida.

Diz-se que um operador está *overloaded* quando suporta simultaneamente diversas operações. Como é evidente, trata-se apenas de *syntatic sugar* e que nada de verdadeiramente novo traz à linguagem. Indesmentível, contudo, é que permite aumentar significativamente a expressividade da mesma.

CONCORRÊNCIA

É sabido que o DDEUA pretendia uma linguagem para desenvolver sistemas de tempo real. Ora isso implica, naturalmente, capacidade e mecanismos para a concorrência. Pretendia-se além disso, que o paradigma de concorrência fosse aplicável a sistemas distribuídos, o que punha de parte mecanismos clássicos como os semáforos e os monitores pois acentam em memória partilhada. Consequentemente, as *tasks*, que gerem a concorrência no Ada, funcionam segundo o princípio do *rendez-vous*. Desta forma, estabelecem-se pontos de sincronização entre as diversas tasks, e a comunicação entre elas é feita nesses pontos.

ADA OBJECT ORIENTED?

Muito se fala por cá, agora, de linguagens orientadas por objectos. É natural. Afinal, já se passaram vinte e três anos sobre o Simula e dezoito sobre a primeira implementação de Smalltalk. Sendo nós conhecidos por estarmos sempre em cima do acontecimento, não admira, pois, que se trate agora de um assunto de extrema actualidade cá no burgo.

Quanto à questão inicial, respostas contraditórias vão surgindo um pouco por todo o lado. Para respondê-la, teríamos de saber o que é, exactamente *object-oriented*. Bertrand Meyer dá uma definição que me parece bastante satisfatória, no seu artigo de 1988, "Object oriented software construction":

Trata-se de um método para construção de sistemas de software, por combinação de unidades básicas chamadas classes. Cada classe é uma implementação, possivelmente parcial, de um tipo abstracto de dados e pode estar ligada a outras classes através de duas relações bem distintas:

– Cliente, possibilitando que a implementação de uma classe utilize facilidades postas à disposição de uma outra por intermédio da sua interface.

– Múltipla herança, onde uma classe é definida como uma extensão ou especialização de uma outra ou outras.

Ora o Ada dispõe apenas de mecanismos que apontam para o primeiro aspecto. Sinto-me, pois, tentado a parafrasear Bjarne Stroustrup (autor do C++), a propósito da vaga evangelizadora que se vem fazendo no sentido de tornar object-oriented o Pascal, o C, o Modula-2 ou o Chill. Dentro em pouco, haverá quem descubra que o Cobol ou o Fortran são, afinal, linguagens de programação por objectos...

A propósito do Ada, Stroustrup acaba por desabafar dizendo que object-oriented se tornou sinónimo de bom, o que tem acabado por produzir na imprensa especializada, silogismos do tipo:

Ada é bom
Object-oriented é bom

---

logo -> Ada é object-oriented

CONCLUSÃO

Espero que tenha ficado claro, ao longo do artigo, que a linguagem Ada não é pequena. Na verdade é bastante grande, sendo esse, aliás, o maior defeito que geralmente se lhe aponta.

Enquanto a linguagem Pascal tem cerca 500 tokens, o Ada aproxima-se dos 1600, o que é significativo. Isso resulta sem dúvida, de o DDEUA ter pretendido, desde sempre, uma linguagem de utilidade absolutamente genérica. Vimos já, a propósito do PL/I, as grandes desvantagens destas linguagens megalómanas: dificuldade em dominá-las em extensão, imprevisibilidade do resultado de interacções entre características e, enfim, tudo o que isto acaba por acarretar para o infeliz do programador.

Além disso, o DDEUA proibiu que se implementassem subconjuntos da linguagem. O Ada é hoje uma marca registada, e qualquer compilador de Ada tem que ser submetido a um exame rigoroso, antes de se poder orgulhar da designação de "compilador de Ada". Pretendia-se, assim, evitar a proliferação de dialectos da linguagem, que acabam sempre por afectar, decisivamente, a portabilidade dos programas. Trata-se, sem dúvida, de um motivo completamente válido, mas o facto de se ter aplicado tal regra a uma linguagem com as dimensões do Ada, terá levado o famoso C.A.R. Hoare a afirmar:

Para teres uma linguagem com subconjuntos, fá-la pequena.

Mas por que não se há-de, afinal, fazer uma linguagem que dê para tudo? Uma só linguagem para aprender, uma só com que trabalhar... Bem, diversas respostas se podem dar a esta questão. A primeira, é que o problema não é das linguagens de programação. Também nos poderíamos interrogar acerca da possibilidade de criar uma ferramenta de carpintaria que servisse para martelar, plainar, serrar e limar, e contudo não o fazemos, normalmente.

A segunda, reveste-se do facto de vivermos num mundo livre. O que é que há-de impedir a uns de fazer linguagens e, a outros, de escolhê-las?

A terceira, e não necessariamente nesta ordem, é que se existisse uma só linguagem, ela resultaria num só artigo...

A despeito da sua extensão, o Ada consegue ser uma linguagem inovadora em muitos aspectos. Além disso, com o suporte do DDEUA, é quase garantido um sucesso de popularidade extremo. Pelo menos entre os militares...

João Cardoso

> **"A linguagem Ada não é pequena. Na verdade é bastante grande, sendo esse, aliás, o maior defeito que geralmente se lhe aponta"**

FRACTAIS

CONJUNTOS DE JULIA E DE MANDELBROT

Conjunto Julia xmin=-1, xmax=1, ymin=-1, ymax=1 c_real=-0.75 c_imag=0.25, it_max=100, 640×350 'r'

FRACTAIS

A palavra **fractal** foi inventada por Benoit **Mandelbrot,** um investigador do Thomas J. Watson Research Center, da IBM. Com esta palavra **Mandelbrot** pretendia designar objectos que são tão fracturados e complicados quando são vistos globalmente como quando são ampliados e vistos em detalhe. Está, por exemplo, neste caso a linha da costa: é uma linha sempre tão rregular quando a observamos de um satélite como quando a observamos de um avião ou na praia. Os fractais são hoje utilizados no estudo e descrição de fenómenos naturais em campos tão variados como a física, a biologia, a química, a econometria, a medicina ou a computação gráfica.

Neste último domínio os fractais são usados para modelar de uma forma simples certos aspectos da natureza que, para serem convincentes em termos de realismo, têm de ser complexos; é o caso de uma floresta, da superficie de um planeta ou de um terreno, dos ramos de uma árvore, de uma linha de costa, etc. A utilidade dos fractais em computação gráfica provém do facto de ser possível produzir imagens aparentemente muito complexas através de um processo ou fórmula simples, sem ser necessário descrever previamente os detalhes dessa complexidade. Analogamente, certas imagens naturais têm uma regularidade intrínseca que faz com que possamos obter fórmulas simples que gerem imagens computacionais semelhantes às imagens naturais. É esta a ideia que está na base de outra das aplicações dos fractais à computação gráfica: a compressão de imagens. É ainda possível usar os algoritmos que geram imagens fractais para a avaliação da *performance* das máquinas; disto falaremos mais adiante.

Vamos, neste artigo, dar alguns exemplos de técnicas

que permitem obter imagens de fractais num computador com uma placa gráfica. Para isso vamos usar dois exemplos que podemos já chamar clássicos, embora as suas imagens sejam conhecidos há cerca de dez anos apenas: o conjunto de **Mandelbrot** e os conjuntos de **Julia** associados à iteração que define o conjunto de **Mandelbrot.**

MANDELBROT E JULIA

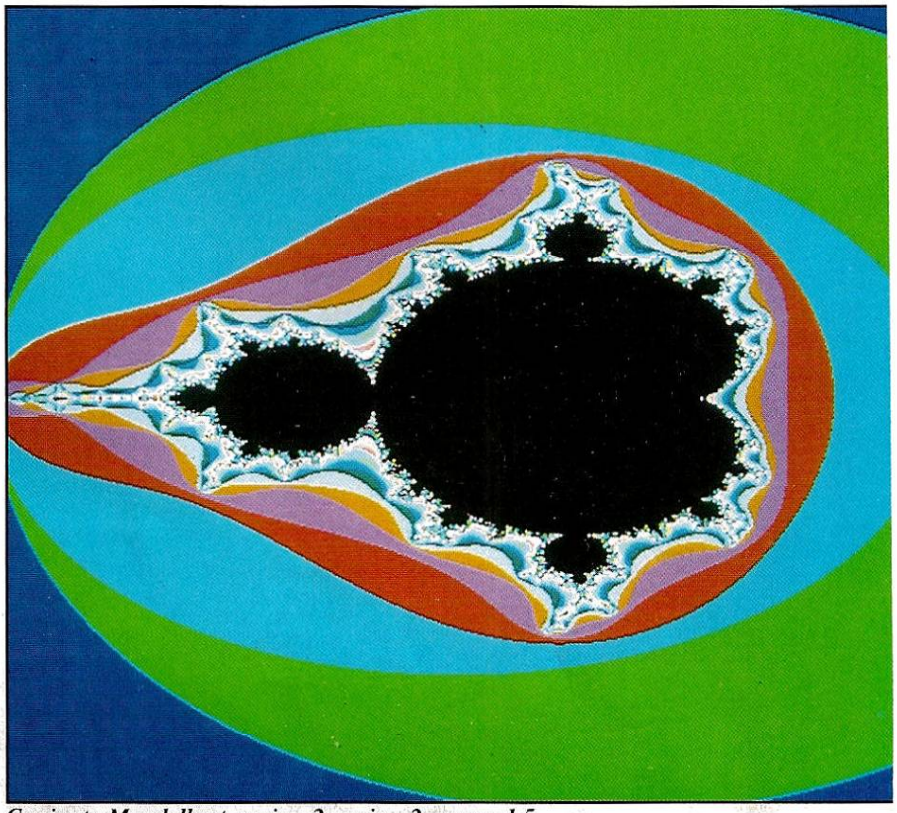
Imagens do conjunto de **Mandelbrot** foram pela primeira vez calculadas num computador pelo próprio **Mandelbrot** em 1980. Este conjunto ficou famoso não só pela sua complexidade como também pela beleza das imagens que é possível obter colorindo com um computador a zona que o rodeia; e tudo isto resulta de um algoritmo espantosamente simples. Propomo-nos descrever aqui o algoritmo que define o conjunto de Mandelbrot e algumas possibilidades para colorir adequadamente a zona adjacente ao conjunto. Vamos, no entanto, começar por outro tipo de fractais: os conjuntos de **Julia.** Para já é conveniente dizer que se dá este nome a uma classe de fractais porque estes conjuntos foram pela primeira vez estudados por dois matemáticos franceses: Pierre Fatou e Gaston Julia.

UTILIZAÇÃO DO PROGRAMA

Aos leitores mais apressados, que queiram experimentar já o progrma FRACTAIS que acompanha este artigo, sugiro que não se preocupem com a velocidade com que surge a imagem: isto deve-se à quantidade de cálculos em vírgula flutuante que é, por vezes, necessário fazer antes que o computador decida qual a cor a atribuir a cada pixel. Note-se que não é necessário ter um monitor a cores para poder apreciar os fractais; mesmo com uma placa Hercules ou CGA monocromática é já possível obter imagens belas e complexas. Ainda assim, as melhores imagens surgem de placas EGA, VGA e de placas mais sofisticadas.

O programa começa por perguntar qual a placa gráfica com que quer trabalhar (o que pode ser útil, como veremos adiante). Deve em seguida decidir se opta pelo conjunto de **Mandelbrot** ou por um conjunto de **Julia**, sendo neste último caso necessário fornecer dois números reais que formam o parâmetro do conjunto de **Julia** preenchido a ser desenhado. Para definir a janela pela qual vamos obsevar a imagem temos que definir as suas coordenadas **xmin, xmax, ymin** e **ymax,** devendo ter-se **xmin < xmax** e **ymin < ymax.** Estes valores podem ser positivos ou negativos e tão próximos quanto se queira, desde que **xmin** seja diferente de **xmax** e **ymin** diferente de **ymax.** Para obter as regiões mais interessantes, os valores absolutos destes números não devem exceder 2. No final deste artigo dão-se algumas sugestões de regiões a explorar. O número máximo de iterações depende de vários factores e em especial do computador que se está a usar e da paciência que se tiver: o mínimo para se obter uma imagem que valha o tempo de cálculo é da ordem de 10 a 15; num XT, não será mesmo possível ir muito além destes valores, a menos que deixe o programa a correr toda a noite. Num AT já se pode dar um valor na ordem das dezenas, sendo um valor de 100 bom para um 386. Estes valores dependem ainda da resolução do desenho e da região escolhida.

Sugiro que comece com uma resolução baixa (CGA, se possível) e com um número máximo de iterações de 10 num XT, 25 num AT e 40 num 386. A partir daí pode ir aumentando o número de iterações (o que dá uma imagem mais recortada e variada) e melhorando a resolução (o que dá uma imagem mais perfeita). Isto vale, em geral, para qualquer nova região que comece a explorar, pois algumas zonas têm um desenho menos interessante. A opção seguinte é mais técnica podendo, para começar, escolher



Conjunto Mandelbrot, xmin=-2, ymin=-2, xmax=1.5, ymax=2, 50 it_max 'r'

"r".

Depois de começar a desenhar uma imagem, pode voltar ao menu principal com ESC, ou gravar a imagem premindo "g". Neste caso, deve dar um nome ao ficheiro onde a imagem ficará armazenada. A actual versão do programa não verifica se o nome escolhido já existe, pelo que é necessário não o repetir. Quando se prime "g" a imagem é gravada num ficheiro IMTEMP.FRC. O nome dado pelo utilizador é depois usado para fazer um *rename* ao ficheiro temporário. No caso de surgir qualquer problema, pode sempre fazer o RENAME a partir do DOS. No entanto, nunca deve deixar uma imagem com o nome do ficheiro temporário, pois seria apagada na gravação seguinte.

Para ver uma imagem gravada, basta digitar o nome do respectivo ficheiro, precedido ou não da via do directório. Se o ficheiro em causa não tiver o formato adequado, o programa dá uma mensagem de erro.

"A utilidade dos fractais em computação gráfica provém do facto de ser possível produzir imagens aparentemente muito complexas através de um processo ou fórmula simples"

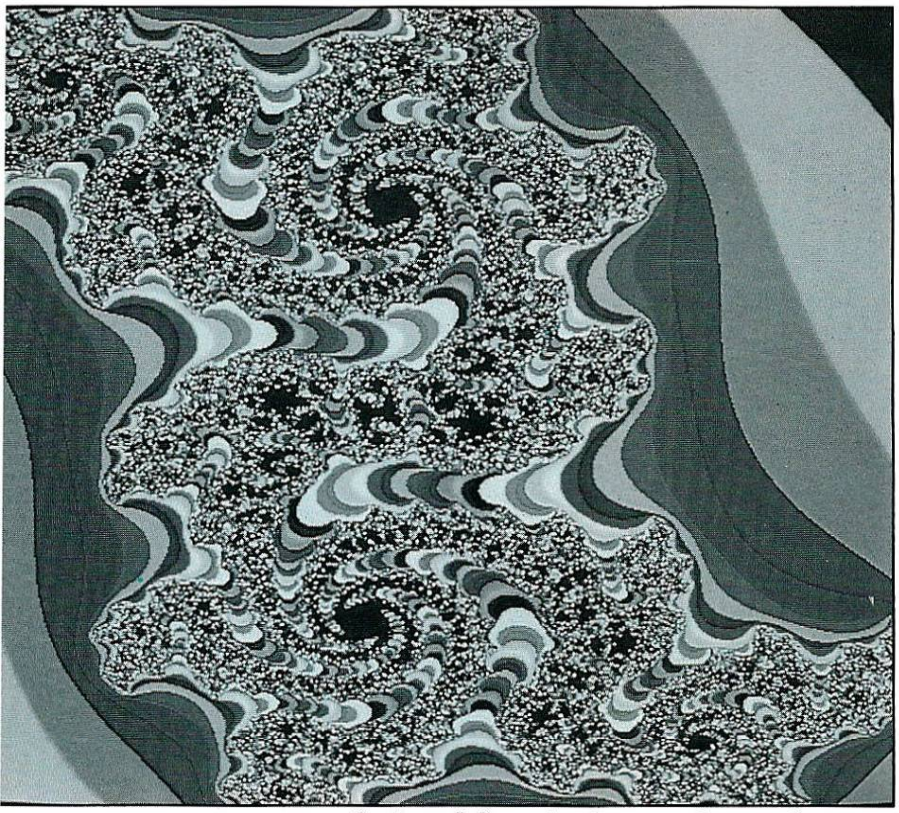
O PLANO COMPLEXO

Os objectos fractais que vamos desenhar existem num plano chamado plano complexo, que não é mais do que um plano geométrico em que a cada ponto está associado um par de números. Podemos, contudo, fazer contas com esses pares de números de modo a obter um resultado que é outro par de números. Estes pares vão ser chamados números complexos. São por exemplo números complexos os pares:

(3,4) (5, -1.5) (-1.23, 6.8) (-.75, .25) (0.0001, -0.004).

Dado um número complexo **(x, y)** é costume chamar a **x** a sua parte real e a **y** a sua parte imaginária. As razões destes nomes são históricas e devem-se ao facto de os primeiros matemáticos a usar os números complexos não estarem convencidos da sua existência real.

Se tivermos os números complexos **(x, y)** e **(a, b)** somamo-los da seguinte forma:



Conjunto Julia, xmin=-1, xmax==1, ymax=1, c_real=0.288 c_imagem=0.48365 320×200 'r'

(x, y) + (a, b) = (x + a, y + b)

Podemos também multiplicá-los, mas desta vez a definição não é aquela que poderíamos esperar:

(x, y) × (a, b) = (x × a - y × b, x × b + y × a).

> **... isto é um pouco como tentar fazer um mapa dos buracos nas ruas de Lisboa andando de automóvel pela cidade e marcando cada buraco à medida que caímos nele**

OS CONJUNTOS DE JULIA

Supunhamos agora que fixamos um certo número complexo **c = (c_real, c_imag).** Se representarmos por **z** o complexo **(x, y),** podemos pensar no seguinte processo

$$z \mapsto z^2 + c$$

isto é, em termos de partes reais e imaginárias,

(x, y) ⟼ (x, y) × (x, y) + (c_real, c_imag) =
= (x × x - y × y + c_real, 2 × x × y + c_imag).

Podemos agora pensar em **x** e **y** como variáveis, guardar lá o resultado das operações efectuadas e assim voltar a repetir o processo quantas vezes quisermos. A questão agora é saber o que vai acontecer com os sucessivos valores de **x** e **y;** será que a partir de certa altura se vão repetir sempre os mesmos valores de **x** e **y**? Dados os valores a **x** e **y** será que eles voltam a repetir-se alguma vez? E quanto aos valores absolutos de **x** e **y:** será que vão ter a mesma ordem de grandeza do valor inicial ou será que, se tentarmos programar o algoritmo, vamos ter um *overflow* da aritmética ao fim de poucos passos?

A resposta a estas questões está longe de ser simples, mas o que nos vai interessar vai ser distinguir dois tipos diferentes de números complexos:

a) aqueles que ao fim de um certo número de passos acabam por provocar um erro de excesso na aritmética de qualquer computador;

b) aqueles que não causam esse problema;

dos pontos que estão nas condições b) dizemos que estão no conjunto de **Julia** preenchido com parâmetro **(c_real, c_imag).**

A ideia seguinte é utilizar o computador para perceber quais são os pontos que estão no conjunto de **Julia** preenchido e quais os que não estão. Mas, dirá o leitor, se tivermos um programa para fazer os cálculos, cada vez que encontramos um ponto na situação a) o programa pára com um erro na aritmética; isto é um pouco como tentar fazer um mapa dos buracos nas ruas de Lisboa andando de automóvel pela cidade e marcando cada buraco à medida que caímos nele. É aqui que entra o trabalho teórico dos matemáticos garantindo que:

– se num dado passo dos cálculos o complexo **(x, y)** verificar a condição $x^2 + y^2 > 4$, então é inevitável que o processo vai acabar na situação a);

Isto permite-nos encontrar qualquer complexo nas condições a) muito antes de surgir qualquer erro na aritmética.

ALGORITMO DE CÁLCULO

O nosso programa vai fazer do monitor uma janela sobre uma região do plano complexo, marcando a negro os pontos b). Os outros pontos serão aqueles que ao fim de um certo número de iterações atingem a situação $x^2 + y^2 > 4$; podemos marcar estes pontos indistintamente a branco (o que corresponde à opção 'c') ou marcar cada ponto com uma cor dependente do número de iterações necessárias até que esta situação se revele (opção 'r'). Como os pontos na situação b) são aqueles para os quais o processo nunca pára, não vamos poder determiná-los exactamente; vamos, sim, marcar a negro os pontos que não passam para a situação a) ao fim de um certo número **it_max** de iterações do processo. Assim, dado um complexo **(x, y)**,temos o seguinte algoritmo:

```
n:=0; {n é o contador de iterações}
while (x*x+y*y <4) and (n<=it_max) do
  begin
      x_aux:=x*x-y*y+c_real; {o conteúdo de x é ainda necessário}
      y:=2*x*y+c_imag;
      x:=x_auc;    {actualiza-se o conteúdo de x}
      n:=n+1  {mais uma iteração}
  end;
```

se a opção for 'r', no final do ciclo colorimos o pixel que corresponde a **(x, y)** com a cor **0** se **n > it_max** e com a cor de código **n** se **n>it_max.** Para os leitores que queiram experimentar programas próprios, sugerimos que pensem noutros métodos para atribuir cor, segundo o valor de **n.**

Resta agora falar do modo como vamos estabelecer a correspondência entre os pixels do nosso ecrã gráfico e os pontos no plano complexo. Para isso é necessário conhecer as coordenadas dos vértices inferior esquerdo **(xmin, ymin)** e superior direito **(xmax, ymax)** da região rectangular que queremos observar. Se o nosso ecrã, no modo gráfico, tem pixels com coordenada horizontal que varia de **0** (à esquerda) até **umax** (à direita) e com coordenada vertical que varia de **0** (em cima) até **vmax** (em baixo), então ao pixel **(u,v)** corresponde o complexo **(x, y)** dado por:

```
x:=xmin + dh * u ;      y;= ymax - dv * v ;
```

onde **dh** e **dv** são dados por:

```
dh:= ( xmax - xmin) / umax ;
dv:= (ymax - ymin) / vmax ;
```

note-se que definimos um conjunto de **Julia** preenchido diferente para cada par de valores (**c_real, c_imag**).

O CONJUNTOD E MANDELBROT

O conjunto de **Mandelbrot** difere dos de **Julia** apenas no papel desempenhado pelo complexo (**x, Y**) correspondente ao pixel a marcar. Neste caso o processo começa sempre com o par (**0, 0**) sendo o papel que antes era desempenhado pelo par (**c_real, c_imag**) agora desempenhado pelo par (**x,y**). O conjunto de **Mandelbrot** vai ser calculado aproximadamente como sendo o conjunto dos pontos para os quais o processo iterativo não cresce demasiado durante as primeiras **it_max** iterações.

O método de atribuição de cor e a transformação de coordenadas do ecrã para o plano são os mesmos que foram utilizados para os conjuntos de **Julia.**

O PROGRAMA

O programa que acompanha este artigo foi escrito e compilado em Turbo Pascal 5, e implementa os algoritmos que desenham conjuntos de **Julia e Mandelbrot.** Estes algoritmos encontram-se nos procedimentos com os nomes dos conjuntos a desenhar e utilizam os dados recolhidos pelo procedimento **def_conjunto.** O procedimento **config_grafica** permite ao utilizador escolher a sua placa gráfica ou então seleccionar a resolução que considerar mais adequada.

O procedimento **desenhar** recolhe os dados necessários para gerar uma imagem e chama o procedimento **gravar**, se se pretender arquivar a imagem apresentada no ecrã. O procedimento **gravar** prepara um ficheiro de imagem, chamando o procedimento **captura_imagem** para gerar o seu conteúdo.

O procedimento **verimag** abre um ficheiro e, depois de verificar se se trata de um ficheiro gerado pelo próprio programa, selecciona e inicializa os gráficos. O procedimento **carregar** restaura a imagem a partir do ficheiro.

O bloco principal faz a escolha entre gerar uma nova imagem, ver uma imagem arquivada ou sair para o sistema operativo.

AVALIAÇÃO DE PERFORMANCE

Como o leitor irá verificar ao experimentar o programa, este é bastante lento quando comparado com o que acontece com a generalidade dos programas. Este problema deve-se, contudo, ao uso intensivo de operações com vírgula flutuante pelos algoritmos que definem os conjuntos. Cada iteração para ao conjunto **Mandelbrot,** por exemplo, exige cerca de 10 operações em vírgula flutuante. Se **it_max** for 20, o que não é muito, para cada ponto que esteja efectivamente no conjunto é necessário efectuar 200 operações. Para uma imagem com boa qualidade poderemos usar **it_max=100,** mas então serão necessárias 1000 operações antes de atribuir cor a um pixel correspondente ao conjunto de **Mandelbrot.** Compreende-se assim que se tivermos uma placa VGA com 640 x 480 = 307 200 pixels, uma imagem completa necessita de 307 200 acessos ao vídeo bem como de muitas dezenas de milhões de operações em vírgula flutuante, sem contar com as operações com inteiros. É, portanto, natural que estes algoritmos sejam usados para comparar e avaliar a velocidade das diversas máquinas. O algoritmo na forma em que foi apresentado poderia ainda ser melhorado para correr mais depressa, mas corríamos o risco de perder em clareza.

CONCLUSÃO

Para concluir deixamos ao leitor algumas sugestões de imagens a gerar para experimentar o programa. O conjunto de **Mandelbrot** é considrado pelos seus entusiastas como o objecto mais complexo de toda a matemática. O leitor tem assim à sua disposição um universo que apesar de já observado em algumas escalas e nalgumas zonas, tem ainda uma infinidade de imagens que nunca foram observadas por ninguém. Pode dizer-se o mesmo dos conjuntos de **Julia,** pelo que estas sugestões podem ser o ponto de partida para muitas explorações. Contudo existem algumas zonas menos interessantes e mesmo monótonas, pelo que é importante seleccionar ciudadosamente a região a observar antes de por o programa a executar cálculos muito demorados. Uma última sugestão: não se esqueça de executar o utilitário GRAPHICS antes de correr o programa, de modo a poder obter uma cópia impressa de qualquer imagem calculada.

> “ **O procedimento** desenhar **recolhe os dados necessários para gerar uma imagem e chama o procedimento** gravar, **se se pretender arquivar a imagem apresentada no ecrã** ”

SUGESTÕES

Para obter uma panorâmica geral do conjunto de **Mandelbrot** pode usar os valores:

```
xmin = -2.1    xmax = 0.8
ymin = -1.7    ymax = 1.7
```

outra região interesante do mesmo conjunto é:

```
xmin = -0.3    xmax = 0.1
ymin = 0.7     ymax = 1.2
```

ou então:

```
xmin = 0.2     xmax = 0.4
ymin = 0.47    ymax = 0.66
```

Relativamente aos conjuntos de Julia pode tentar:

```
c_real = -0.75 c_imag = 0.25
```

na região:

```
xmin = -1      xmax = 1
ymin = -1      ymax = 1
```

se tiver à mão um AT com monitor EGA (16 cores) e quiser esperar 2 horas, experimente calcular este conjunto com um máximo de 100 iterações. Na mesma região tente também:

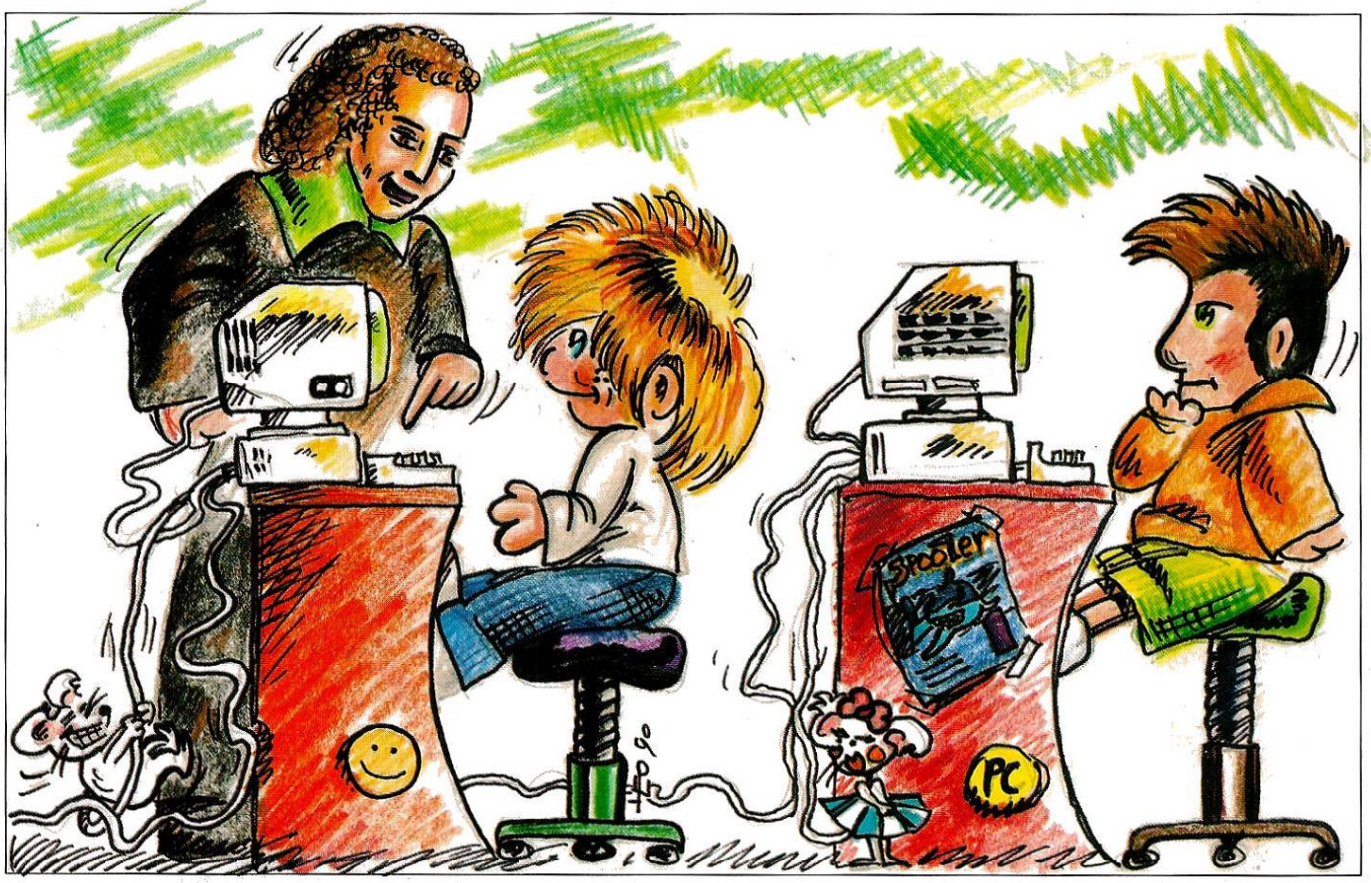
```
c_real = 0.288 c_imag = 0.48365
```

A partir destas (e de outras) imagens o leitor pode tentar ampliar sub-regiões particulares, de modo a ver os conjuntos com maior detalhe. Se em simultâneo for aumentanto o número de iterações, estes conjuntos continuarão a revelar-lhe a sua estrutura, sempre bela, complexa e surpreendente, qualquer que seja a escala em que a observe.

Carlos Albuquerque *

* *Departamento de Matemática da Faculdade de Ciências de Lisboa.*

PROGRAMAR EM PASCAL (parte I)



“A linguagem Pascal define um processo altamente estruturado de fazer programas para computador, sendo considerada muito justamente como a linguagem de programação mais didáctica”

BREVE INTRODUÇÃO

A linguagem Pascal define um processo altamente estruturado de fazer programas para computador, sendo considerada muito justamente como a linguagem de programação mais didáctica — o algoritmo de resolução de um problema é transposto para o programa através do uso de regras bem definidas, muito próximas dos próprios conceitos usados na elaboração do algoritmo. Nota-se actualmente, nos programadores, uma tendência generalizada pelo uso da linguagem “C”. Apesar disso, os conceitos de programação ministrados na maioria das universidades de engenharia ainda têm como base o Pascal, salientando o carácter pedagógico desta linguagem.

O Pascal é uma linguagem compilada — isto quer dizer que se escreve o programa com um editor de texto e depois se fornece ao compilador, para que este produza um ficheiro executável. Os compiladores de Pascal mais divulgados em Portugal são, inequivocamente, o da Borland (Turbo Pascal) e o da MicroSoft (MS-Pascal). Uma das grandes preocupações das *software houses* que criaram estes produtos foi a de respeitar as regras ISO (ISO=International Organization for Standardization) para a linguagem Pascal. Estas regras foram originalmente definidas pelos inventores da linguagem, os engenheiros K. Jensen e N. Wirth (não, não foi o sr. Pascal!!) e descritas numa obra dos mesmos autores, intitulada “Pascal User Manual and Report”. A sigla ISO designa uma organização mundial que se dedica exclusivamente a elaborar normas que são aplicadas a praticamente todas as áreas da engenharia e que, neste caso, estabelecem como *standard* da linguagem Pascal a definição feita pelos seus criadores.

Um programador que escreva um programa em Pascal respeitando estritamente as regras ISO tem a garantia de que o seu programa pode ser compilado sobre qualquer sistema operativo e com qualquer outro compilador — que também respeite estas regras — sem que seja necessário alterar sequer uma vírgula! O compilador MS-Pascal segue rigorosamente a definição ISO, pondo ainda à disposição do programador numerosas funções, consideradas extensão — é claro que acerca destas *extended functions* não há garantia que funcionem com outros compiladores. O compilador Turbo Pascal não segue com rigor a definição ISO, apresentando algumas diferenças de sintaxe mesmo em instruções básicas; no entanto, os programadores que usem esta *package* podem beneficiar

do já famoso "ambiente Borland" no desenvolvimento dos seus programas (com o editor, compilador e *debugger* integrados), além de várias livrarias de funções já implementadas, constituindo um enorme auxílio para o programador.

Do que acima se disse poderemos tirar as seguintes conclusões: se o importante no *software* for afportabilidade, o compilador a escolher deverá ser o MS-Pascal e os programas não deverão usar quaisquer funções além das *standard*; por outro lado, se os programas não se destinarem a ser usados fora do ambiente MS-DOS, então a opção deverá recair sobre o Turbo Pascal, já que as imensas facilidades postas à disposição do programador lhe permitirão desenvolver melhores programas em menos tempo.

A revista **Spooler** vai falar de Pascal nos seus próximos quatro números, sendo este que o leitor tem presentemente em mãos o primeiro dos quatro. Vamos partir do zero e tentar abordar todos os temas ligados a esta linguagem. Em todos os números apresentaremos exemplos ilustrando as questões abordadas e incluiremos esses mesmos exemplos na disquete; vamos ainda desenvolver um pequeno jogo, que sofrerá evoluções em todos os números de forma a integrar em cada versão as novas instruções tratadas.

SUMÁRIO

- **Estrutura de um programa. Variáveis e constantes. Afectação.**
- **Expressões e Operadores. INPUT/OUTPUT: leitura do teclado e escrita no ecrã.**
- **Instruções de controlo do programa: IF e CASE. A instrução GOTO.**

ESTRUTURA GERAL DE UM PROGRAMA EM PASCAL

A estrutura geral de um programa em Pascal define-se basicamente em três partes:

– um cabeçalho que identifica o programa através de um nome e define os parâmetros do programa;

– uma zona de declarações para constantes, variáveis, *labels*, tipos definidos pelo programador, funções e procedimentos; algumas destas declarações são opcionais;

– um corpo de programa, delimitado pelas palavras **BEGIN** e **END**, e terminado por um ponto final; o ponto final "." indica ao compilador o final do programa.

```
{Cabeçalho do programa}
PROGRAM UM (INPUT,OUTPUT);

{Zona de declarações}
VAR NOME:STRING;

{Corpo do programa}
BEGIN
  WRITELN('Como te chamas ? ');
  READLN(NOME);
  WRITELN('Olá ',NOME,'!!');
END.
```

IDENTIFICADORES

Identificadores são os nomes atribuídos pelo programador aos diversos elementos constituintes de um programa em Pascal: constantes, variáveis, procedimentos, funções, etc. Os identificadores têm de ser obrigatoriamente iniciados por uma letra na gama 'A'..'Z'.

Depois, podem ser seguidos por qualquer combinação de letras, dígitos ou sublinhado (*underscore*), até ao máximo de 63 caracteres. Os compiladores de Pascal não fazem distinção entre maiúsculas e minúsculas nos identificadores: **nome** e **Nome** são considerados o mesmo identificador.

DECLARAÇÃO DE VARIÁVEIS

A sintaxe é a seguinte:

VAR <identificador>:<tipo>;
ou
VAR <lista de identificadores>:<tipo>;

A palavra chave VAR indica ao compilador que se vai proceder à declaração de variáveis. De seguida, indica-se a variável ou variáveis a declarar; os dois pontos ":" têm aqui a função de separador entre a lista dos identificadores e o tipo. O tipo indica o género de dados que se pretende colocar na variável, para que o compilador possa reservar o espaço adequado a cada situação. No caso do exemplo já apresentado, declarou-se a variável **NOME** como sendo uma **STRING**, ou seja, um conjunto de caracteres constituindo uma linha de texto.

> **"Identificadores são os nomes atribuídos pelo programador aos diversos elementos constituintes de um programa em Pascal"**

TIPOS PREDEFINIDOS DE VARIÁVEIS

	Identificador do tipo	Espaço reservado pelo compilador	Gama de valor aceite
STANDARD	BYTE	1 byte	0..255
	WORD	2 bytes	0..65535
	INTEGER	2 bytes	-32768..32767
	REAL	6 bytes	2.9E-39..1.7E+38
	CHAR	1 byte	ASCII (0..255)0
	BOOLEAN	1 byte	TRUE, FALSE
	STRING	1 a 256 bytes	'ajhg65271NN nmc...'
EXTENSÃO TURBO PASCAL	SHORTINT	1 byte	128..127
	LONGINT	4 bytes	-2147483648..-2147483647
	COMP	8 bytes	9.2E18..9.2E18
	SINGLE	4 bytes	1.5E-39..1.7E38
	DOUBLE	8 bytes	5.0E-45..3.4E38
	EXTENDED	10 bytes	3.4E-4932..1.1E4932
EXTENSÃO MS-PASCAL	INTEGER4	4 bytes	
	REAL4	4 bytes	
	REAL8	8 bytes	

Os tipos BYTE, WORD, INTEGER e REAL são tipos numéricos. O tipo CHAR recebe caracteres de toda a gama ASCII, entre o caracter de código 0 e o caracter de código 255. O tipo BOOLEAN é um tipo lógico, recebendo os valores TRUE (verdadeiro) e FALSE (falso). Por último, o tipo STRING recebe conjuntos de caracteres, podendo a sua dimensão variar de forma dinâmica. As strings possuem algumas características interessantes, que discutiremos na segunda parte deste artigo (ou seja, no próximo mês). O Turbo Pascal e o MS-Pascal apresentam extensões aos tipos INTEGER e REAL, com diferentes capacidades de armazenamento de valores: o Turbo Pascal apresenta os tipos SHORTINT, LONGINT e COMP como extensões ao tipo INTEGER e os tipos SINGLE, DOUBLE e EXTENDED como extensões ao tipo REAL. No caso do MS-Pascal, o INTEGER4 é equivalente ao LONGINT do Turbo Pascal e os tipos REAL4 e REAL8 equivalentes, respectivamente, aos tipos SINGLE e DOUBLE.

Exemplos de declaração de variáveis de diferentes tipos:

> “Uma constante é um valor conhecido antes de iniciar o programa e que não muda durante a sua execução”

```
VAR RESPOSTA:BOOLEAN;
    A,B,C:INTEGER;
    Z:CHAR;
    NOME_1:STRING;
```

Como pode ver-se, não é necessário colocar a palavra VAR mais de uma vez (apesar de ser permitido colocá-la uma vez por cada linha de declaração) e as variáveis de um mesmo tipo podem ser colocadas em lista, separadas por vírgulas.

CONSTANTES

Uma constante é um valor conhecido antes de iniciar o programa e que não muda durante a sua execução. A zona de declaração de constantes é definida pela palavra chave CONST. Por exemplo, num programa que efectua o cálculo do perímetro duma circunferência, podemos definir a constante PI como sendo o valor 3.1416.

```
PROGRAM DOIS (INPUT,OUTPUT);
    CONST PI=3.1416;
          RAIO=10;
    VAR PERIMETRO:REAL;
    BEGIN
        PERIMETRO:=2*PI*RAIO;
        WRITELN(PERIMETRO);
    END.
```

As constantes podem ser de tipo numérico, lógico ou caracter. Por isso, são consideradas válidas as seguintes declarações de constantes:

```
CONST  MES='DEZEMBRO';
       SIM=TRUE;
       NAO=FALSE;
       LETRA='a';
       NUMERO_REAL=10.5;
       NUMERO_INTERIRO=-33;
```

AFECTAÇÃO DE VARIÁVEIS

A afectação de uma variável com um dado conteúdo faz-se com o operador ‘’:=’’. Exemplos:

```
VAR RESPOSTA:BOOLEAN;
 A:INTEGER;
 NOME_1:STRING;
 Z:CHAR;
BEGIN
 RESPOSTA:=TRUE;
 A:=10;
 NOME_1:='SPOOLER Magazine';
 Z:='*';
END.
```

Tal como é mostrado no exemplo, uma variável só pode ser afectada com um conteúdo do tipo para a qual foi definida. Serão, por isso, erradas as afectações do tipo A:='xis' ou RESPOSTA:=10. Note-se ainda que uma *string* é uma sequência de caracteres delimitados por plicas (caracter’’’’’).

INICIALIZAÇÃO DE VARIÁVEIS NO ACTO DE DECLARAÇÃO

O acto de declarar uma variável não determina, por si só, o seu conteúdo inicial. Uma das formas de resolver o problema é efectuar as respectivas inicializações no topo do programa; outra forma é indicar ao compilador qual o valor a colocar nas variáveis no acto da sua criação – note-se que com este segundo processo, é o compilador que inicializa as variáveis, ao passo que no primeiro é o programa que, ao ser executado, coloca os valores, tornando-se por isso mais extenso e mais lento. Para que o programador possa indicar o valor inicial que a variável deve ter, ambos os compiladores referidos definem uma estratégia a seguir.

No caso do MS-Pascal, usa-se a palavra chave VALUE para esse fim:

```
PROGRAM TRES_A (INPUT,OUTPUT);
  VAR A, B:INTEGER;
      C, D:REAL;
 VALUE A:=10;
       B:=100;
       C:=10.5;
 BEGIN
       D:=A*10+B+C;
       WRITELN(D);
       A:=A*2;
       WRITELN(A);
 END.
```

No caso do Turbo Pascal, esta afectação é feita de forma muito *sui generis*, pois as variáveis são declaradas na área das constantes – apesar disto, são verdadeiras variáveis, cujo valor pode ser alterado no decorrer do programa. Vejamos:

```
PROGRAM TRES_B (INPUT,OUTPUT);
  CONST A:INTEGER=10;
        B:INTEGER=10;
        C:REAL=10.5;
  VAR   D:REAL;
  BEGIN
        D:=A*10+B+C;
        WRITELN(D);
        A:=A*2;
        WRITELN(A);
  END.
```

EXPRESSÕES E OPERADORES

Uma expressão consiste numa fórmula que, depois de avaliada pelo computador, produz um resultado. A expressão é constituída por operadores (que indicam as acções a efectuar) e por operandos (valores sobre os quais são efectuadas as acções). Em Pascal existem três tipos fundamentais de operadores:

Aritméticos

+	Adição
-	Subtracção
*	Produto
/	Quociente real
DIV	Quociente inteiro
MOD	Resto do quociente inteiro

Relacionais

=	Igual
<>	Diferente
>	Maior
>=	Maior ou igual
<	Menor
<=	Menor ou igual

Lógicos

AND	E
OR	Ou
NOT	Não

Os resultados das expressões dependem fundamentalmente dos operadores: uma operação aritmética devolve um resultado numérico e uma operação relacional ou lógica produz um resultado lógico. Exemplos:

10+5 ..... O resultado é 15 (inteiro)
10.5+5..... O resultado é 15.5 (real)
10>5 ..... O resultado é TRUE (lógico)

Sobre as expressões em Pascal falta ainda salientar três pormenores:

– Uma operação aritmética em que um dos operandos seja real produz um resultado do tipo real (a 6 bytes); por esta razão, se eventualmente se pretender guardar este resultado numa variável, terá de ser numa variável declarada como REAL. Entre números reais, a divisão é obrigatoriamente efectuada com o operador ''/''.

15.4/4 .... O resultado é 3.85 (real)
15/4..... O resultado é 3.75 (real, apesar de os operandos serem inteiros)

– No caso de as operações entre números inteiros, há que ter em atenção que a divisão inteira se efectua com o operador DIV. Existe ainda uma operação extra entre números inteiros, que é o resto da divisão inteira, através do operador MOD.

15 DIV 4..... O resultado é 3 (inteiro)
15 MOD 4 .... O resultado é 3 (inteiro)

– Uma operação lógica que envolva dois ou mais operandos que sejam eles próprios expressões relacionais terá de ser obrigatoriamente parentesiada.

```
((A>=10) AND (XIS<15.5)) OR (RESULTADO=10)
```

Os parêntesis usam-se também quando se deseja alterar a prioridade de avaliação dos operadores.

12*5+2 ..... O resultado é (12*5)+2=60+2=62
12*(5+2)..... O resultado é 12*(5+2)=12*7=84

LEITURA DE DADOS

A leitura de dados do teclado faz-se através da instrução READLN, como é indicado:

```
READLN(<identificador>);
READLN(<lista de identificadores>);
```

O identificador refere a variável dentro da qual serão colocados os dados lidos do teclado. Em variáveis do tipo STRING podem ser colocados quaisquer caracteres do teclado; em variáveis numéricas só podem ser colocados dígitos, eventualmente antecedidos pelos sinais ''+'' ou ''-''. A instrução READLN não pode ser efectuada sobre variáveis do tipo BOOLEAN.

```
PROGRAM QUATRO(INPUT,OUTPUT);
   VAR    A:INTEGER;
          B:STRING;
          X,Y,Z:REAL;
   BEGIN
      READLN(A);
      READLN(B);
      READLN(X,Y,Z);
   END.
```

Existe também uma outra instrução de leitura de dados do teclado (READ) com sintaxe idêntica ao READLN. Apesar de ser permitida a sua utilização, não deve ser empregue, pois o seu funcionamento não é adequado para a leitura do teclado. Passemos a explicar: a instrução READLN lê caracteres do teclado até receber uma mudança de linha (*carriage return*); a mudança de linha é produzida pela tecla ENTER. Após a recepção do ENTER, a instrução READLN afecta a respectiva variável e limpa totalmente o *buffer* do teclado. A instrução READ tem funcionamento semelhante, mas ao limpar o buffer do teclado não retira de lá o carácter de mudança de linha. Como resultado disto, a próxima instrução READ ou READLN que for efectuada já encontra um carácter no buffer, que utiliza (incorrectamente) como se tivesse sido um ENTER dado pelo utilizador. A situação piora definitivamente se o programa usar apenas READs, pois após a primeira leitura do teclado o buffer ficará com um carriage return que nunca mais será retirado, fazendo com que os READs deixem positivamente de funcionar! Como nota de curiosidade fica a indicação de que a instrução que se coloca no cabeçalho do programa após o identificador do nome (INPUT,OUTPUT) serve para fazer saber ao compilador que se pretendem usar os ficheiros de STANDARD INPUT (o teclado) e STANDARD OUTPUT (o ecrã); no Turbo Pascal esta indicação é opcional.

“Os resultados das expressões dependem fundamentalmente dos operadores: uma operação aritmética devolve um resultado numérico e uma operação relacional ou lógica produz um resultado lógico”

ESCRITA DE DADOS

A escrita de dados no ecrã faz-se com as instruções WRITE e WRITELN:

```
WRITE(<identificador>);
WRITE(<constante>);
WRITE(<lista de identificadores e/ou constantes>);
WRITELN(<identificador>);
WRITELN(<constante>);
WRITELN(<lista de identificadores e/ou constantes>);
```

A diferença entre o WRITE e o WRITELN consiste no facto de o WRITELN escrever no ecrã com mudança de linha e de o WRITE não o fazer. Apresentam-se, de seguida, diversos exemplos válidos de utilização destas instruções:

```
WRITE('O nome é ',NOME,' e a morada é ',MORADA);
WRITELN(); {apenas muda de linha}
WRITELN(10);
WRITELN('RESULTADO=',(11+A)*500);
```

CONTROLO DE FLUXO

O controlo de um programa é definido por instruções que testam condições e seleccionam para execução um ou outro bloco de instruções em alternativa. Cabe neste local explicar que, quando se pretende executar um grupo de instruções como se fosse uma única, se ''encapsula'' o grupo de instruções com as palavras BEGIN (no início) e END (no fim). Desta forma, o bloco assim delimitado passa a ser encarado como uma única instrução e executado como tal. O Pascal coloca à disposição dos programadores as instruções IF e CASE para controlo de fluxo.

```
IF   <expressão> THEN
        <instrução executada no
        caso da expressão ser  TRUE>
[ELSE
        <instrução executada no
        caso da expressão ser FALSE>]
```

A expressão, ao ser avaliada, produz um resultado booleano – TRUE ou FALSE – que o IF vai testar para decidir qual das duas acções vai executar. Um IF não tem obrigatoriamente associado um ELSE; se não existir o

“Para aplicar os conhecimentos de programação em Pascal abordados neste número da Spooler, elaborámos um pequeno programa...”

ELSE e a expressão for falsa, o IF simplesmente não executa qualquer acção e o programa continua. Saliente-se ainda o facto de a instrução que precede o ELSE nunca possuir ponto-e-vírgula.

Exemplo:

```
IF X>=0 THEN
    WRITELN ('É número positivo.')
ELSE
    WRITELN ('É número negativo.');
```

Exemplo:

```
IF TECLA='1' THEN
   BEGIN
      WRITELN ('Premiu a tecla 1');
      WRITELN ('Agora outra ...');
      READLN (TECLA);
      WRITELN ('Premiu a tecla ',TECLA);
      WRITELN ('*** OBRIGADO ***');
   END;
WRITELN ('Fim');
```

Como se verifica no segundo exemplo, se a condição TECLA='1' for falsa, o programa avança imediatamente para a instrução que imprime a palavra ''Fim'' sem executar qualquer das outras acções intermédias.

```
CASE  <expressão> OF
      <valor1>: <acção1>
      <valor2>: <acção2>
       ...
      <valorN>: <acçãoN>
      OTHERWISE <acção por defeito>
END;
```

A instrução CASE possui uma expressão que é avaliada e uma lista de alternativas, das quais apenas uma será escolhida, em função do resultado.

Se a expressão tiver o **valor1**, é executada APENAS a **acção1**; se a expressão tiver o **valor2**, é executada APENAS a **acção2**; etc ...

Se a expressão produzir um valor que não esteja previsto na lista, será efectuada a acção associada ao ramo OTHERWISE.

É de notar que o compilador Turbo Pascal não reconhece a palavra chave OTHERWISE, sendo usada a palavra ELSE em sua substituição.

Exemplo da utilização do CASE:

```
PROGRAM CINCO (INPUT,OUTPUT);
   VAR TECLA:CHAR;
   BEGIN
      WRITELN ('Premir uma tecla 0..9 —>');
      READLN (TECLA);
      CASE TECLA OF
         '0': WRITELN ('Zero');
         '1': WRITELN ('Um');
         '2': WRITELN ('Dois');
         '3': WRITELN ('Três');
         '4': WRITELN ('Quatro');
         '5': WRITELN ('Cinco');
         '6': WRITELN ('Seis');
         '7': WRITELN ('Sete');
         '8': WRITELN ('Oito');
         '9': WRITELN ('Nove');
      ELSE WRITELN ('Não premiu uma tecla válida!');
   END;
END.
```

Existe ainda uma outra instrução que permite enviar o programa para uma localização determinada: é a instrução **GOTO <label>**. A *label* (etiqueta) tem de ser definida na zona de declarações do programa pela palavra chave LABEL. Esta não é considerada uma verdadeira instrução de controlo de fluxo, porque não permite decisão, tratando-se apenas de um salto incondicional (mas que altera a sequência normal do programa!).

```
PROGRAM SEIS (INPUT,OUTPUT);
  LABEL INICIO;
  VAR A:STRING;
  BEGIN
     INICIO:
     WRITELN ('Escreva o seu nome:');
     READLN (A);
     IF (A='fim') THEN
        WRITELN ('Adeuzinho ...')
     ELSE
        BEGIN
           WRITELN ('Olá ',A);
           GOTO INICIO;
        END;
  END.
```

Apesar de neste exemplo a utilização da instrução GOTO ser ilustrada com a implementação de um ciclo de repetição, ela não é usada para este fim (para isso há instruções apropriadas); regra geral, o GOTO usa-se para interromper o curso normal do programa e enviá-lo para para um local predeterminado (por exemplo, devido à ocorrência de um erro).

APLICAÇÃO PRÁTICA DOS CONHECIMENTOS

Para aplicar os conhecimentos de programação em Pascal abordados neste número da **Spooler**, elaborámos um pequeno programa que (como já foi dito) iremos desenvolvendo nos próximos números, à medida que formos ''avançando na matéria''...

VERSÃO BASE: Jogo extremamente simples para dois jogadores, em que o primeiro insere um número numa gama pré-estabelecida e o segundo vai ''dando palpites'' até acertar. No final, o programa indica quantas tentativas foram feitas.

Por fim, para que se possa considerar o assunto encerrado por hoje, falta dizer que, num programa em Pascal, todo o texto que se encontrar entre chavetas ''{ blá, blá, blá }'' é ignorado pelo compilador – usa-se esta facilidade para adicionar comentários ao texto do programa; em vez das chavetas pode usar-se o par de parentesis curvo e asterisco ''(* mais blá, blá, blá *)'' com os mesmos resultados práticos.

SUMÁRIO DO PROXIMO MÊS

Declaração e utilização de tipos não *standard*. Tipos *enumerated* e *subrange*. Tipos ordinais. Conjuntos. Ciclos de repetição: FOR, REPEAT e WHILE. *Arrays*.

Nova versão do jogo (recorrendo às novas instruções) que passará a permitir duas opções: ser o computador a sortear o número a adivinhar e alterar o grau de dificuldade.

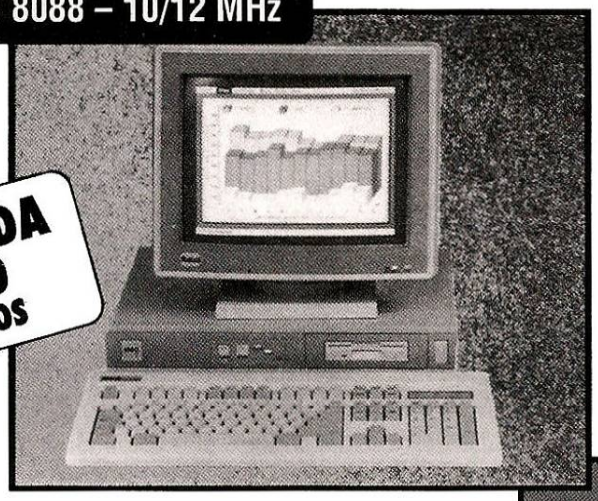
Paulo Bernardo

abc – INFORMÁTICA, LDA.

Rua dos Sapateiros, 160 - 2.º – 1000 LISBOA
Telefs: 32 50 70-347 59 40 – Telex: ABCINF 13348 P – Fax: 346 22 40

CAMPANHA DE VERÃO

ABC 8088 – 10/12 MHz



PARA REVENDA E ENSINO CONSULTE-NOS

ABC 8088

- Compatível PC XT
- CPU 8088-1 (CPU V20)
- 4,77/10 (12) MHz

➥ **12 Mhz – 640 Kb RAM exp. até 1 Mb**

- Drives de 5 ¼" ou 3 ½"
- Portas série e paralelo
- Carta monocromática
- Teclado 102 teclas
- Monitor monocromático

PREÇO: ABC 8088 – 12 MHz ..... 125 000$00*

ABC 80286 – 12/16 MHz

REDES UNIX e XENIX CONTE CONNOSCO SOMOS ESPECIALISTAS

ABC 80286

- Compatível PC AT
- CPU 80286 – 12/16 MHz

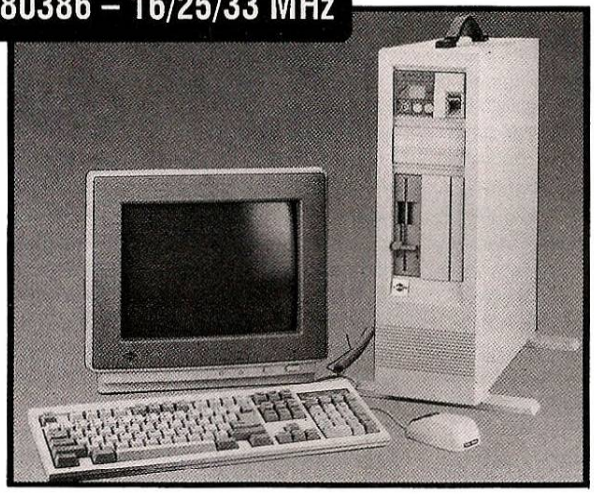
➥ **1 Mb RAM**

- Drives de 5 ¼" ou 3 ½" – Alta densidade
- Portas série e paralelo
- Relógio em tempo real / calendário / CMOS W / Bateria
- Carta monocromática (VGA opcional)
- Teclado 102 teclas
- Monitor monocromático

DISCOS E CONSUMÍVEIS AOS MELHORES PREÇOS DO MERCADO

PREÇO: ABC 80286 – 12 Mhz ..... 175 000$00*
– 16 MHz ..... 199 000$00*

ABC 80386 – 16/25/33 MHz



ABC 80386

- Compatível PC AT
- CPU INTEL 80386 – 16/25/33 MHz

➥ **1 Mb RAM expansíveis até 8 Mb**

- Drives de 5 ¼" ou 3 ½" – Alta densidade
- Portas série e paralelo
- Relógio em tempo real / calendário / CMOS W / Bateria
- Carta monocromática (VGA opcional)
- Teclado 102 teclas
- Monitor monocromático

PREÇO: ABC 80386 – 16 Mhz ..... 265 000$00*
– 25 Mhz ..... 365 000$00*
– 33 Mhz ..... 610 000$00*

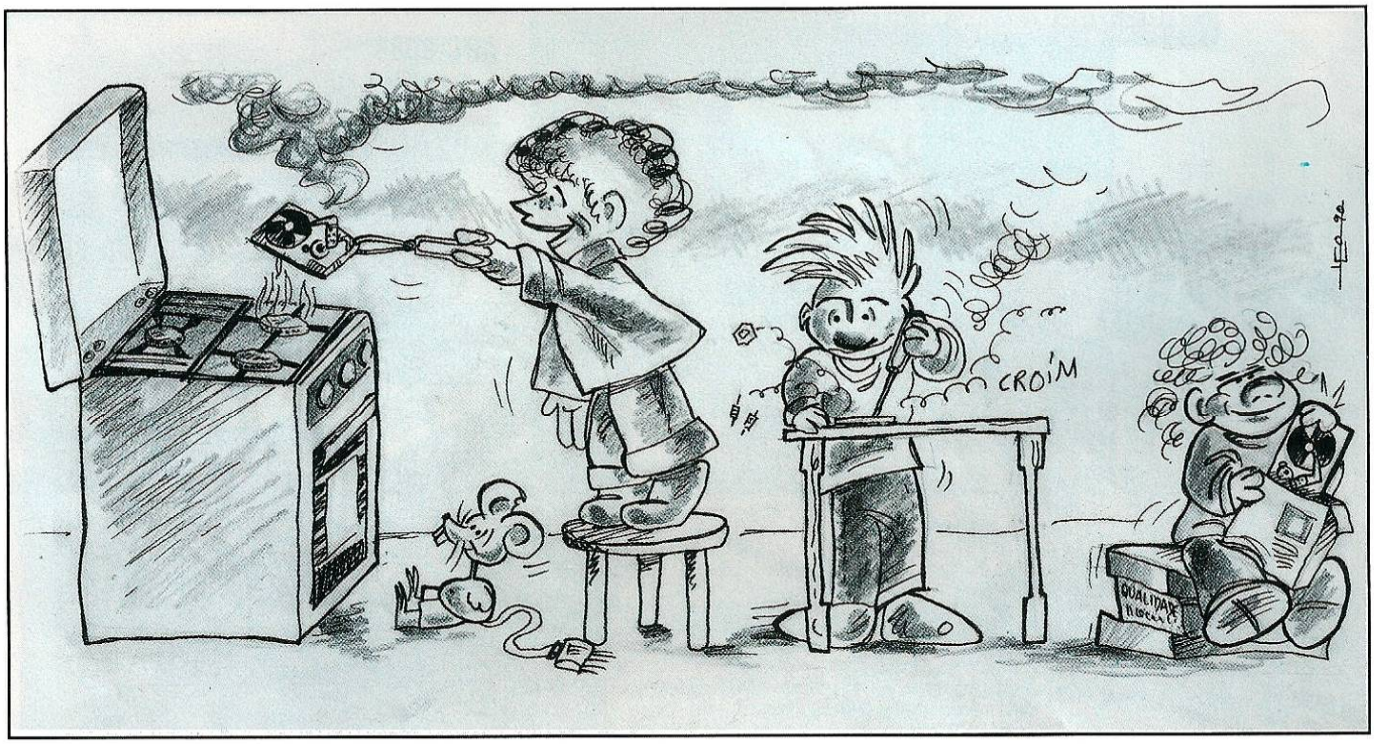
**Aos preços indicados serão acrescidos 17% IVA*

FACILIDADES DE PAGAMENTO

abc AGENTES:

LISBOA: Rua da Assunção, 67 – Telf: 32 46 47
BRAGA: Avenida Central, 85 - 1º – Telfs: 72 798-74 369
S. JOÃO ESTORIL: Av. Florinda Leal, Lote 1-A Telef: 267 07 33
VISEU: Rua Direita, 77 – Telf: 27 664
PORTIMÃO: Rua D. Carlos I – Telfs: 83 653-85 670
SETÚBAL: Largo da Misericórdia, 28 – Telf: 31 432

DISCOS RÍGIDOS – A REPARAÇÃO



“Os cuidados com o disco iniciam-se logo com a chegada à oficina onde são empregues processos de completa protecção electrostática”

No número anterior e na primeira parte deste artigo ''falámos'' e chegámos à conclusão que o actual nível de *performance* dos sistemas computorizados obriga a uma grande capacidade de memória de massa, o que levou ao desenvolvimento de um mecanismo conhecido como *winchester disk* ou mais vulgarmente, ''disco rígido''. O disco rígido (como explicado no número anterior) é uma combinação de elementos mecânicos, electrónicos e magnéticos extremamente sofisticados. Devido a estas razões tem de ser construído com altos padrões de qualidade e num ambiente extremamente limpo.

Como muitos dos mecanismos feitos pelo homem, não é perfeito, pelo que ocasionalmente falha. Quando tal acontece (e nada do que foi aconselhado no número anterior resulta), o disco ou é deitado para o lixo ou é reparado num centro especializado onde as condições de reparação se assemelham às utilizadas na produção original. É claro que estas condições vão desde electrónica extremamente avançada até às condições de purificação do ambiente onde se trabalha (pelas razões anteriormente explicadas).

Para melhor se ficar a conhecer as condições de reparação dos discos rígidos vamos aqui descrever todo o processo de reparação utilizado por uma''oficina'' de discos rígidos recém-instalada no nosso País, cujo nome não mencionamos por razões de isenção publicitária.

Todo o processo de reparação e garantia de qualidade (testes) está descrito na fig. 1 que poderá, durante a leitura desta ''conversa'', ser consultada sempre que necessário pois descreve, ''por alto'', tudo o que vai ser dito nas próximas linhas.

A REPARAÇÃO

Os cuidados com o disco iniciam-se logo com a chegada à oficina onde são empregues processos de completa protecção electrostática. Nesta fase é efectuada a pesquisa de estragos, tanto no contentor em que o disco foi transportado, como no próprio disco. A documentação é verificada, e a qualidade do empacotamento, examinada, para que possa ser utilizada no reenvio após a reparação do disco.

Nos centros de reparação onde se trabalha com grande número de discos ao mesmo tempo torna-se bastante compensador (diríamos mesmo que indispensável) a vocacionalização do sistema informático para recolha e lançamento de informação através de código de barras. Assim, a cada disco que chega ao centro é atribuída uma folha-de-obra com um código de barras, que o acompanhará através de todo o processo de reparação e onde serão registados todos os testes e problemas detectados durante a sua ''estadia''. A vantagem mais evidente da utilização do código de barras (apesar de haver muitas outras) é a possibilidade de em qualquer altura se poder localizar o disco pretendido no meio de milhares e,consequentemente, saber-se em que fase da reparação este se encontra.

Quando a fase da reparação propriamente dita se inicia, a primeira coisa que se faz é um teste denominado ''teste básico'' e que tem como fim determinar a área geral da avaria, que tanto pode ser mecânica como electrónica, ou ambas.

Se um erro mecânico ou HDA – ''Head Disk Assem-

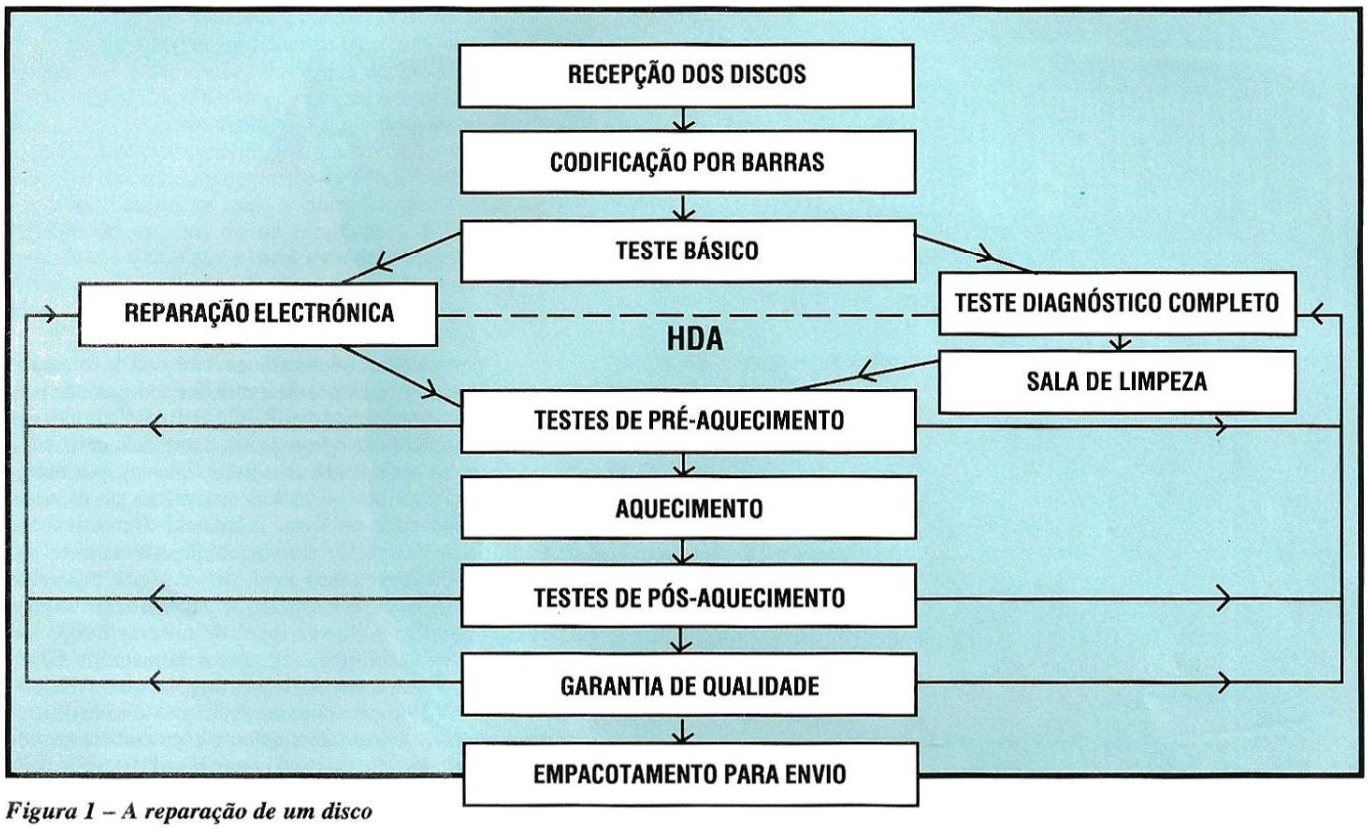


Figura 1 – A reparação de um disco

bly'' (montagem da cabeça de disco) for diagnosticado no teste base, então é levado a cabo um teste de diagnóstico completo, onde são necessários avançados meios tecnológicos para diagnosticar a avaria na cabeça do disco – que está hermeticamente fechada. As recomendações da técnica a utilizar são postas na folha-de-obra e, de seguida, o disco passa para as salas de limpeza.

Depois do teste básico é feito um diagnóstico completo e, após a reparação, é feito um teste de pré-aquecimento. O teste que antecede o aquecimento e o teste que o segue confirmarão se o disco está a trabalhar correctamente, tanto à temperatura ambiente como a temperaturas elevadas obtidas com a operação de aquecimento.

Se o erro detectado é de natureza electrónica, então o disco é conduzido para um departamento especializado nesta reparação. A reparação é feita a todos os níveis incluindo o ''Surface Mount Device''-SMD (superfície gravada). De seguida é realizada uma série de testes e, se algum disco falhar num desses testes, é reenviado ao departamento de reparação electrónica para ser, de novo, submetido a todo o processo até passar em todos os testes com sucesso.

Já foi feita referência à necessidade de extrema limpeza do ar que está hermeticamente selado dentro dos discos. Assim, é indispensável trabalhar em salas onde o ar seja purificado e filtrado, pelo que, nessas salas, a pureza do ar é de 100 partículas por pé cúbico e, por cima de cada mesa de trabalho, o ar condicionado envia ar limpo a menos de 10 partículas por pé cúbico. O nível de pureza do ar, tanto dentro da sala como no disco selado é medido regularmente por um medidor de partículas. Cada sala de limpeza tem o seu próprio equipamento de produção de ar limpo, o que assegura a continuidade das operações de reparação. Não é normalmente considerado o facto de o corpo humano libertar (em andamento) 5 milhões de partículas por minuto. É também grande o número de partículas libertadas pelo vestuário. Por esta razão todos os operadores dentro da sala de limpeza têm que usar um fato de protecção que cubra totalmente o corpo, com excepção dos olhos. Também são usadas luvas extremamente finas que asseguram ao operador a sensibilidade necessária às operações delicadas requeridas para montar e desmontar o disco. Não é permitido o uso de cosméticos dentro da sala.

Nas salas de aquecimento os discos são testados por um período entre 24 e 48 horas, de acordo com a especificação do fabricante. Este teste é realizado por cada reparação efectuada, sendo muito rigoroso e contribuindo bastante para a confiança do cliente quando o disco lhe é devolvido. Por vezes a elevação da temperatura dentro da sala de aquecimento iguala a temperatura existente nas condições normais de funcionamento na utilização normal.

Depois de completados todos os passos de reparação e verificação anteriormente descritos, ainda passa pelo departamento de controlo de qualidade onde cada disco deve obedecer aos padrões em vigor. Além de todos estes controlos de de qualidade, um determinado número de discos será ainda seleccionado por amostragem e submetido a um teste operacional completo.

A última etapa da reparação consiste no empacotamento e envio do disco aos seus proprietários. No entanto, para garantir um padrão de qualidade ainda mais elevado, alguns dos discos serão ainda desempacotados e enviados de novo para o departamento de controlo de qualidade. Esta derradeira fase é um teste funcional para garantir uma alta qualidade na reparação dos discos.

E, basicamente (sem entrar em grandes pormenores técnicos), são estes os passos mais importantes dados por um disco depois de entrar num centro especializado de reparação.

Praticamente todo o tipo de discos pode ser reparado, tanto no aspecto da capacidade como no de marca.

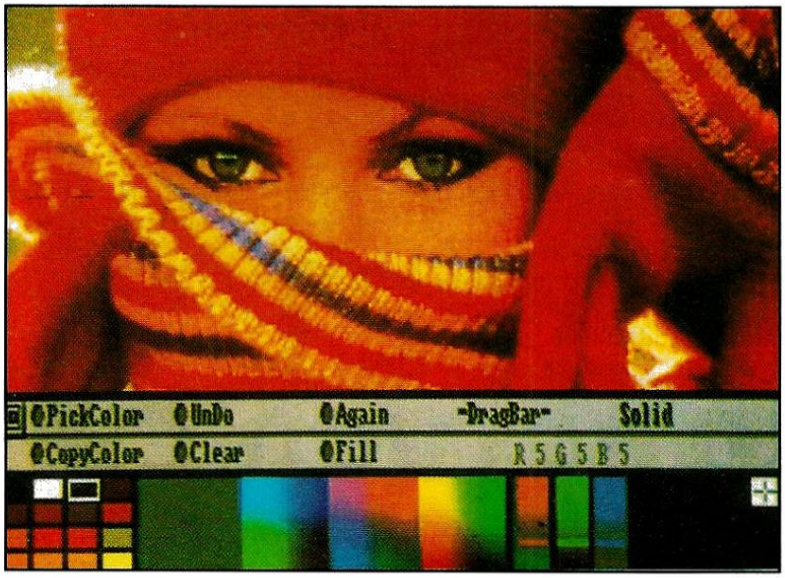
Por agora nada mais há a dizer sobre técnica de reparação de discos. No entanto o leitor já sabe que a Spooler está sempre em cima do acontecimento, pelo que se alguma novidade surgir por aí a ''besourar'' os nossos atentos ouvidos, cá estaremos para a transmitir.

Vítor Guerreiro.

“Praticamente todo o tipo de discos pode ser reparado, tanto no aspecto da capacidade como no de marca”

AMIGA

(II)



Depois da descrição que demos no último número da **Spooler** sobre o *hardware* do Amiga, chega a vez do *software*. Um bom aproveitamento das possibilidades gráficas dadas pelo COPPER e BLITTER, num ambiente de trabalho moderno, baseado em janelas e ícones, requer um sistema operativo sofisticado, e vamos ver que o Amiga OS é, efectivamente, um dos sistemas operativos mais sofisticados oferecidos em microcomputador.

“No sentido clássico do termo, o Amiga não tem, de facto, um sistema operativo, mas antes um conjunto de “softwares de sistema” que cooperam na prestação de serviços”

NÃO TEM SISTEMA OPERATIVO?

No sentido clássico do termo, o Amiga não tem, de facto, um sistema operativo, mas antes um conjunto de ''softwares de sistema'' que cooperam na prestação de serviços. A estrutura do software de sistema é perfeitamente hierárquica. Sem entrar em detalhes que apenas podem interessar ao programador, podemos dizer que o nível mais básico do sistema se encontra praticamente junto do hardware. São os ''device drivers'', conjuntos de rotinas encarregadas de gerir um determinado recurso, de o virtualizar do nível do hardware para o do software.

Veremos mais pormenores adiante. Passemos antes à espinha dorsal do sistema, a EXEC.LIBRARY. Esta colecção de rotinas faz a gestão básica do tempo de processador, permitindo o famoso *multitasking* do Amiga.

É na EXEC.LIBRARY que se encontra o único endereço garantidamente fixo do sistema. Esse endereço é a âncora através da qual se conhecem todas as características de sistema e se acedem todos os serviços. Porque o Amiga está definido como um sistema dinâmico: desde o início da sua concepção que se excluíram coisas como tabelas, dado que mais tarde ou mais cedo estas se iriam revelar de dimensões inadequadas aos sistemas. Optou-se, como estrutura de dados básica, pela lista ''linkada''. Esta estrutura de dados é, por definição, extensível, ocupando espaço na memória proporcionalmente aos dados requeridos. A relativa inconveniência de forçar um acesso linear, elemento a elemento, foi compensada pela provisão, sempre a nível da EXEC.LIBRARY, de todas as rotinas básicas para operar sobre listas linkadas, numa forma muito optimizada. São listas linkadas, por exemplo, a lista de librarys do sistema; a lista de áreas de memória ocupada ou livre; a lista de dispositivos de memória de massa; de volumes lógicos montados; das fontes de caracteres para ecrã; das tarefas e processos suspensos, activos ou a activar; de janelas, ecrãs, ícones, e assim por diante. As vantagens desta aproximação são evidentes: não há limites (excepto o da memória disponível) ao número de instanciações de objectos do sistema, por exemplo. Adicionalmente, estão previstas nas estruturas em lista linkada duas informações que em muitos casos se revelam preciosas: o nome de cada lista e de cada um dos seus elementos e o índice de prioridade do elemento. Esta regularidade de estrutura estende-se a todos os níveis, incluindo os referidos device drivers, os quais têm um interface mínimo funcional em todos os periféricos ou emulações de periféricos.

DEVICES

Os device drivers aceitam operações genéricas, como inicialização, escrita e leitura, e operações específicas (por exemplo, formatação). Podem representar periféricos, como a porta paralela, ou dispositivos de mais alto nível, como a impressora propriamente dita. Para ilustrar estes conceitos, podemos referir que a porta paralela é efectivamente gerida por um device driver, e que os pedidos de impressão são geridos por outro. Ao primeiro poderíamos chamar, por exemplo, parallel.device; ao segundo, printer.device. Está assim assegurado um nível de independência dos periféricos específicos: se a impressora estiver ligada à porta paralela, o printer.device chama o parallel.device; se a impressora estiver ligada à porta série, será chamado, de forma transparente, o serial.device. Os devices podem ainda ser totalmente constituídos por software, como é o caso dos RAMDISKs.

Ainda em relação ao printer.device, merece realce o facto de, para o programa principal, não ser necessário conhecer as características específicas da impressora desejada para poder ser feita a impressão de gráficos! De facto, os códigos de controlo da impressora, na perspectiva do programa de aplicação, são sempre os mesmos (inspirados num conjunto de códigos de comando ANSI). É o printer.device que faz a tradução e adequação dos comandos internos para a linguagem específica da impressora. Assim, qualquer programa pode tirar o máximo de partido do hardware disponível, sem implementar os seus próprios drivers de impressora.

TRANSMISSÃO

Os device drivers recebem mensagens! De facto, cada device driver (ou instanciação do mesmo) dispõe das chamadas portas de mensagens. Eis mais uma modelar aplicação do conceito de listas linkadas: cada porta de

mensagens não é mais do que um endereço de base duma lista cujos elementos são pacotes de informação. É, assim, possível enviar e receber mensagens de forma totalmente assíncrona (opcionalmente, síncrona), e qualquer número de processos ou tarefas pode partilhar o mesmo periférico, de forma transparente, encarregando-se o device driver de gerir os múltiplos pedidos. O envio e a recepção de mensagens que, como vimos, são caso particular de elementos de listas linkadas, é também suportado a nível de sistema. As portas de mensagens são, forçosamente, parte de uma lista de portas, em que cada uma pode ter nome e prioridade. Assim, nenhuma tarefa precisa absolutamente de conhecer o endereço de outra para com ela partilhar dados ou comandos. Para assegurar a consistência dos dados num sistema multitarefa como o Amiga, existem ainda os chamados semáforos, que não são mais do que bits de guarda de recursos, cuja alteração é garantidamente atómica, ou seja, não interrompível por qualquer tarefa.

NÍVEIS DE EXECUÇÃO

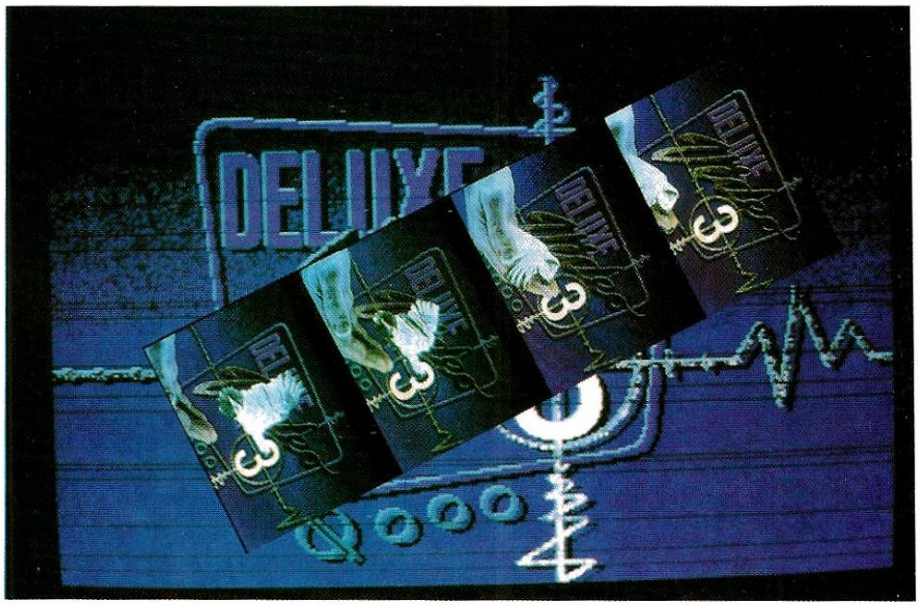
Referimos, atrás, interrupções, tarefas e processos. São estes os três níveis possíveis de execução de código e distinguem-se pelo contexto em que surgem: as interrupções são geradas por acontecimentos de hardware e, como podem surgir a qualquer momento, têm um contexto completamente autónomo (excepto no quadro das portas de mensagens); as tarefas são as unidades básicas de execução do sistema operativo, incluindo o seu contexto uma descrição das operações correntes em termos de device drivers. Os processos são a unidade básica de tratamento quando se requer, além do contexto de tarefa, um contexto relativo a ficheiros e directórios.

Todos estes níveis de execução são prioritizados, de modo a garantir que operações com a mesma prioridade tenham parcelas iguais do tempo de processador. Qualquer nível de execução pode variar a sua própria prioridade ou, mesmo, inibir a execução de outras operações. É por isso que o Amiga OS se designa por "real time, time slicing, preemptive multitasking operating system". "Real time", porque se pode definir rotinas próprias de gestão de acontecimentos assíncronos. "Time slicing" porque a transição do contexto de execução de uma tarefa para outra pode ocorrer segundo critérios de divisão de tempo. "Preemptive" porque os níveis de execução podem alterar a sua própria prioridade, podendo assim dispor de tempo de CPU segundo as suas necessidades. E "Multitasking" porque podem existir múltiplas tarefas em aparente funcionamento simultâneo.

LIBRARYS

Os recursos do sistema são agrupados em módulos a que se dá o nome de library. No caso específico do Amiga, ganham o nome de shared librarys, porque podem ser partilhadas entre processos. Existem, por exemplo, as librarys gráficas, de cálculo em vírgula flutuante, de gestão de ícons, de fontes de caracteres, etc. Porém, independentemente do número de tarefas que requeiram uma determinada operação contemplada numa library, esta é sempre a mesma.

Adicionalmente, as librarys não são necessariamente residentes, podem ser invocadas a partir de disco ou disquete. Deste modo, apenas ocupam RAM quando são necessárias. Um dos reflexos desta metodologia é a pequena dimensão dos programas, dado que muitas das operações estão já definidas a nível do sistema. Alem disso, certas funções são automática e transparentemente aceleradas caso haja o hardware necessário, como por exemplo o cálculo em vírgula flutuante. Normalmente, é executado pelo 68000, mas caso esteja disponível um dos



coprocessadores aritméticos Motorola 68881 ou 68882, todos os programas passam a tirar partido daquele, sem alteração ou reconfiguração.

Em certo sentido, podemos dizer que há *caching* de código executável: quando uma tarefa requere uma library, esta é carregada do disco a menos que já esteja presente em memória. Por outro lado, se nenhuma tarefa declara a sua intenção de usar uma library residente em memória, o espaço ocupado pode ser reclamado, indo juntar-se é lista de espaços de memória livre do sistema.

INTUITION

Provavelmente a mais importante library de sistema, a seguir à EXEC, é a chamada INTUITION.LIBRARY. Esta encarrega-se de gerir os aspectos mais fundamentais da interação com o utilizador. Entre as suas responsabilidades, temos a gestão do canal de mensagens fundamental, em que se agrupam acontecimentos como a pressão de teclas, movimentação do rato, introdução ou retirada de disquetes, etc. A própria estrutura de mensagens do sistema faz a distribuição pelas tarefas interessadas dos acontecimentos, deste modo convencionalizados e dirigidos.

Outra tarefa importante da INTUITION.LIBRARY é gerir o aspecto interactivo dos ecrãs, janelas e menus. A estrutura de dados descritiva destes é muito curiosa: existe uma lista de ecrãs, constantemente actualizada. A cada um dos ecrãs está associada uma lista de janelas, e a cada uma das janelas uma lista de "gadgets" (pequenos ícones que quando seleccionados provocam acções como o fecho ou redimensionamento das janelas) e um menu. Os menus, que são por sua vez estruturas arborescentes baseadas, naturalmente, em listas linkadas, podem ser compostos por texto ou gráficos, e implementam de forma transparente a equivalência entre selecção de opções do menu por rato e por teclas ("hot-keys").

FICHEIROS

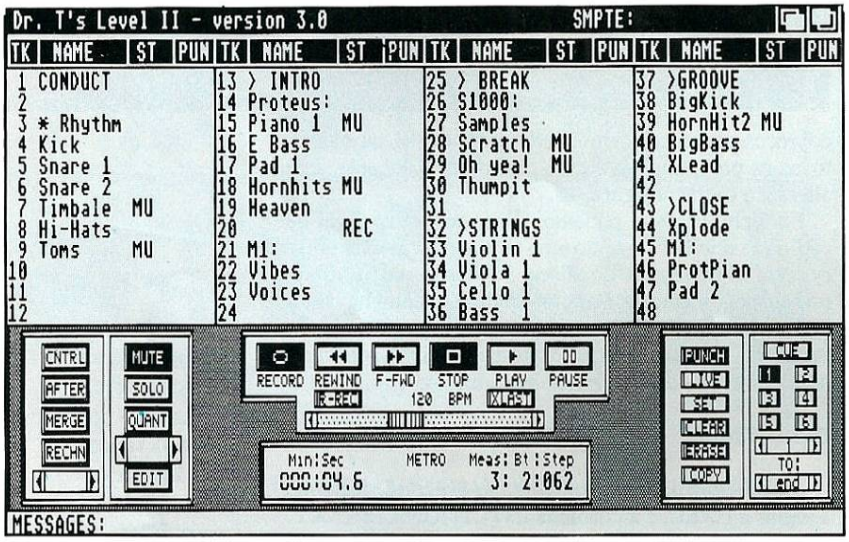
Sobre a estrutura de device drivers e librarys, encontramos o Amiga DOS propriamente dito. Interessa sublinhar que não é obrigatçrio usar o DOS como veículo para toda e qualquer tarefa, como noutras máquinas. No Amiga, o sistema operativo de disco é mesmo relativo a discos e disquetes, apenas cuidando da estrutura de directórios e de volumes, assim como da escrita/leitura de ficheiros.

Originalmente, o Amiga DOS foi implementado como

"Os recursos do sistema são agrupados em módulos a que se dá o nome de library. No caso específico do Amiga, ganham o nome de shared librarys, porque podem ser partilhadas entre processos"

um dos primeiros sistemas operativos de exploração do CPU 68000. Fruto de uma tese de doutoramento sobre arquitectura de sistemas operativos era, na altura da sua concepção, conhecido por Tripos. O facto de o seu desenvolvimento não ter orientação comercial levou à adopção de objectivos e princípios muito sólidos, ou seja, à ideia de uma arquitectura "como deve ser", não apressadamente concebida de forma a apresentar um produto o mais depressa possível. Estes aspectos teóricos trouxeram ao Tripos e ao seu descendente Amiga DOS algumas das características mais notáveis de qualquer sistema operativo.

Em primeiro lugar, temos a rubustez. Um disco ou disquete organizado segundo os princípios do Amiga DOS é muito resistente a acidentes. Por exemplo, pode perder-se completamente a área de directório de uma disquete, e mesmo assim é possível a recuperação automática dessa estrutura. Essa recuperação é efectuada através de um utilitário fornecido com o sistema, e que assenta o seu funcionamento numa estrutura de lista linkada que une os blocos pertencentes a um mesmo ficheiro.



Segundo, a capacidade possível: a dimensão máxima de um único volume é de 2,9 GBytes! É completamente suportada a possibilidade de num mesmo periférico existirem vários volumes. Além disso, o DOS trata correctamente suportes amovíveis. Os volumes são designados por nomes o que, no caso específico das disquetes, permite que as mesmas sejam identificadas independentemente da unidade em que se encontram inseridas.

Terceiro ponto avançado do Amiga DOS: o interface de comando, conhecido por CLI. O CLI é um programa que permite ao utilizador fazer a gestão de ficheiros e lançar outros programas. Característica notável é a possibilidade de dispor de vários CLIs em funcionamento simultâneo. Inclusivamente, um CLI pode passar um ficheiro de texto a outro, de tal forma que se tem uma capacidade de execução batch multitarefa. Os comandos do CLI são os tradicionais de cópia de ficheiros, listagem de directórios, etc., mas com um aspecto pouco comum: a esmagadora maioria dos comandos é transiente, quer dizer, reside como programa na disquete, sendo lido e executado apenas quando necessário. Esta característica permite ao utilizador actualizar facilmente as versões destes comandos, e é em certos casos um nível adicional de segurança, já que na ausência de programas de apagamento de ficheiros não é possível apagar ficheiros. As versões mais recentes do Amiga DOS aceitam ainda uma extensão a este conceito: podem selectivamente definir comandos como residentes na memória. Note-se ainda que dentro da filosofia da independência de dispositivos, um CLI pode ser aberto em qualquer periférico interactivo, ou seja, que aceite entradas e saídas, o que inclui o ecrã, a porta série e a porta paralela.

Quarto ponto (ou, melhor dizendo, pontos): o Amiga DOS aceita nomes de ficheiros com até 63 caracteres, praticamente sem limitações de quais possam ser esses caracteres. Os ficheiros podem ter espaços e sinais de pontuação no nome, portanto. O carregador de programas do Amiga DOS é um "overlay linker/loader" completo: a definição final dos endereços das rotinas é feita na altura do carregamento do programa ("linker/loader"), e os programas podem ter partes residentes em disquete, carregadas sempre que necessário (os chamados "overlays"). Resta referir que as funções previstas a nível do Amiga DOS são altamente compatíveis com as do Unix, facilitando a portabilidade dos programas.

> **"No Amiga, o programador dispõe de meios raramente postos à disposição dele. Basta referir que é possível ao programador contactar directamente com TODOS os níveis do software de sistema"**

WORKBENCH

Opcionalmente, todos os módulos de software referidos podem ser integrados no chamado Workbench. Aqui, temos um interface ao utilizador que unifica os gráficos do Intuition e os ficheiros do Amiga DOS num ambiente de janelas e ícones, directamente descendente dos interfaces do Xerox Star, MacIntosh, etc...

Ao contrário daqueles, porém, é possível dispor em simultâneo de um interface icónico e de linha de comando, dado que se podem criar janelas CLI no Workbench. Outro aspecto importante é que o Workbench é multitarefa, suportando, por exemplo, vários programas em execução simultânea, cada um com a sua janela, sendo automaticamente redirigido o canal de entradas/saídas do Intuition de acordo com as janelas seleccionadas em cada momento.

Os ficheiros de programas e de dados são identificados por ícones, havendo, no entanto, um certo grau de controlo sobre que ficheiros aparecem ou não. Os ficheiros de dados têm ainda associadas informações que permitem seleccionar automaticamente o programa que os deve tratar. Por exemplo, um ficheiro contendo um gráfico pode invocar automaticamente um programa de tratamento de gráficos. Ainda mais interessante, um ficheiro de texto pode invocar automaticamente um processo batch, dando ao Amiga uma característica invejada pela maioria dos outros computadores comandados por ícones e janelas: a execução de tarefas batch por invocação gráfica.

A VISÃO GLOBAL

Depois da descrição sumária dos principais componentes do software de sistema do Amiga, resta ver a forma como os mesmos se integram num todo harmonioso. No Amiga, o programador dispõe de meios raramente postos à disposição dele. Basta referir que é possível ao programador contactar directamente com TODOS os níveis do software de sistema, de uma forma disciplinada, e mesmo compartilhar o controlo de periféricos, o que se tivermos em conta que estamos a falar de uma máquina em funcionamento multitarefa não deixa de ser notável.

Estes diversos níveis de intervenção assumem especial relevância pelo tipo de trabalho repetitivo que evitam: os menus, por exemplo, são partilháveis, e existem mesmo utilitários para a criação automática destes. O mesmo se passa com a organização de ecrãs, "gadgets" e "requesters" (estes últimos são constituídos por uma janela à qual é ligada uma colecção de "gadgets").

Toda a organização do Amiga aponta, portanto, para uma elevada modularidade e normalização. Para além das estruturas de dados e de interface com o sistema operativo, merece especial referência o formato externo de dados conhecido por IFF (Interchange File Format).

Trata-se de um formato normalizado de dados, que permite a transferência de ficheiros entre aplicações. É frequentemente utilizado nos gráficos (caso em que deverá, correctamente, referir-se como IFF-ILBM), e permite que as muitas dezenas de aplicações gráficas do Amiga possam colaborar num sistema de reforço mútuo. Qualquer programa pode aceitar, em princípio, imagens geradas em qualquer outro. Inclusivamente, o IFF-ILBM constitui a estrutura básica do outro grande formato, orientado à animação, o formato IFF-ANIM. Ainda dentro da gama de formatos IFF, estão cobertos formatos de som (''samples'', formato IFF-8SVX), de música (executável quer pelos geradores internos do Amiga, quer por MIDI), de texto formatado, de descrição de geometrias tridimensionais, etc...

AS APLICAÇÕES

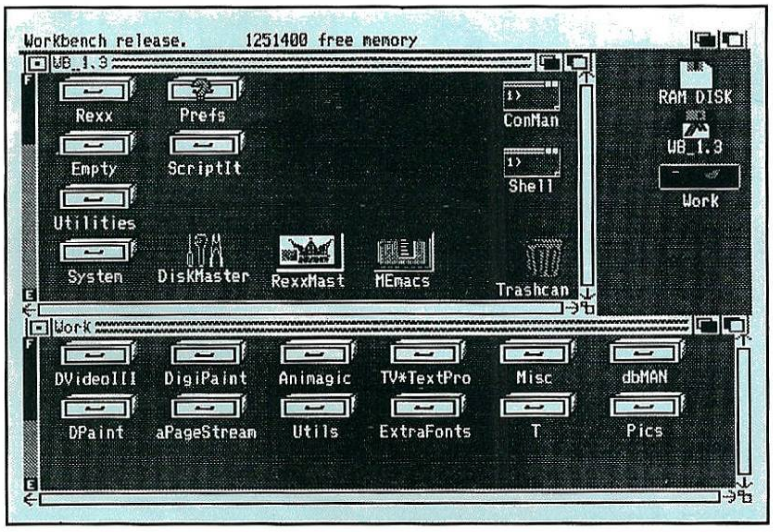
Apesar das más línguas, podemos garantir que o que não falta ao Amiga são programas de aplicações. Numa contagem de modo algum exaustiva, realizada pela revista Amiga World, aparecem referências a mais de 1000 programas de orientação profissional (explicitamente excluindo jogos, portanto). As grandes áreas são, como seria de esperar, os gráficos, os geradores de imagem de síntese, os programas de música, os de animação. Mas também se encontra uma gama interessante de folhas de cálculo extremamente potentes (incluindo ''clones'' do 1-2-3) e de gestores de bases de dados (incluindo clones do dBase, mas também bases de dados integrando som e imagem). A área em que o Amiga se manifesta francamente aquém do seu potencial é o CAD. Se excluirmos a fase de concepção do projecto, para a qual o Amiga dispõe de programas imbatíveis noutras máquinas, a fase do ''drafting'' propriamente dito está limitado a programas de 2 dimensões, em alguns casos notáveis pela sua velocidade, mas inferiores em possibilidades aos grandes do PC ou do Mac. A excepção será, possivelmente, um programa designado por ''Home Builders CAD'', que integra meios de planear habitações e de fazer o seu custeamento.

Outra área que está a ganhar imenso ímpeto é a área do DTP. No Amiga, com a capacidade gráfica de que dispõe, estranho seria que não fosse contemplado o DTP a cores, e de facto é-o. A direcção em que o Amiga aposta foge um pouco da direcção em que os PC e o Mac se têm vindo a deslocar. Apesar de se tomar o PostScript como um meio previlegiado de emissão dos resultados do DTP, o Amiga tem vindo a explorar uma alternativa que é, possívelmente, de maior qualidade. Referimo-nos à tecnologia Compugraphic, desenvolvida pela Agfa, e que é em Portugal e não só a preferida da maioria das tipografias. As capacidades desta tecnologia são especialmente notáveis na área do texto, em que estão disponíveis as melhores fontes, ainda por cima a um custo extremamente baixo, quando comparado com os custos duma fonte Adobe.

A BIBLIOTECA MÍNIMA

Podemos destacar alguns programas, que caracterizaremos com brevidade, entre os muitos disponíveis para o Amiga.

O programa ''obrigatório'' é, sem dúvida, o Deluxe Paint III. Trata-se de um programa de pintura, com capacidades de animação, que deve estar presente em todos os Amigas. Aproveita as capacidades gráficas do Amiga nos modos de 16, 32 e 64 cores, e dada a sua grande integração no sistema tem uma resposta extraordinariamente rápida. Entre as suas características mais marcantes, conta-se a possibilidade de utilizar todas as fontes do sistema (incluindo fontes a cores). As facilidades para animação são também muito completas, possibilitando a



automatização de movimentos, se necessário recorrendo a perspectivas.

Para trabalhos gráficos mais ''fotográficos'', um programa sem dúvida recomendável é o DigiPaint III. Este programa trabalha no modo HAM do Amiga, tendo portanto a possibilidade de lidar com até 4096 cores no mesmo ecrã. Algumas das possibilidades únicas são o tratamento de transparências e a possibilidade de editar ecrãs virtuais, muito maiores que o visível no monitor.

O domínio do ''ray-tracing'' está excepcionalmente bem servido no Amiga (não é à toa que o Amiga conquistou 70% do mercado de imagem de síntese nos EUA). Há muito por onde escolher, mas poderemos destacar o Sculpt-Animate 4D, pelas suas capacidades de animação e pela possibilidade de gerar imagens compatíveis com outros computadores; e merece também menção o Turbo Silver 3.0, que tem amplas facilidades para adicionar texturas a objectos.

No DTP, há essencialmente duas opções de vulto: para sistemas pequenos, o PageStream é um programa notável pela simplicidade de utilização e pela qualidade da impressão, mesmo em impressoras de impacto. Para utilização profissional pesada, temos o Professional Page, muito capacitado para o tratamento de imagens a cores.

No que diz respeito a folhas de cálculo, a dificuldade está na escolha. O MaxiPlan, por exemplo, tem uma gama muito completa de funções, e as amplas dimensões possíveis da folha tornam-no interessante para os grandes modelos.

Na área das bases de dados, podemos destacar o dbMan (um ''clone'' muito completo do dBase) e o SuperBase Professional, um gestor que aceita inclusivamente imagens e sons como componentes dos ficheiros.

Em relação a linguagens, quase se torna obrigatório usar o C, mas estão também disponíveis FORTRAN (o 68000 é uma CPU muito potente para o FORTRAN), Pascal, Forth, e vários sabores de BASIC, incluindo compiladores compatíveis com o GWBASIC.

Os processadores de texto são também muitos, mas podemos destacar o WordPerfect e o ProWrite.

CONCLUSÃO

Por estranho que pareça, não abordámos imensas áreas em que o software do Amiga, tanto a nível de sistema como a nível de aplicações. O espaço disponível não perdoa, porém, e teremos de deixar isso para outra altura. Até lá, recomendamos que na primeira oportunidade experimente um Amiga. Certamente será uma experiência interessante.

Armando Santos

“Apesar de se tomar o PostScript como um meio previlegiado de emissão dos resultados do DTP, o Amiga tem vindo a explorar uma alternativa que é, possívelmente, de maior qualidade”

SISTEMAS DE GESTÃO DE BASES DE DADOS

(PARTE II*)

“Pretendemos, com este artigo, e com aqueles que se lhe seguem, mostrar ao utilizador como é fácil utilizar esta linguagem e abordar alguns temas preponderantes da mesma.”

INTRODUÇÃO

Neste número da **Spooler** vou começar com aquilo a que poderemos chamar um curso de iniciação à linguagem SQL por capítulos, fazendo acompanhar cada um deles de análise a bases de dados que utilizem esta linguagem. Neste primeiro número começaremos por ver as características e módulos acompanhantes do SGBD Oracle. Aconselho, no entanto, o leitor a ler os dois artigos anteriores (de bases de dados, intitulado SGBD!!?, **Spooler** n.º 2; dBASE IV, **Spooler** n.º 4), uma vez que este poderá servir de continuação, pelo menos ao primeiro, onde é apresentado o historial da SQL.

COMO FOI ESTRUTURADO ESTE MINICURSO?

Pretendemos, com este artigo, e com aqueles que se lhe seguem, mostrar ao utilizador como é fácil utilizar esta linguagem e abordar alguns temas preponderantes da mesma. Assim, vamos começar por apresentar um pequeno problema – ''Uma possível informatização das assinaturas da **Spooler**'', e tentar ir mostrando, em partes separadas como funciona a DDL – Data Definition Language, a DML – Data Manipulation Language, e a DCL – Data Control Language. Outros autores descrevem, no entanto, níveis de separação mais detalhados entre as diversas instruções SQL: o Nível da Linguagem de descrição da Base de Dados, o Nível da Linguagem de definição das relações e visualização de partes da Base de Dados, o Nível da Linguagem de manipulação da informação, o Nível da Linguagem de Controlo e Integralidade, e o Nível da Linguagem de gestão das transacções. Apesar de tudo, neste primeiro artigo, iremos ficar pelo primeiro ponto, mencionado por último, que é também o da DDL. Ao longo dos próximos artigos iremos apresentando também outras funções utilizadas por esta linguagem e mostrando, ao mesmo tempo, as diferenças de implementação desta linguagem nas várias bases de dados que iremos analisando.

ANÁLISE E IMPLEMENTAÇÃO

O problema que se põe pode ser levado a uma forma simples, se considerarmos as fichas que os leitores têm de preencher para se tornarem assinantes da **Spooler**.

Aquilo a que normalmente chamamos ''campos'' tem o termo de ''colunas'' em SQL. Assim, os tipos de dados que iremos utilizar são Char, para conjuntos de caracteres; Number, para tipos numéricos, tal como o número da assinatura, e Date para datas. Vamos, assim criar três tabelas, a que chamaremos Assinantes, na qual introduziremos os dados dos assinantes; Hardware, onde introduziremos os dados sobre os computadores em que trabalham esses mesmos assinantes, e Software, onde introduziremos os dados sobre os programas com que os assinantes trabalham, com as instruções SQL da fig. 1:

```
CREATE TABLE          assinantes
(número               NUMBER (6) NOT NULL,
nome  CHAR (30),
dt_nasc               DATE,
profissão             CHAR (20),
cod_hard              CHAR (5),
morada                CHAR (30),
cod_postal            NUMBER (4),
local  CHAR (20),
num_inic              NUMBER (4),
num-fin               NUMBER (4));

CREATE TABLE          hardware
(cord_hard            CHAR (5) NOT NULL,
marca CHAR (20),
modelo                CHAR (6),
veloci NUMBER (2));

CREATE TABLE          software
(cod_soft             CHAR (5) NOT NULL,
nome_soft             CHAR (30),
tipo   CHAR (20));
```

Figura 1

NOTA: É de salientar que os tipos de dados aqui apresentados estão consoante o SQL do SGBD Oracle e podem ser diferentes dos tipos-padrão. Assim, aconselhamos os nossos leitores que quiserem experimentar, a consultarem os manuais das suas implementações. Na primeira tabela (Assinantes) introduziremos os dados relativos aos assinantes (número de assinatura, nome, profissão, morada, código postal e localidade, número inicial e final da assinatura, e dois atributos que servirão de relação entre a Tabela de Assinantes e as outras duas tabelas apresentadas: a tabela (hardware) que conterá computadores e algumas especificações; e a tabela (software) que conterá nomes e tipos de software utilizado. Deste modo poderemos obter, nos próximos capítulos, informações, pelos dados que introduziremos, qual ou quais os tipos de máquinas com os quais os nossos utilizadores trabalham, e qual o tipo de software que utilizam, permitindo-nos assim satisfazer melhor as necessidades do leitor.

** A primeira parte foi publicada na Spooler N.º 2 – página 12*

Mas, voltando ao tema...

O leitor deve ter notado, quase de certeza, na expressão NOT NULL que figura no número de assinatura e nos códigos das tabelas, tanto de hardware como de software. Ela deve-se ao facto de não querermos permitir que sejam deixados em branco estes valores. Podemos criar também, para controlo dos pagamentos, uma outra tabela que conterá o número da assinatura, o número do vale postal ou o número do cheque e o nome do banco pelo qual o cheque foi passado (fig. 2).

```
CREATE TABLE        pagamento
(número             NUMBER (6),
cheque_vale         CHAR (20),
entidade            CHAR (5));
```

Figura 2

No entanto, estaríamos a criar mais uma tabela quando poderíamos ter também estes dados na Tabela de Assinantes para lhe introduzir mais estes novos dados. Assim, apagamos a tabela Pagamento e alteramos a tabela Assinantes, adicionando-lhe os atributos cheque_vale (fig. 3):

```
DROP TABLE pagamento;
ALTER TABLE assinantes
ADD (cheque_vale CHAR (10));
```

Figura 3

Se achássemos necessário, poderíamos ainda introduzir o atributo ENTIDADE e alterar os comprimentos das colunas (fig. 4):

```
ALTER TABLE assinantes
ADD (entidade CHAR (5));

ALTER TABLE assinantes
MODIFY (cheque_vale CHAR (20));

ALTER TABLE assinantes
MODIFY (entidade      CHAR (5));
```

Figura 4

Porém, como todos sabemos, o número de assinantes da **Spooler** tem vindo a aumentar significativamente. assim, pretendemos criar um índice na Tabela de Assinantes, que permita melhorar os acessos a esta tabela. Assim, damos a seguinte instrução (fig. 5):

```
CREATE UNIQUE INDEX num_ass
ON assinantes (número);
```

Figura 5

A cláusula Unique é colocada para indicar que não serão permitidos valores duplicados no índice. Até agora, esta linguagem não passa de mais uma. **É, no entanto, na sua forma de linguagem de interrogação que ela se assume como uma das mais poderosas e eficazes linguagens de interrogação à Base de Dados.** Há até quem diga que a SQL foi uma das melhores coisas criadas pela IBM. Não perca, pois, o próximo capítulo.

SUMÁRIO DE COMANDOS DA DDL

Criação de Tabelas:
CREATE TABLE nome_da_tabela
(nome_da_coluna_1 tipo de dados (tamanho)
nome_da_coluna_2 tipo de dados (tamanho)
nome_da_coluna_n tipo de dados (tamanho));

Criação de índices:
CREATE INDEX nome_do_índice
ON nome_da_tabela (nome da coluna);

Alteração da estrutura da tabela:
I) Adicionar uma coluna:
ALTER TABLE nome_da_tabela
ADD nome_da_coluna tipo de dados (tamanho);

II) Alterar uma coluna:
ALTER TABLE nome_da_tabela
MODIFY nome_da_coluna tipo de dados (novo_tamanho);

Apagar uma tabela:
DROP TABLE nome_da_tabela;

Apagar um índice:
I) Se só existe um índice com esse nome;
DROP INDEX nome_do_índice;

II) Se existem vários índices (de outras tabelas) com o mesmo nome;
DROP INDEX nome_do_índice
ON nome_da_tabela;

DICIONÁRIO DE SQL

Structures Query Language:
Um *user-interface* básico para armazenar dados e retirar informação de uma base de dados.

Base de Dados Relacional:
Uma base de dados que surge ao utilizador como uma colecção de tabelas, relacionadas entre si.

Definições:
1) Uma relação denota uma Tabela;
2) Um atributo é chamado de Coluna;
3) Um simples registo é chamado de Linha;
4) Um valor individual, na intersecção de uma Linha com uma Coluna é chamado de valor Atómico.

Tabela
Estrutura principal da base de dados em forma de tabela rectangular com as seguintes propriedades:

1) É homogénea nas colunas que a constituem, ou seja, numa coluna todos os dados existentes são do mesmo tipo;
2) Cada item é apenas um simples número ou string (conjunto de caracteres);
3) Todas as linhas da tabela têm de ser distintas;
4) A ordem das linhas na tabela é imaterial;
5) As colunas têm nomes distintos, e a sua ordem é imaterial;

O nome de uma tabela pode ser precedido pelo nome da pessoa que a criou, seguido por um ponto (.), e tem de ser diferente do nome de um comando.

Existem dois tipos de tabelas em SQL:
I) As Tabelas de Base, que existem realmente com os dados lá carregados;
II) As Tabelas Virtuais (View), que exitem apenas na definição do Catálogo. Um View é chamado desta maneira porque não existe realmente, como uma tabela base, mas sim como reconstrução ds dados existentes nesta, sempre que é chamado.

CATÁLOGO OU DICIONÁRIO DE DADOS

Uma base de dados relacional que empregue SQL conterá, também, um catálogo. Este não é mais do que uma base de dados que contém informação acerca de Tabelas de Base, Views, Permissões, Identificações, etc., que podem ser obtidos através do comando Select do SQL. No entanto, o seu formato específico depende de implementação para implementação.

DDL, The Data Definition Language – Linguagem de Definição da Estrutura de Dados. Fazem parte desta linguagem os comandos que permitem Criar Tabelas, Criar Índices, Alterar a estrutura das Tabelas, Apagar Tabelas e Apagar Índices.

DML, The Data Manipulation Language – Linguagem de Manipulação dos Dados. Fazem parte desta linguagem os comandos que permitem Inserir, Alterar, Apagar, Consultar e Criar Views.

DCL, The Data Control Language – Linguagem de Controlo da Integridade dos Dados. Fazem parte desta linguagem os comandos que permitem conceder e retirar permissões (Grant e Revoke), bem como os comandos que asseguram a integridade dos dados existentes (Commit e Rollback).

DICIONÁRIO DE SQL

DBA, The DataBase Administrator – Um utilizador autorizado a conceder e retirar permissões de acesso à base de dados a outros utilizadores, a modificar opções que afectam todos os utiizadores, e que possa executar outras funções administrativas.

Atributo – Uma coluna numa Tabela.

Campo ou Valor Atómico – Uma parte de uma Tabela que contém um átomo de informação; a intersecção de uma linha com uma coluna.

Chave – Uma coluna de uma Tabela que serve para identificar um valor único. A(s) coluna(s) que formam uma chave são, normalmente, indexadas.

Índice – Uma característica usada principalmente para melhorar os tempos de acesso e que impõe unicidade na informação.

Palavra reservada – uma palavra com um significado especial em SQL e que, portanto, não pode ser usada pelo utilizador como o nome de uma tabela, coluna ou view.

Coluna virtual – Uma coluna que é obtida dos valores existentes noutras colunas.

Paulo Pinheiro

AS BBS EM PORTUGAL

Ao inaugurarmos a BBSpooler, e estando certos de que este acontecimento vai despertar no leitor a apetência por este meio privilegiado de comunicação, fornecemos a seguir uma lista das BBS já em funcionamento no nosso País. No próximo número da nossa/vossa revista falaremos mais em pormenor da BBSpooler e das BBS em geral.
Para já, não deixe de nos contactar. Embora, ainda longe daquilo que pretendemos que seja, a BBSpooler está aí, razoavelmente operacional. Escusado será dizer que contamos e agradecemos a sua colaboração. A Spooler sempre na procura de novas iniciativas no âmbito da divulgação e desenvolvimento da informática ao nível dos utilizadores, apresenta, com a BBSpooler mais um meio de comunicação de, e, para os seus leitores. Cá esperamos pelas suas sugestões e críticas.
Então? De que está à espera? Ligue já!

AS BBS NACIONAIS POR ORDEM ALFABÉTICA

Designação: **Amitech Portuguese Division**
Telefone: **(01) 4196963**
Horário: **24 Horas**
Operador: **Jorge Carrilho**
Tema Principal: **Software para Amiga**
Acesso: **Privada**
Software: **Ami-Express**

Designação: **BBS MINERVA**
Telefone: **(01) 2954365**
Horário: **24 Horas**
Operador: **João Freitas**
Tema Principal: **Educação**
Acesso: **Público, criada para possibilitar projectos inter-escolas**
Software: **RBBS PC 17.2B**

Designação: **CATS BBS**
Telefone: **(01) 3526422/3/4/5/6**
Horário: **24 Horas**
Operador: **Víctor Hugo Marques/Miguel Vitorino**
Tema Principal: **Geral (Tem uma conferência Spooler)**
Acesso: **Público (não Gratuito)**
Software: **PCBOARD**

Designação: **CAV 12 BBS**
Telefone: **(01) 9211030**
Horário: **18:00 H – 9:00 H, Fins-de-semana 24 Horas**
Operador: **João Alves & Ricardo Portela**
Tema Principal: **Esta BBS foi criada por alunos do 12º ano**
Acesso: **Público**
Software: **PCBOARD 12**

Designação: **DADUS BBS**
Telefone: **(062) 841285**
Horário: **Das 09:00 H às 19:00 H, fins-de-semana 24 Horas**
Operador: **Ricardo Silva**
Tema Principal: **Comunicação com clientes**
Acesso: **Público**
Software: **Wildcat***

Designação: **DATALINK BBS**
Telefone: **(01) 7262878 (RingBack)**
Horário: **Das 20:00 H às 9:00 H**
Operador: **António Cerqueira**
Tema Principal:
Acesso: **Privada.**
Software: **RBBS 17.2B**

Designação: **FAST BBS**
Telefone: **(053) 27566**
Horário: **Das 22:00 H às 8:00 H**
Operador: **Rui Caridade**
Tema Principal: **Apoio á programação de computadores**
Acesso: **Público**
Software: **QuickBBS 2.64**

Designação: **HOST BBS**
Telefone: **(053) 75118**
Horário: **19:00 H às 23:00 H, Fins-de-semana 11:00 H às 23:00 H**
Operador: **Luís Manuel**
Tema Principal: **Divulgação cientifica**
Acesso: **Público**
Software: **RBBS 17.3**

Designação: **NAJA BBS**
Telefone: **(082) 414 717**
Horário: **22:00 H – 24:00 H, Fins-de-semana 15:00 H – 2:00 H**
Operador: **José Martins**
Tema Principal: **Informação geral, apoio aos computadores Atari ST e Amiga**
Acesso: **Público**
Software: **QuickBBS 2.5**

Designação: **OPORTO-BBS**
Telefone: **(02) 578347**
Horário: **22:30 H – 8:00 H, Fins-de-semana 12:00 H – 20:00 H e 22:30 H – 8:00 H**
Operador: **João Romeu**
Tema Principal: **Telecomunicações e Gestão**
Acesso: **Público**
Software: **QuickBBS 2.04**

Designação: **PRIMUS INTER PARES**
Telefone: **(089) 415229**
Horário: **23:00 H – 9:00 H, fins-de-semana 24 Horas**
Operador: **Luís e Vítor Guerreiro**
Tema Principal: **Programação para todas as linguagens**
Acesso: **Público**
Software: **RBBS PC 17.2B**

Designação: **RS BBS**
Telefone: **(01) 9594925**
Horário: **Fim-de-semana, 22:00 H – 08:00 H**
Operador: **Rui Silva**
Tema Principal: **–**
Acesso: **Público**
Software: **QuickBBS**

BBSpooler

☎ (01) 793 87 69

Designação: **SCHOOL BBS**
Telefone: **(068) 24231**
Horário: **14:00 H – 18:00 H**
Operador: **João Fragoso**
Tema Principal: **Vocacionada para o ensino**
Acesso: **Público**
Software: **QuickBBS**

Designação: **SEVEN STARS**
Telefone: **(01) 689491 (RingBack)**
Horário: **17:00 H – 24 H, Fins-de-semana 12:00 H – 24:00 H**
Operador: **Pedro Cardoso**
Tema Principal: **Sistema em RingBack**
Acesso: **Público**
Software: **RBBS PC 17.3**

Designação: **SKYSHIP BBS**
Telefone: **(01) 527623**
Horário: **24 Horas**
Operador: **Mario Pozetti**
Tema Principal: **Intercâmbio de mensagens e ficheiros.**
Acesso: **Público**
Software: **PCBoard 14.2**

Designação: **BBSpooler**
Telefone: **(01) 793 87 69**
Horário: **Dias úteis: das 18.30 H às 10.30 H. Fins-de-Semana: 24 Horas**
Operador: **Jorge Rebelo**
Tema Principal: **Geral. São privilegiados os assinantes da Spooler**
Acesso: **Público e Gratuito**
Software: **RBBS**

Designação: **TELEFORUM**
Telefone: **02) 724917**
Horário: **24 Horas excepto entre às 2:30 H e 3:30 H**
Operador: **Fernando Silva**
Tema Principal: **Correio internacional.**
Acesso: **Público**
Software: **QuickBBS 2.03**

Designação: **VISUS BBS**
Telefone: **(01) 7935839 ou (763697/774377 – das 20.00 H às 08.00 H)**
Horário: **24 Horas**
Operador: **José Câmara**
Tema Principal: **Optometria**
Acesso: **Público**
Software: **PCBOARD**

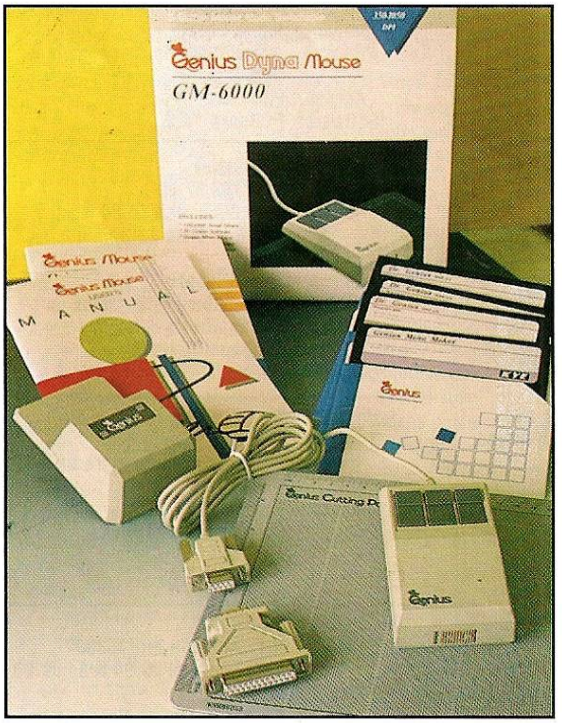
Designação: **WILDCAT BBS**
Telefone: **(01) 4100640**
Horário: **Noite**
Operador: **Filipe Ferreira**
Tema Principal:
Acesso: **Público**
Software: **WildCat***

* RingBack – Significa que deve chamar a BBS uma primeira vez, deixar tocar o telefone 2 vezes, e então voltar a chamar de novo para obter ligação.

MONTRA Spooler

Para encomendar os produtos constantes deste catálogo, basta preencher o cupão publicado para o efeito, e remetê-lo para os nossos serviços. O leitor NÃO deve enviar qualquer forma de pagamento. Todos os produtos serão enviados à cobrança e, aos preços indicados, serão acrescidos os portes e taxas de cobrança. Devido ao facto de alguns dos produtos aqui apresentados esgotarem rapidamente, pensamos ser este o melhor processo para evitar contratempos como seja encomendar produtos que já se encontram esgotados, etc. etc..

GENIUS DYNA MOUSE



A EMBALAGEM DO RATO GENIUS DYNA GM-6000 INCLUI:

- **O rato Genius GM-6000**
- **Software Dr. Genius**
- **Genius Menu Maker**
- **Genius Menu Library**
- **Tapete Genius**
- **Adaptador 9-25 pinos**
- **Caixa suporte para o rato**

Para compatíveis PC/XT/AT/PS-2

Ref.: C001
Preço: 15 620$00

Representante: CEIS

MURATA M900 – FAX

INSTRUÇÕES EM PORTUGUÊS

Ref.ª F001
PREÇO ESPECIAL DE PROMOÇÃO:
115 830$00

O **Murata M900** é um pequeno telecopiador de alta qualidade dedicado a um grande número de utilizadores.

- Compatível **Grupo 3 CCITT**
- Alimentação automática para 5 documentos.
- Velocidade de transmissão de **9600** bps.
- Modo **FINE** para maior detalhe nos documentos importantes
- Controlo manual de contraste
- O **Murata M900** náo necessita de linha telefónica própria. Devido à possibilidade de optar por recepção manual ou automática, uma única linha pode ser usada para conversação e para o FAX.
- Um conjunto de 5 LEDS dão uma informação completa sobre as principais funções do **Murata M900**.
- Também o pode usar como fotocopiador.

Dimensões: 29 x 25 x10 cm
Peso: 3,8 Kg

Representante: TELFAX

KITS PARA LIMPEZA DE DRIVES

Ref.ª C003
Disquete 3 ½
2 340$00

Ref.ª C004
Disquete 5 ¼
2 340$00

Kits de limpeza de drives incluindo disquete e 15 embalagens com o líquido de limpeza. Prática embalagem com suporte auto-adesivo. Garantia de qualidade CLEANDISK®

Representante: CONSULDATA

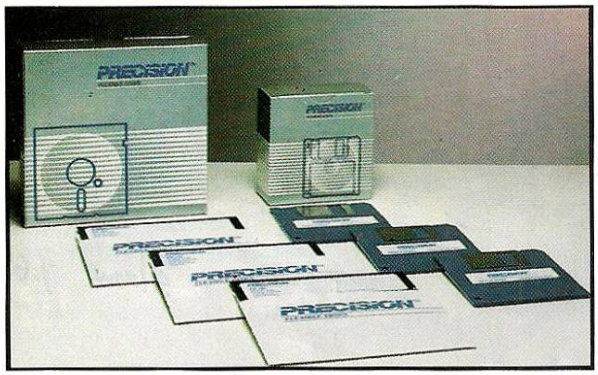
DISQUETES PLATINUM

Representante:
CONSULDATA

Caixas com 10 disquetes:		
– 5,25" HD 1,6 Mb	Ref.: D008	Preço: 2 106$00
– 5,25" DS/DD 48 TPI	Ref.: D009	Preço: 1 287$00
– 3,5" 2D	Ref.: D010	Preço: 3 276$00
– 3,5" HD 2Mb	Ref.: D011	Preço: 7 722$00

DISQUETES PRECISION

Caixas com 10 disquetes:		
– 5,25" HD 1,6 Mb	Ref.: D005	Preço: 3 276$00
– 5,25" DS/DD 48 TPI	Ref.: D006	Preço: 1 521$00
– 3,5" 2D	Ref.: D007	Preço: 3 510$00



Representante: CONSULDATA

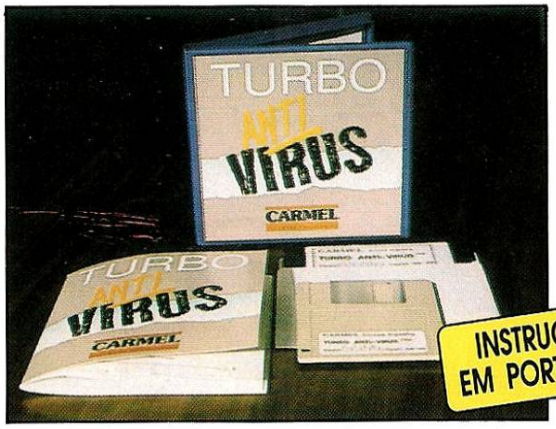
CAIXAS ARQUIVADORAS PARA DISQUETES

Todos os modelos com fechadura e duas chaves

– Mod. DX 50A – 50 disq. 5,25"	Ref.: H006	Preço:	2 223$00
– Mod. DX 100N – 100 disq. 5,25"	Ref.: H007	Preço:	2 340$00
– Mod. SS50 – 50 disq. 3,5"	Ref.: H009	Preço:	2 223$00
– Mod. 80 L – 80 disq. 3,5"	Ref.: H010	Preço:	2 340$00

Representante: CONSULDATA

TURBO ANTI-VIRUS



Finalmente! Um programa capaz de, com segurança, rapidez e eficiência, detectar, eliminar e imunizar o seu sistema de mais de 100 vírus conhecidos.

Com este programa é possível:
- Procurar e eliminar vírus na memória
- Procurar e eliminar vírus em disquetes e discos rígidos
- Imunizar permanentemente os sectores «Boot» e todos os ficheiros
- Reparação automática dos danos causados pelos vírus (incluindo secores danificados)
- 12 meses de garantia. *Upgrades* gratuitos para as novas versões

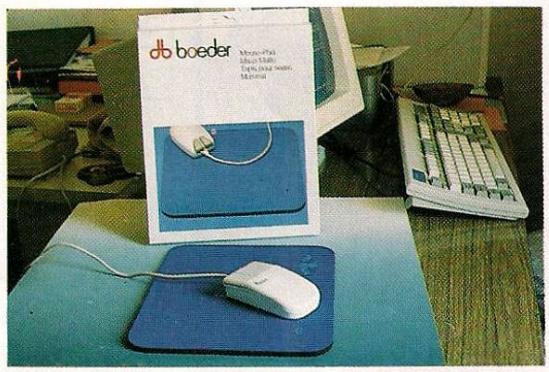
Disponível em disquete de 5 ¼ e de 3½".

Refª TNT 3,5 (Disquete 3 ½")	23 400$00
Refª TNT 5,25 (Disquete 5 ¼")	23 400$00

Representante: S.I.G.I.L.

TAPETE PARA MOUSE

Ref.ª H011
1 521$00



Representante: CONSULDATA

Tapete para Mouse de elevada qualidade. Anti-estático. Permite-lhe uma maior precisão e facilidade no seu trabalho.

MODEMS ACEX 2400 – 9600 BPS

MODEM ACEX 2400BPS/MNP5

Modem compatível "Haynes" comutável a velocidades até 2400, incluindo 1200/75 (CCITT V.23) em modo síncrono e assíncrono, este último com protocolo MNP de correcção de erros até nível 5. Possibilidade de armazenar duas diferentes configurações e quatro números de telefone em memória não-volátil. Um óptimo modem para ligar às suas BBS preferidas mesmo em condições de má qualidade das linhas telefónicas.

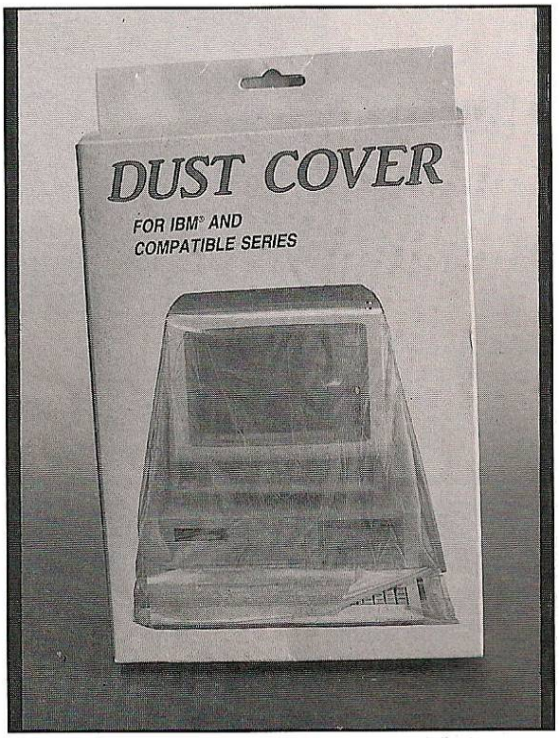
MODEM ACEX 9600BPS (CCITT V.32)/MNP5

Modem compatível "Haynes" comutável a velocidades até 9600 Full Duplex (9600bps com recomendação CCITT V.32). Possui protocolo MNP de correcção de erros até nível 5 em comunicações assíncronas, com memória não volátil para armazenar duas diferentes configurações bem como quatro números telefónicos. Modem de alta velocidade para transmissões de grandes quantidades de dados.

MODEM ACEX-2400	**Ref.: M001**	**Preço: 65 205$00**
MODEM ACEX-9600	**Ref.: M002**	**Preço: 215 280$00**

Representante: CATS

CAPA PROTECTORA PARA PCs



Capa protectora para computadores em plástico especial reforçado para a completa protecção do seu equipamento contra fumos, poeiras, humidade e sujidades de toda a espécie.
Adaptável a qualquer tipo de PC.

Ref.: C010	**PREÇO: 2 691$00**

Representante: CONSULDATA

Recorte ou fotocopie o cupão e remeta-o para:

SPOOLER MAGAZINE
Rua Alfredo Roque Gameiro, 21 - 1º Dtº
1600 LISBOA

CUPÃO DE ENCOMENDA – MONTRA SPOOLER

Spooler nº 10

Desejo receber à cobrança (com os custos acrescidos de portes e taxas) os seguintes produtos:

Quant.	Refª.	Designação	Preço
		TOTAL	

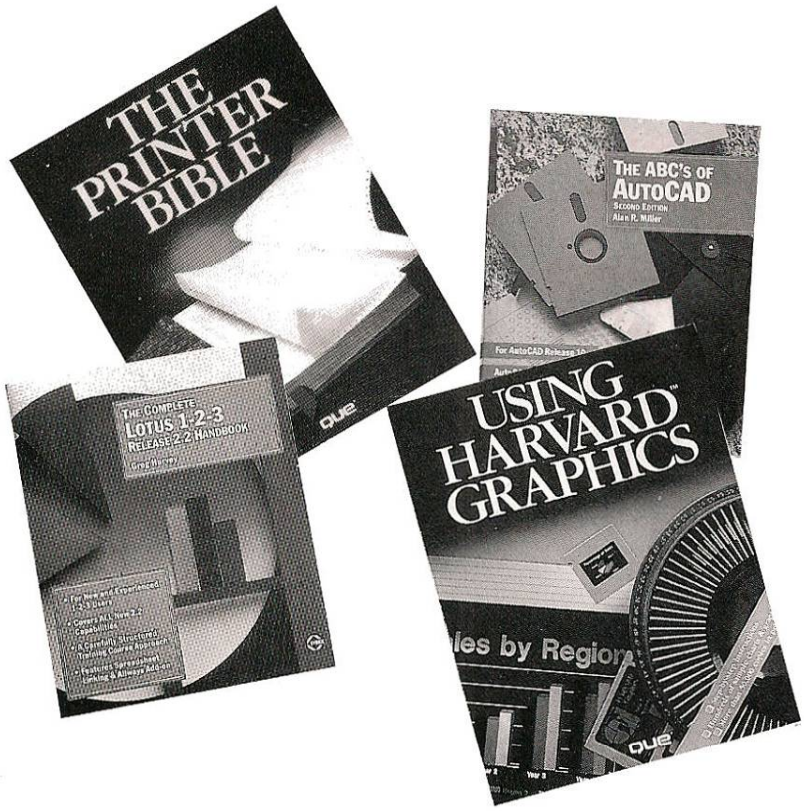
NOME .....
MORADA .....
C. POSTAL ..... LOCALIDADE .....
TELEFONE ..... SOU ASSINANTE Nº .....

JOYSTICK QUICK SHOT

Joystick para compatíveis PC/XT /AT e Placa com selector de 3 velocidades. Óptimo desempenho e *design* atractivo. Um novo dinamismo nos seus jogos predilectos. Preço e qualidade imbatíveis.

Joystick	**Ref.: I001**	**PREÇO: 4 000$00**
Placa	**Ref.: I002**	**PREÇO: 3 500$00**

LIVROS DE INFORMÁTICA «QUE» E «SYBEX»



Representante: LIDEL

LIVROS DE INFORMÁTICA DE DUAS PRESTIGIADAS EDITORAS INTERNACIONAIS

VEJA NA DISQUETE SPOOLER Nº 9 O CATÁLOGO DAS OBRAS DISPONÍVEIS. TRANSCREVA PARA O CUPÃO DE PEDIDO AS REFERÊNCIAS, TÍTULOS E PREÇOS EXACTAMENTE COMO FIGURAM NA LISTAGEM DA DISQUETE. NOTA: TODOS OS LIVROS ESTÃO ESCRITOS EM LÍNGUA INGLESA.

Assinatura conjunta com a CATS-BBS — 12 números da Spooler+12 meses de utilização total da CATS (artigo sobre a CATS/BBS na Spooler Nº 4). Nesta modalidade poupa 2 020$00 no total das duas assinaturas

Recorte ou fotocopie o cupão, preencha-o com letra bem legível e remeta-o, acompanhado do respectivo pagamento em cheque ou vale de correio, para:

SPOOLER MAGAZINE
R. Alfredo Roque Gameiro, 21 - 1º Dtº
1600 LISBOA

NOTA: Os números 1, 2 e 3 da **Spooler** encontram-se esgotados.

** Os dados assinalados com um asterisco podem não ser preenchidos. Estas informações destinam-se a um maior conhecimento dos leitores por parte de todos os que fazem mensalmente esta publicação, a fim de tentar satisfazer melhor as suas necessidades como utilizadores de computadores.*

Spooler nº 10

CUPÃO DE ASSINATURA

Assinante nº:
A PREENCHER PELOS NOSSOS SERVIÇOS

NOME .....
DATA DE NASCIMENTO* ____/____/____
PROFISSAO* .....
TRABALHA COM COMPUTADORES?* SIM☐ NÃO☐
SE SIM ESPECIFIQUE A MARCA E MODELO* .....
ENDEREÇO .....
.....
C. POSTAL ..... LOCALIDADE ..... TELEF(S) .....

DESEJO ASSINAR A SPOOLER A PARTIR DO Nº.___ (inclusive) E DURANTE 12 NÚMEROS.

☐ **MODALIDADE COM DISQ. DE 5 ¼"** ..... **4 000$00**
☐ **MODALIDADE COM DISQ. DE 3 ½"** ..... **7 400$00**
☐ **ASSIN. ANUAL SPOOLER(5 ¼")+CATS-BBS** ..... **11 000$00**
☐ **ASSIN. ANUAL SPOOLER(3½")+CATS-BBS** ..... **14 400$00**

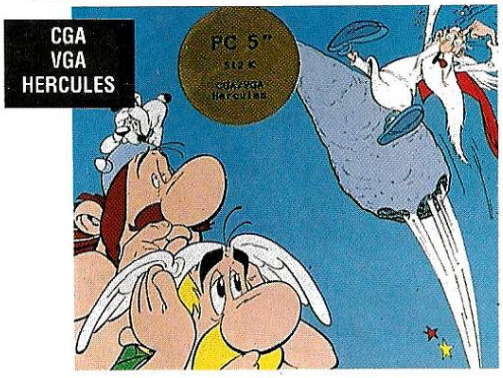
PARA PAGAMENTO DA ASSINATURA ENVIO :
☐ O CHEQUE Nº. ..... S/ O BANCO .....
☐ O VALE DE CORREIO Nº. .....

STOCK LIMITADO

JOGOS

ATENÇÃO: COM PLACA EGA, OS JOGOS SÓ REPRODUZEM 4 CORES.
AS INSTRUÇÕES ORIGINAIS SÃO FORNECIDAS EM FRANCÊS.
PARA QUEM O SOLICITAR, ENVIAREMOS, EM SEPARADO A RESPECTIVA TRADUÇÃO.

TODOS OS JOGOS SÃO ORIGINAIS, INCLUINDO A(S) DISQUETE(S), CONDICIONADA(S) EM EMBALAGEM PRÓPRIA ADEQUADA A CADA JOGO, E ACOMPANHADA(S) POR POSTERS E/OU MANUAIS DE INSTRUÇÕES.
***SÃO PROIBIDAS A REPRODUÇÃO, POR FOTOCÓPIA, DA DOCUMENTAÇÃO IMPRESSA, E A CÓPIA DOS PROGRAMAS AQUI COMERCIALIZADOS.**

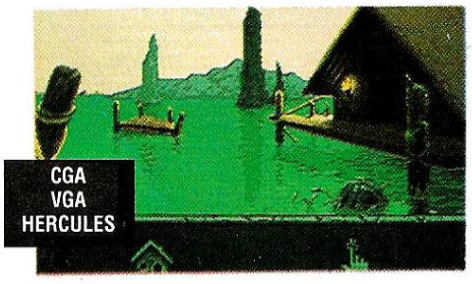


ASTERIX E A PANCADA DO MENIR

O Panoramix levou com um menir na cabeça. É necessário ajudá-lo a lembrar-se da fórmula da poção mágica, procurando e misturando diversos ingredientes escondidos num bosque repleto de javalis e soldados romanos.

Acção e astúcia constituem a mistura explosiva deste superjogo inspirado no filme homónimo.

REF. J007 **PREÇO: 5 265$00**



A LENDA DE DJEL

Procurar a todo o custo salvar o reino da fome e das epidemias, entre outros males não menos graves, é apenas uma das muitas tarefas de que está incumbido ao iniciar este jogo.

Ao longo de cerca de três dezenas de ecrãs tudo se tornará possível. A vitória ou a derrota dependerão em grande parte de si.

REF. J002 **PREÇO: 5 265$00**

SKIDOO

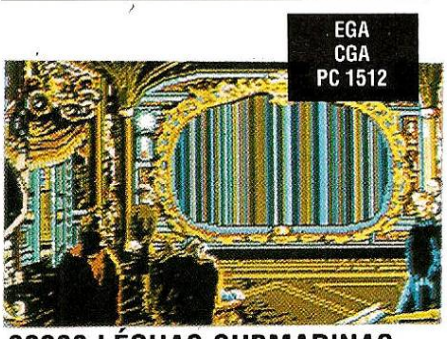
Pilotando uma *scooter* para neve, você faz um *raid* de 6 meses para completar as etapas do campeonato.

A 180 Km/h, o seu bólide dos gelos atravessa as florestas, lagos gelados e montanhas do grande norte-canadiano.

Em cada uma das 25 povoações, você pode participar em corridas diabólicas sobre gelo, onde nove adversários tudo tentam para o fazer «comer» o gelo.

– Simulação excitante de condução dum *Skidoo*.
– Mapa realista do nordeste canadiano (4 000 000 km²)
– Paisagens inesquecíveis.
– 25 percursos de provas diferentes.

REF. J008 **PREÇO: 5 265$00**



20000 LÉGUAS SUBMARINAS

Heróis desta fantástica aventura de Julio Verne, os jogadores vivem momentos inquietantes e misteriosos nas profundezas do oceano, percorrendo as páginas de um romance clássico de uma forma muito mais interessante e... menos clássica.

REF. J004 **PREÇO: 5 265$00**

FREEDOM

Numa plantação do século XVIII a sua missão é tentar uma evasão na posição de líder de uma rebelião de escravos. Mesmo com a ajuda dos restantes escravos a tarefa não será nada fácil: enfrentar todos os inimigos, os cães, e o próprio dono da plantação não é um trabalho que qualquer um possa empreender.

Um excelente jogo de estratégia e acção.

REF. J005 **PREÇO: 5 265$00**

AFRICAN RAIDERS

Simulação de condução 4x4 no deserto.

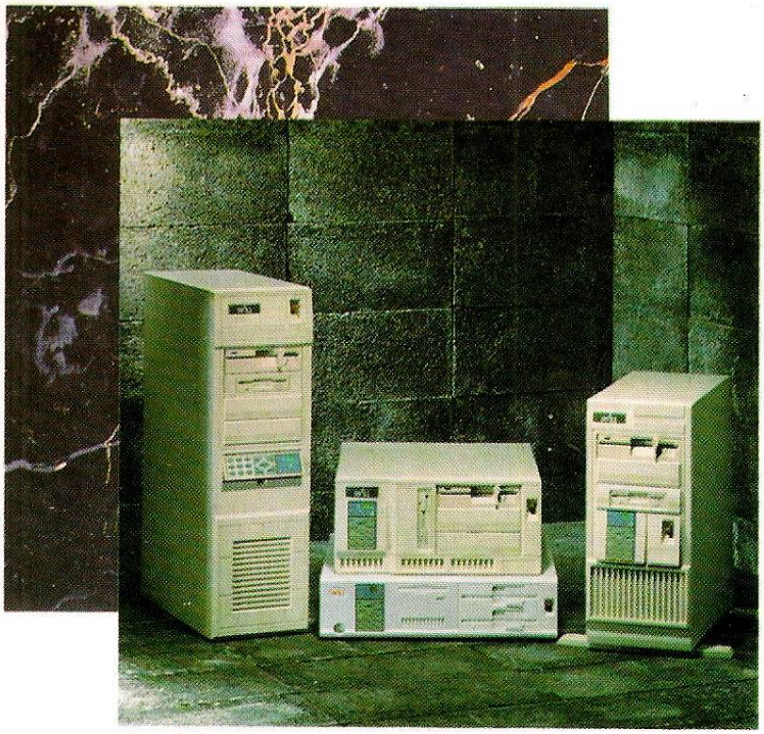
Simulação bastante realista, sem estradas, com algumas pistas, obstáculos naturais, e numerosos participantes para vos dificultarem a vida até Dakar. Dispondo de um satélite de bordo, uma bussola, e de um mapa como instrumentos principais de trabalho a tarefa de chegar "a bom porto" não se mostra nada fácil.

REF. J001 **PREÇO: 5 265$00**

Pense Classe

Pense Inovação

Pense Wugo



Adicionámos um toque de classe à tecnologia, oferecemos uma gama completa de máquinas poderosas e fiáveis, sob as mais diversas opções.

O sistema de protecção por password oferece completa segurança, impedindo o acesso à máquina por pessoal não autorizado.

Velocidade de processamento, inovação, design, qualidade e boa assistência, são o nosso lema e sua garantia.

Venha visitar-nos, temos o nosso melhor conselho e solução para as suas necessidades.

CATS

Telefones 52 40 27, 352 80 65 Fax 352 64 21, BBS - 352 64 22/3/4/5/6 Telepac 005 229 048

ALGUÉM TEM QUE SER MELHOR QUE TODOS OS OUTROS

Destaque Nacional
1989

Conferido a

Consuldata

em reconhecimento à sua valiosa contribuição no desenvolvimento das vendas de

Office Market

3M Portugal, Lda.

3M

António Barbino
Div. Sales Mkt. Manager

James T. Mahan
Managing Director

PLACA DOURADA
1.º Prémio de 1989

Consuldata
SERVIÇOS DE INFORMÁTICA, LDA.

ADMINISTRAÇÃO E VENDAS:
R. da Quinta do Almargem, 7-A, 7-B — 1300 LISBOA — PORTUGAL
Telefs: 64 48 54/64 51 38/64 41 50 — Telex: 63 755 ULDATA P — Telefax: (01) 64 91 08