if

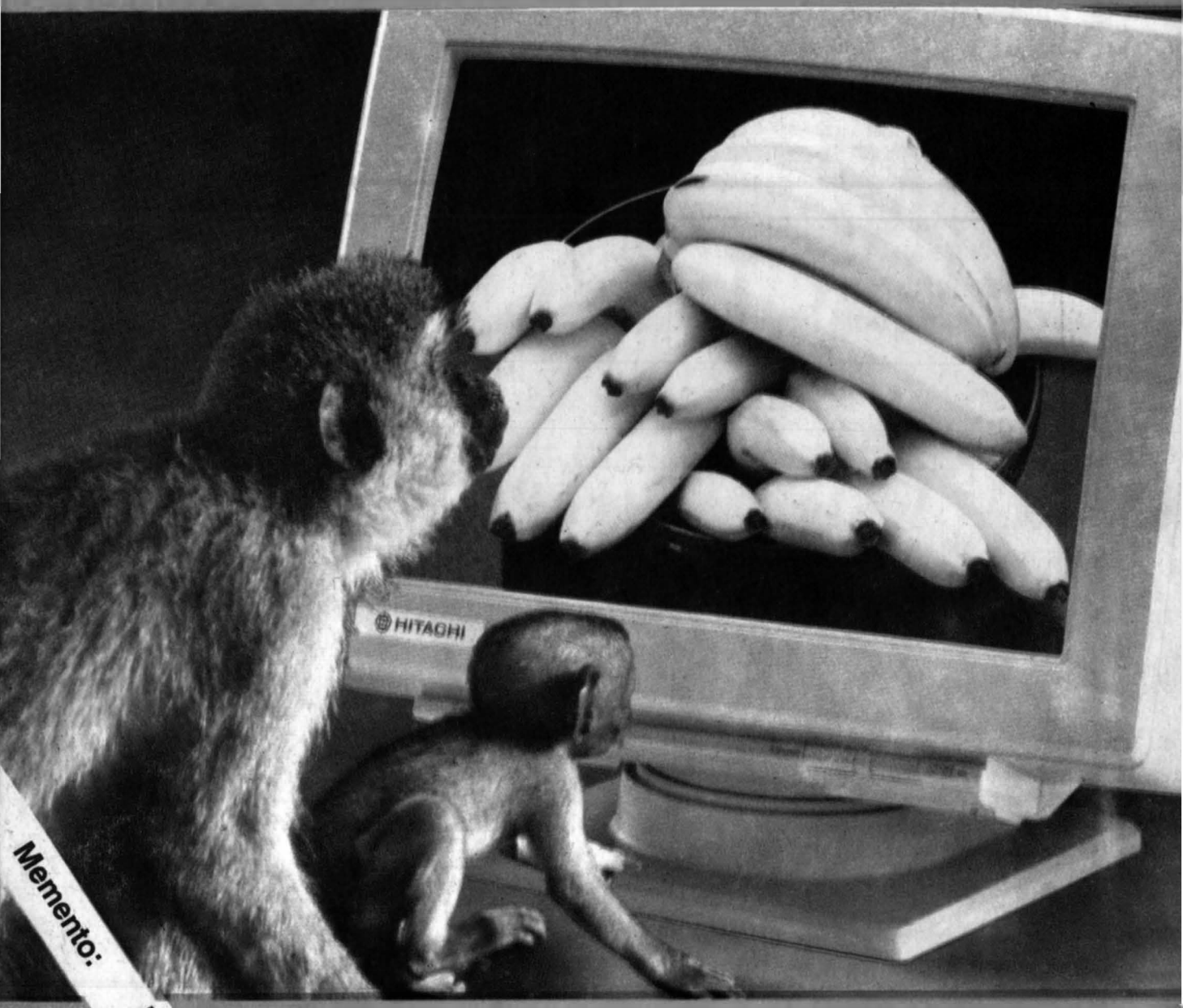
CALCULATOARE PERSONALE

5 / 91

(8)

57 Lei

Revistă lunară editată de Micro ATCI Tîrgu Mureş



Memento:

Funcţii Turbo-C

MS-DOS 5.0 vs. DR-DOS 5.0

Dosar: Turbo C++ 1.0

Paradigma Paradox

Standarde grafice:

Tiga, 8514/A, XGA

AbMod

Abrihan & Moldovan s.n.c.

vă oferă **tehnologie americană**
inteligenţă europeană
preţuri orientale **la**

produse din domeniile:

- **tehnică de calcul (hardware şi software)**
- **telematică şi birotică**

servicii în aceleaşi domenii:

- **lucrări de cercetare / proiectare**
- **studii de dezvoltare şi modernizare**
- **studii de marketing pe plaţa internă/externă**
- **consulting şi asistenţă tehnică**
- **instalare, şcolarizare, service**
- **sisteme "la chele"**

condiţii de livrare: 2 - 4 săptămîni

Cîteva exemple din lista noastră de preţuri:

XT

<< Preţ sistem complet 89.000 lei >>

AT-286 & AT-386

<<Preţ sistem complet [lei] > >

AT	Winchester capacitate formatată	monitor + interfaţă		
		mono 14"	EGA 14"	VGA 14"
286	40 Mb	199.000	249.000	299.000
	80 Mb	275.000	325.000	375.000
386	80 Mb	399.000		499.000
	160 Mb	549.000		649.000

Imprimante matriciale

Tipul	Lăţimea	Nr. ace	Preţ [lei]
LC 20	80 car	9 pin	49.000
LC 15	132 car	9 pin	79.000

Mouse serial compatibil Microsoft . . . . . 7.700 lei

Copiatoare FC-2 (Toshiba BD 2810) . . . . . 255.000 lei

MS-DOS v4.01 . . . . . 12.000 lei

Cel Interesaţi pot obţine gratuit lista de preţuri adresîndu-se la

AbMod Magazin Hardware-Software:

b-dul 22 Decembrie nr. 135

AbMod Sediul central:

str. Moţilor nr. 91, 3400 Cluj-Napoca, tel./fax 95-111090

AbMod Sucursala Oradea:

str. Dacia nr. 40, 3700 Oradea, tel. 991-60278, 44476, fax 991-56881

if

revistă de informatică
editată de firma **Micro ATCI**

Director: ing. Dumitru Dunea

Redacţia:

ing. Iosif Fettich,
ing. Ingrid Maier,
ing. Romulus Maier,

Colaboratori externi:

stud. Daniel Buleu
ing. Cristian Malide
stud. Ioan Tiberiu Socaciu

Tiparul: tipografia Tîrgu Mureş
Tipografi:

Halaţiu Mihai,
Cormoş Mircea

Revista apare lunar.

Preţ: 57 lei

Adresa şi telefonul redacţiei:
Micro ATCI,
RO-4300 Tîrgu Mureş,
C.P. 64, O.P. 1,
tel. 954/31660 (direct),
33.612, 24.158, 20.057,
33.511, int. 134 sau 189
fax 954/35208,
telex 65354.

Manuscrise originale sau listing-uri de programe sînt primite cu plăcere de redacţie, cu condiţia să nu fi fost publicate şi în altă parte. Prin expedierea unui manuscris pe adresa redacţiei, autorul consimte implicit la publicarea materialului său în cadrul revistei. Onorariul se negociază cu redacţia. Materialele nepublicate nu se înapoiază şi nu se reţin.

Revista noastră vă oferă spaţiu pentru reclamă şi publicitate. Pentru amănunte vă rugăm să luaţi legătura cu redacţia.

Cei care doresc să anexeze revistei pliante publicitare tipărite în regie proprie, sînt rugaţi şi ei să se adreseze redacţiei.

Fantázia în primejdie

O revistă "îţi conferă puteri însemnate"(*), "dar n-ai voie să le foloseşti", "niciodată n-ai voie să intervii, indiferent de ce vei vedea, căci începînd din clipa aceasta propria ta părere nu mai are nici o importanţă". "Trebuie să laşi să se întîmple tot ce se întîmplă. Totul trebuie să aibă aceeaşi importanţă pentru tine, şi binele şi răul, şi ceea ce-i frumos, şi ceea ce-i urît, şi prostia, şi înţelepciunea". "Nu ai voie decît să cauţi şi să întrebi, nu însă să şi judeci după propria ta judecată. Să nu uiţi asta niciodată"!

Bine, dar sîntem muritori şi sîntem slabi şi nu sîntem noi aleşii. Şi uităm şi greşim!

Nu puţine publicaţii şi nu doar cele aflate "de o anumită parte a presei" au avut în ultimele zile titluri de genul: "Este ameninţată existenţa presei independente din România". Noi ce-am mai putea adăuga. Sîntem şi noi în derivă în acest triunghi al Bermudelor: hîrtie - spaţiu tipografic - difuzare. Cînd toate aceste elemente se scumpesc, în NLL (noua limbă de lemn) "se liberalizează", nu putem alege decît între a scumpi şi noi revista, sau a renunţa la editarea ei. Ne este ruşine pentru fiecare scumpire şi nu ne consolăm făcînd comparaţii de genul: de preţul unei reviste cîte felii de salam (cu sau fără soia) s-ar putea cumpăra. Oricum, "primum vivere".

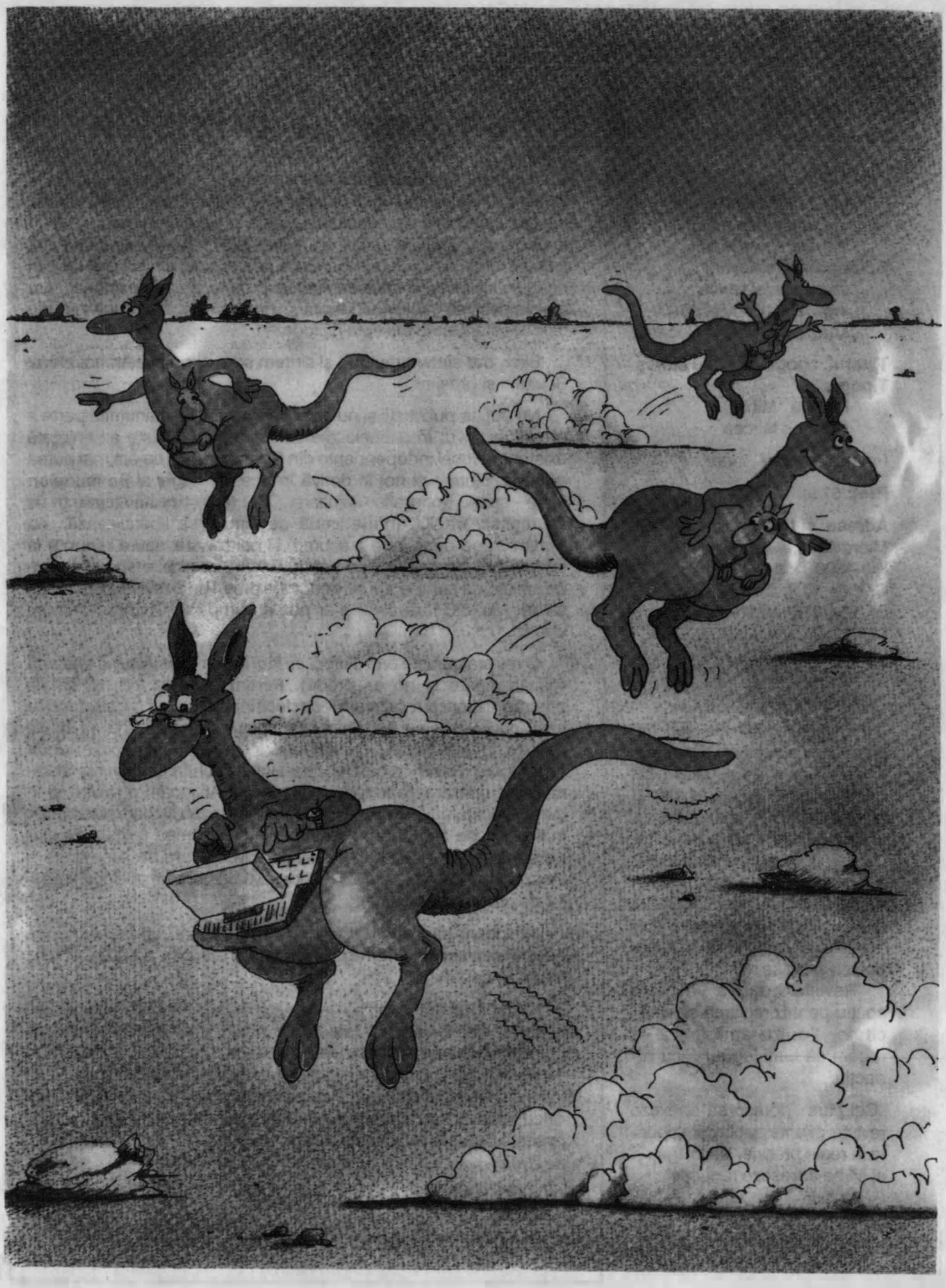
O revistă trăieşte prin cititorii ei. Noi dorim ca revista să trăiască (garda moare şi nu se predă). Pentru aceasta trebuie să ne păstrăm cititorii şi n-o putem face decît într-un singur fel: oferindu-le ceva de care ei au nevoie. Acest ceva este informaţia la zi, de calitate, interesantă, cuprinzătoare. Încercăm şi în acest număr acest lucru. Revenim cu o aducere la zi în ceea ce priveşte sistemele de operare şi adaptoarele grafice. Aruncăm o privire şi în curtea vecinului şi vă prezentăm familia de calculatoare Macintosh în intenţia de a vedea ce se întîmplă şi în lumea calculatoarelor personale care nu sînt compatibile IBM. Facem de asemenea o incursiune în lumea harddisk-urilor şi vă prezentăm pe scurt limbajul de programare C.

Nu dorim să destabilizăm, aşa că dacă nu aţi încheiat încă negocierea salariilor vă rugăm să nu citiţi articolul "60.000 DM deja pentru începători".

Cît de mult am reuşit să ne apropiem de scopul propus rămîne la aprecierea D-voastră. Noi vă semnalăm doar instaurarea Nimicului. "La început era doar ceva foarte mic, o nimica toată, cît un ou de bibilică. Dar astfel de locuri s-au tot lăţit, şi dacă din greşeală intra cineva cu piciorul acolo, dispărea piciorul - sau mîna - sau orice intra acolo. De altfel nu doare, numai că respectivului îi lipseşte brusc o bucată din el."

Ing. Romulus Maier

(*) Michael Ende - "Povestea fără sfîrşit"



Cuprins

Adio ASCII

Producători de vîrf din domeniul calculatoarelor vor să înlocuiască în scurt timp setul de caractere ASCII; cu "Unicode", setul ISO ar fi astfel dintr-o dată învechit.

Mai ales la legături care trec de hotarele unei ţări, setul de caractere ASCII impune restricţii partenerilor de comunicaţie. În prezent, cîţiva dintre cei mai importanţi producători de calculatoare vor să prezinte o alternativă: "global computer code for storage and transmission of text around the world". Întreprinderile au fondat Unicode Inc., care va promova setul de caractere universal cu acelaşi nume. Codul de 16 biţi folosit de Unicode asociază fiecărui semn utilizat în prezent un număr şi permite codificarea a circa 65.000 de semne, care ar acoperi "toate limbile vii mai importante" folosite în lume. Inclusiv unele idiomuri din Japonia, China, Taiwan şi Coreea, semnele scrise din alfabetele chirilic, ebraic, arab, grec şi sanscrit. În afară de acestea, noul set conţine simboluri matematice şi tehnice, precum şi semne diacritice pentru deasupra sau dedesubtul literelor, precum şi coduri de comandă (ca de exemplu "avans linie"). Dacă Unicode se impune, munca de ani şi ani de zile depusă de ISO (International Standard Organization) va deveni inutilă. Firme importante (Xerox, Apple, IBM, Microsoft, Sun, Novell, Aldus, Go şi Next) şi-au declarat sprijinul pentru Unicode.

Netware unificat

La CeBit '91, Hanovra, Novell a anunţat versiunea v2.2 a sistemului său de operare în reţea, Netware. Un interes deosebit prezintă faptul că prin aceasta, produsele de pînă acum (ELS I, ELS II, SFT şi Advanced Netware) sînt reunite pentru prima dată într-o unică serie de produse.

În viitor, un potenţial cumpărător nu va mai trebui să-şi elaboreze cu minuţiozitate decizia de achiziţionare a unei reţele de 4, 8 sau nr. maxim posibil de posturi de lucru. Ceea ce pînă acum nu era un banal amănunt tehnic, ci o decizie complicată şi foarte serioasă.

Căci versiunile pentru 286 de pînă acum difereau nu numai după numărul de utilizatori, ci şi calitativ; erau scrise fiecare în alt cod şi nu permiteau treceri de la o variantă la alta decît prin înlocuire.

Trecerea la Netware v2.2, care va putea fi achiziţionat în versiuni pentru 5, 10, 50 şi 100 de utilizatori, va uşura nu numai viaţa utilizatorilor, ci şi a producătorilor de reţele. Căci organizarea service-ului şi întreţinerii pentru un singur produs sînt mai ieftine şi mai sigure pentru un singur produs decît pentru patru.

Astfel că despărţirea de versiunile de Netware de pînă acum, care nu se mai "produc" din ianuarie, va fi fără regrete. Că acest lucru nu va fi adevărat şi pentru vînzători şi revînzători, pare evident. Căci instalarea mai complicată, extensia, service-ul, etc., însemnau desigur afaceri mai mari. Toate acestea din aprilie/mai anul acesta, se vor fi terminat, odată cu lansarea noii versiuni.

Prin restructurarea versiunilor pentru 286, Novell modifică semnificativ tocmai acea serie de produse din care pînă acum s-au făcut cele mai multe vînzări. Căci, măsurînd în bucăţi, ELS I, ELS II, SFT şi Advanced Netware reprezentau circa 90% din vînzări (celelalte 10%: Netware 386).

Văzută dinspre piaţă, introducerea versiunii noi, unificatoare de Netware trebuia să se întîmple chiar de mai mult timp. Căci concurenţa (fie LAN-Manager, fie Vines) oferă deja de timp îndelungat o versiune funcţional echivalentă şi diferind numai prin numărul de utilizatori.

Noul Netware dintr-o privire

Odată cu unificarea liniei sale de produse pentru 286, Novell a introdus cîteva îmbunătăţiri. Cele mai importante sînt:

- instalare simplificată. Procesul de instalare de pînă acum (Netgen) a fost înlocuit printr-un program de instalare complet nou. >Install< cere utilizatorului cîţiva parametri şi face apoi singur configurarea. Procesul de instalare (numit de Novell "intuitiv") e sprijinit de help-uri detaliate, apelabile oricînd cu F1.

- manuale mult îmbunătăţite. Analog instalării mult simplificate, şi manualele s-au subţiat mult. Iar un "Getting started" de 26 pagini va permite o orientare rapidă a utilizatorului.

- Help on-line, prin care utilizatorul are acces direct la documentaţie, ceea ce în marea majoritate a cazurilor ar trebui să facă superflue căutările în manuale.

- structură îmbunătăţită a memoriei file-serverului, ceea ce face posibile mai multe FSP-uri (File Service Process) simultan.

- suport extins pentru Macintosh. Noua versiune Netware poate conlucra cu Appletalk Phase 2, Appletalk Filing Protocol v2.0 şi Tolkentalk.

- opţiuni îmbunătăţite pentru imprimantă. Serverul de imprimantă este conţinut în Netware 2.2, ca VAP şi Exe. S-au făcut îmbunătăţiri, s-au introdus comenzi noi, etc.

- suport extins pentru staţii de lucru. În această categorie intră posibilitatea de a face automat un "Workstation Update"; funcţii noi la suportul de workstation pentru DOS, Macintosh şi OS/2.

- Suplimentar în Netware v 2.2 sînt integrate drivere de 3COM, s-au făcut modificări în "trustee rights" şi au fost operate îmbunătăţiri în utilitare ca FConsole, Login, Map şi Filer.

HITec
Electronic Engineering SRL

ş ţ ă î

sau

Computerele şi Limba Română

Întrebarea dacă şi la noi calculatoarele vor ajunge vreodată să joace vreun rol în viaţa şi activitatea noastră nu mai are sens. Este clar că avem nevoie de ele. Ce ne facem însă că nu prea ştiu să vorbească româneşte, iar cînd o fac, vorbesc atît de peltic, încît nici nu ştim să rîdem sau să plîngem. Evident, pînă acum s-a recurs la multe soluţii, care mai de care mai ingenioasă, mai simplă sau mai complicată, şi mai ales mai "incompatibilă". Fără a. avea pretenţia de a fi găsit cea mai bună soluţie, în urma studiilor noastre s-au conturat cîteva sugestii-rezolvări.

Pentru ca un calculator să poată fi folosit şi în limba română, este necesar înainte de toate să codificăm cumva şi literele româneşti în setul de caractere cunoscut de calculator. Este esenţială stabilirea de comun acord şi acceptarea la nivel naţional a unui asemeni set românesc de caractere, pentru a asigura compatibilitatea între utilizatori. Pe MS-DOS există deja un mecanism de comutare a seturilor de caractere (code page system), dar literele româneşti nu au fost prevăzute în nici unul din seturile disponibile. Două sînt seturile de interes pentru noi: setul obişnuit (437) şi setul internaţional extins (850, care sacrifică o parte din caracterele semigrafice în favoarea unor litere speciale). Deoarece introducerea literelor româneşti în tabelă nu se poate face decît prin renunţarea la alte caractere, este foarte importantă alegerea judicioasă a acestora. Propunerea noastră este următoarea:

(0x83; 131)	â
(0x86; 134)	ă
(0x8C; 140)	î
(0x8F; 143)	Ă
(0x91; 145)	ţ
(0x92; 146)	Ţ
(0xA6; 166)	ş
(0xA7; 167)	Ş
(0xA8; 168)	Â
(0xAD; 173)	Î

Această amplasare conduce doar la pierderi minime de caractere (cîteva litere din alfabetul ţărilor nordice şi două semne de punctuaţie din limba spaniolă). În plus această substituire este posibilă în ambele tabele, atît cea normală cît şi cea internaţională, conducînd la apariţia a două seturi de caractere: românesc şi românesc extins.

A doua mare problemă este claviatura. Fiecare ţară europeană a reinventat, cu mai mult sau mai puţin succes, o altă aşezare a tastelor (vezi claviatura germană foarte dificil de utilizat pentru orice altceva decît editare de texte). Dificultatea proiectării unei claviaturi româneşti provine şi din faptul că două dintre cerinţe sînt contradictorii: una impune ca literele româneşti să fie amplasate în dreapta, în continuarea celorlalte litere (ca la maşina de scris: <K L Ş Ţ> <O P Ă Î>) iar cealaltă cere ca semnele de punctuaţie, parantezele drepte şi acoladele să rămînă în aceeaşi poziţie. Rezultă clar că o claviatură românească tebuie să aibă două moduri comutabile funcţie de activitatea preponderentă: dezvoltare de programe sau editare de texte. În mod normal, claviatura va genera aceleaşi coduri ca o claviatură americană, cu menţiunea că folosite împreună cu Alt sau Alt+Shift, tastele din dreapta vor genera codurile literelor româneşti. În modul editare de texte, funcţia acestor cinci taste se inversează, generînd direct <ş ţ ă î>, semnele de punctuaţie fiind generate doar apăsînd simultan Alt sau Alt+Shift.

Afişarea în ecran, pentru modul alfanumeric obişnuit, nu este o problemă, plăcile EGA şi VGA permiţînd definirea completă a setului de caractere.

O ultimă problemă importantă este afişarea pe imprimantă. Pentru imprimantele matriciale compatibile IBM cu set de caractere definibile prin soft, nu sînt dificultăţi majore, putîndu-se încărca în imprimantă imaginile celor zece caractere speciale româneşti. De asemenea, în cazul imprimantelor cu laser compatibile Adobe PostScript, se pot redefini destul de uşor fonturile de caractere, încît literele româneşti să apară corecte la orice dimensiune, înclinare, direcţie, ş.a.m.d., cu inegalabila calitate proprie fonturilor Adobe.

După cum am spus şi la început, aceasta nu se vrea soluţia definitivă a problemei literelor româneşti pe calculatoare. Mai rămîn destule pete albe în dosarul acestei probleme (ca de exemplu modificarea nenumăratelor fonturi folosite de programele care lucrează în mod grafic, cum ar fi Windows 3.0, Ventura Publisher, etc., sau a imprimantelor cu laser care nu folosesc limbajul PostScript). Ceea ce ne dorim să realizăm este o incitare la discuţii, cu convingerea că în cele din urmă hotărîrea ce se va adopta se va respecta la nivel naţional şi va satisface cele mai exigente preferinţe.

În prezent firma HiTec are în curs de dezvoltare extensia cu literele româneşti pentru editorul WordStar 6.0, propunîndu-şi să realizeze şi o versiune pentru Microsoft Word.

Pentru orice lămuriri, sugestii şi propuneri, ne puteţi contacta la următoarea adresă:

HiTec Electronic Engineering SRL R-70317 Bucureşti, Str. Gen. Ipătescu Nr. 4 Tel. (90) 426100 Tlx. 11391 global r

ROMORGADATA

Desfăşurată între 7.-10.05.1991, în Complexul Expoziţional din Piaţa Scînteii (pardon, Presei Libere), expoziţia specializată organizată de ROMEXPO S.A. şi I.E.G. Salingen a fost, se pare, un succes. Cel puţin pe partea care ne interesează - PC-uri & Co.

Au fost prezentate multe exponate, majoritatea de vîrf, şi s-au putut face contracte pe loc. De la imprimante laser HP Laser Jet IIID şi calculatoare de elită ca HP Vectra 486/33T pînă la banale (dar importante) casete de toner, dischete sau hîrtie xerox, cartele de extensie de tip fax/modem, soundblaster sau laptop-uri Toshiba 486 - cam toată panoplia PC-urilor a fost prezentă. Nu în ultimul rînd Borland a stîrnit interes, atît prin "programul de amnistie" anunţat cît şi prin preţurile în general accesibile pentru marea majoritate a produselor. Şi au putut fi văzute - şi cumpărate - ultimele noutăţi; inclusiv Borland C++ 2.0 (143.055 lei la Logic) sau Novell Netware v2.2 (188.605 lei pentru 5 utilizatori). Se pare că a venit vremea cînd - pe lei! - se poate cumpăra (aproape) orice hard sau soft. Timpuri noi?

Perestroika

Joc de acţiune pentru EGA/VGA - primul joc shareware din URSS! Un joc pasionant, cu multă acţiune, efecte sonore comice şi o grafică foarte bună. Jocul are 25 de nivele de joc şi necesită din partea dumneavoastră capacitate de reacţie rapidă şi nervi tari. Scopul jocului este de a depăşi cu micuţa broscuţă verde a perestroicii (GORBI) multe obstacole mici, mobile şi de a aduna puncte. Cu cît creşte nivelul de joc, creşte şi gradul de dificultate şi dintr-o dată apar broaşte uriaşe, urîte şi roşii (STALINOS) care vă vor viaţa. Perestroika, împreună cu Commander Keen şi Shooting Gallery, sînt considerate cele mai bune jocuri shareware ale anului 1990. Pe cînd la noi...? Jocul poate fi comandat sub codul S666 de la Software IHEK Rubröder, Ermener Weg 3, D-W4503 Dissen, Fax + Tel. 05421 - 2231, ptr. circa 12 DM.

Creionul electronic ca dispozitiv de intrare.

notepad-urile nu vor avea nevoie de tastatură sau mouse.

În curînd: Notepad + Pen

După laptop şi notebook, un nou tip de calculatoare se pregătesc să intre în competiţie: notepad-urile. Cu greutăţi maximale de 2 kg, avînd dimensiunea ideală de mărimea unei foi A4, cu o înălţime care în cele mai avantajoase cazuri nu va depăşi 2 cm. Aceste dimensiuni devin posibile prin harddisk-uri din ce în ce mai mici; cele mai noi produse Quantum de ex. au o înălţime de numai 1cm, la o capacitate de 40 Mbyte.

Revoluţionară este (va fi) interfaţa om-calculator: "paperlike interface". Ca pe o hîrtie, utilizatorul scrie cu creionul electronic (stylus) pe o tabletă digitizoare transparentă, montată deasupra displayului propriu-zis, realizat cu cristale lichide. Tableta are un strat care permite determinarea exactă a poziţiei actuale a creionului în momentul cînd se apasă cu acesta asupra tabletei. Drumul parcurs de creion pe tabletă este afişat - asemănător scrisului normal - pe display. IBM apreciază că într-o zi acest gen de comunicare om-calculator va fi la fel de important cum este în prezent sistemul cu tastatură şi mouse.

Microsoft a făcut recent cunoscută o listă de producători de calculatoare pentru a semnaliza proiectanţilor de soft că pentru "Pen Windows" există o largă bază hardware. Un total de 21 de producători de calculatoare sau echipamente periferice şi-au făcut cunoscut sprijinul pentru noua suprafaţă - sistem de operare a Microsoft-ului. Lista conţine aproape toate "numele" din extremul orient, printre ele Toshiba, NEC, Sharp, Mitsubishi, Sanyo, Seiko, Epson, Kyocera, Fujitsu şi Samsung. Canon, Hitachi şi OKI plănuiesc dezvoltarea de 'creioane" pentru Pen Windows. NCR, Wang, Grid precum şi producătorii de periferice Calcomp şi Summagraphics şi-au declarat deasemenea sprijinul. Lipsesc de pe lista Microsoft giganţi ca IBM, Compaq şi Hewlett-Packard.

60.000 DM deja pentru începători

Salariile din industria calculatoarelor cresc vertiginos. Absolvenţii unei şcoli de profil n-ar trebui să semneze contracte de muncă cu un salariu anual mai mic de 60000 DM. Salariul mediu pentru un începător se afla la începutul anului 1991 la 63800 DM, conform unei anchete efectuate de întreprinderea de consultanţă Interconsult din Germania.

În total 116 întreprinderi din industria electronică sînt chestionate anual de Interconsult, din Vaihinger, în legătură cu mărimea salariilor pe care le plătesc. La ancheta din ianuarie şi februarie 1991 au participat 116 întreprinderi din industria electronică, dintre care 96 erau producători iar 20 distribuitori.

În urma chestionarului a rezultat că salariile din industria calculatoarelor vor creşte şi în continuare în favoarea angajaţilor. Conducătorii afacerilor caselor de hardware şi software PC au obţinut, conform aşteptărilor, salarii de vis. Salariile lor au crescut faţă de anul anterior cu 8000 DM ajungînd la 262000 DM anual.

Cîştiguri mai mari pentru superiori

Ancheta celor din Vaihing s-a terminat cu rezultate îmbucurătoare şi pentru absolvenţii de licee şi universităţi. Chiar şi începătorii în meserie au cîştigat la începutul lui 1991, în medie, mai mult de 60000 DM. Chiar şi în anul precedent această limită a fost depăşită şi s-a ajuns la 60500 DM salariu anual, iar în 1991 media salariului anual va ajunge la 63800 DM.

Şansa unui începător de a cîştiga în anii următori mai mult decît 60000 DM anual depinde în mare măsură de capacitatea lui de a ajunge după un timp superiorul unor colegi din echipă. Acest lucru se poate observa la prima vedere din tabelele furnizate de Interconsult.

Cei care lucrează de ex. în ramura sistemelor cu microprocesoare, ca ingineri proiectanţi, se pot aştepta la un salariu anual între 60000 DM şi 65000 DM, dar acumulînd experienţă în domeniu, în cîţiva ani, pot să obţină cu 20000 - 30000 DM mai mult. Dacă în decursul anilor începătorul devine şef de secţie, şi va avea deci o funcţie de răspundere, atunci saltul va putea fi mai mare.

Un inginer proiectant care concomitent este şi şeful unei secţii cu cel puţin 6 ingineri, va putea cîştiga cel puţin dublul salariului său de începător.

În general în industria electronică s-au încetăţenit o serie de prestaţii secundare care se plătesc în afara salariului tarifar negociat de sindicate. Astfel 88% dintre producători şi 71% dintre distribuitori plătesc pentru angajaţii lor fonduri de pensii care le va asigura o pensie suplimentară pe lîngă pensia asigurată legal.

Asigurări sociale

Şi în cazuri de îmbolnăviri angajaţii din industria electronică sînt asiguraţi mai bine decît prevede legea. 88% dintre întreprinderi garantează plătirea în continuare a salariului peste termenul limită de 6 săptămîni prevăzut de lege, unele chiar pînă la 2 ani.

Cca. 96% dintre cei care se ocupă de relaţiile întreprinderii cu exteriorul sînt cuprinşi în această reglementare; în plus aceştia sînt asiguraţi pentru caz de deces sau de invaliditate, cel mai adesea cu poliţe de asigurări pe viaţă, care se ridică la 100000 DM şi care, de regulă, sînt valabile 24 ore/zi şi nu doar pe timpul serviciului.

Concediul anual variază între 29 şi 32 de zile, la fel ca în cele mai multe ramuri ale industriei.

(I. M.)

Salarii
Sisteme dotate cu microprocesor

Servicii interne (post de lucru fix)	mii DM/an
Inginer proiectant, absolvent	60-65
Inginer proiectant, 2-4 ani vechime	71-83
Inginer proiectant, vechime > 4 ani	85-94
Şef echipă proiectare cu 2 sau mai mulţi tehnicieni	95-119
Şef proiectare cu mai mult de 6 ingineri	123-146
Inginer de aplicaţii, pînă la 3 ani vechime	77-93
Inginer de aplicaţii, mai mult de 3 ani vechime	98-120
Inginer producţie-marketing, pînă la 3 ani vechime	87-103
Inginer producţie-marketing, mai mult de 3 ani vechime	111-129
Director producţie-marketing (director tehnic)	125-143
Servicii externe (muncă de teren)	
Inginer expert într-un domeniu, pînă la 3 ani experienţă în producţie	76-94
Inginer expert într-un domeniu, mai mult de 3 ani experienţă în producţie	100-118
Inginer comercial, pînă la 3 ani în serviciul extern	102-111
Inginer comercial, mai mult de 3 ani în serviciul extern	115-129
Şef comercial zonal, şef al unui inginer	137-160
Şef comercial zonal, şef a cel puţin 3 ingineri	167-224
Director	238-304

Oameni şi şoareci

Recent, mouse-ul, fără de care lucrul cu un PC a devenit de neconceput, şi-a sărbătorit a 28-a aniversare. Motiv suficient pentru a scotoci puţin prin istoria lui.

Şoarecii timpurii

"În vremea aceea, în urmă cu 28 de ani, experimentam mai multe aparate de introducere date. De fiecare dată mouse-ul s-a dovedit a fi foarte folositor, astfel încît comparîndu-l cu alte instrumente de introducere, a cîştigat întrecerea". ne spune Doug Engelbart, inventatorul mouse-ului.

Prototipul din lemn al primului mouse, pe care l-a creat Doug Engelbart la Institutul de Cercetări Stanford, în 1963, a fost conceput pentru calculatorul Augment. Ulterior diferite generaţii de PC-uri s-au lăsat influenţate de ideile lui Engelbart, începînd cu Star-ul firmei Xerox, prin Lisa lui Apple şi pînă la Macintosh-urile firmei Apple. Dar Engelbart nici nu şi-ar fi putut închipui ce avînt va lua invenţia sa.

Mouse-ul inventat de Engelbart era un simplu instrument de introducere, analogic, care transmitea soft-ului cîte un semnal la fiecare mişcare a mouse-ului, influenţînd mişcarea cursorului. În interiorul acestui corp de lemn se aflau două rotiţe metalice, legate între ele prin două rezistenţe variabile.

Dar ideea de a lucra cu mouse-ul a devenit cunoscută abia în momentul în care, la începutul anilor 70, cunoscutul centru de cercetări Palo Alto Research Center (PARC) al firmei Xerox l-a însărcinat pe Jack S. Hawley să construiască prima variantă digitală a "şoricelului" lui Engelbart.

În paralel Xerox-ul a dezvoltat calculatorul Alto care urma să fie livrat împreună cu mouse-ul respectiv. Cu toate că din acest tip de calculator au fost vîndute mai puţin de 100 de exemplare, drumul pentru dezvoltarea calculatoarelor personale şi al mouse-urilor a fost deschis.

La cererea firmei Xerox, Hawley elaborează în 1975 un nou standard de mouse, care a fost adoptat de mulţi producători şi care s-a menţinut mult după anii '80. În continuare Hawley lucrează la propria sa firmă "Mouse House" din Berkeley (California), la configurarea şi producerea de mouse-uri.

După ce mouse-ul firmei Xerox a trezit tot mai multă atenţie, Microsoft a hotărît să creeze un mouse propriu. Între timp la Microsoft a ajuns un fost colaborator al firmei Xerox, pentru a permite operarea la un nou produs, Microsoft Word, cu ajutorul mouse-ului. În acelaşi timp Bill Gates, Paul Allen şi Raleigh Roark discutau extinderea paletei de produse Microsoft cu un produs hardware adecvat. Astfel s-a pus baza primului mouse Microsoft.

Dintr-un boţ de lut

Cu concepţia primului mouse Microsoft a fost însărcinat un designer, care a construit conform dorinţei colectivului Microsoft, un model din lut care era prevăzut în partea de dedesubt cu pioneze care înlocuiau partea glisantă.

Strămoşul din lut/lemn

Raleigh Roark îşi aduce aminte de prezentarea acestui mouse: "Unii dintre noi stăteau ceasuri în şir la masa de conferinţe, ocupaţi cu mişcarea înainte şi înapoi a acelei grămăjoare de lut, pentru a afla dacă ne place sentimentul pe care ni-l dă această mişcare. Dar nici unul dintre noi nu era pe deplin mulţumit, astfel încît i-am schimbat înfăţişarea şi dimensiunile după imaginaţia noastră. Cu un nou model de lut în mînă am zburat apoi la Tokio pentru a da "chestia" unui producător de electronică s-o construiască".

Roark a fost însoţit de Kay Nishi, care pe atunci era vicepreşedintele Microsoft şi preşedinte al comisiei ASCII din Japonia. Nishi şi Roark s-au întîlnit acolo cu producători de aparatură electronică pentru a discuta ideile lor. Discuţiile s-au sfîrşit

repede şi brusc, producătorii n-au crezut că ideea celor de la Microsoft ar fi realizabilă. Ei au avut rezerve că unitatea de codificare ar putea încăpea în acea carcasă minusculă.

Roark povesteşte: "s-a discutat mult, de ce acest lucru ar fi imposibil, important era doar că nu se putea. Pînă cînd la un moment dat s-a făcut o linişte deplină şi inginerul şef ne-a comunicat că se va retrage timp de o oră, după care ne va da o soluţie. Într-adevăr inginerii au revenit cu o propunere de carcasă care avea capacitatea de a funcţiona, iar cîteva luni mai tîrziu Microsoft poseda primul mouse".

Prima generaţie

"Crearea mouse-ului serial era pentru mine o problemă captivantă, o deschidere a concepţiilor, nerealizată de nimeni pînă atunci", spune Raleigh Roark conducătorul secţiei de dezvoltare pentru un mouse serial a Microsoft-ului.

În 1983 Microsoft prezintă un nou produs pentru IBM-PC, denumit Microsoft busmouse, un mouse mecanic avînd două taste, sprijinindu-se pe o sferă de oţel şi cîteva role, care înregistrau mişcările mouse-ului pe orice suprafaţă. "Îndemînarea" lui era dictată de o cartelă de 8 biţi, care era echipată cu un cip Intel 8255, ca interfaţă periferică programabilă, şi cu cîteva cipuri auxiliare. Acest tip de mouse avea avantajul că nu consuma multă energie electrică şi deci nu avea nevoie de o sursă distinctă de alimentare.

La un an de la apariţia busmouse-ului Microsoft a conceput o variantă serială de mouse, atingînd un stadiu tehnologic avansat, deoarece putea fi cuplat direct la interfaţa serială RS232. Acesta nu avea nevoie nici de o placă de bus nici de alimentare separată, deoarece prin intermediul unui procesor CMOS integrat putea aborbi energia necesară prin interfaţa RS232.

Odată cu apariţia, în anul 1983, a sistemului de operare MS-DOS 2.0, mouse-ul a avut un cuvînt important de spus. MS-DOS era echipat cu un aşa-zis driver de periferic instalabil la dorinţă, ceea ce simplifica foarte mult configuraţia oricărui calculator echipat cu sistemul de operare DOS şi cu un mouse.

În 1985, la apariţia AT-urilor IBM, odată cu creşterea numărului de cartele grafice de înaltă rezoluţie s-a ţinut cont şi de cele două noi versiuni de drivere.

Generaţia a doua

Mouse-ul firmei Microsoft cu o rezoluţie de 200 puncte per inch a schimbat modul de utilizare al primului mouse. Dublarea rezoluţiei a avut ca efect faptul că utilizatorul nu mai trebuia să plimbe mouse-ul pe întreaga masă de scris pentru a mişca cursorul pe ecran.

Steve Shaiman proiectantul principal al softului pentru mouse-ul 5.0 al firmei Microsoft ne spune: "În 1985 a fost prezentată o altă variantă a mouse-ului Microsoft. Modificările erau observabile la prima vedere: culoarea gri a tastelor care atingeau acum marginile, carcasa nou concepută şi bila îmbrăcată în cauciuc în locul celei îmbrăcate în oţel. Adevărata diferenţă însă s-a simţit, nu s-a văzut. Cu dublarea rezoluţiei la 200 dpi (dot per inch - puncte pe inch), mouse-ul putea fi manevrat mult mai uşor, necesita o rază de acţiune mult mai mică pentru deplasări (aprox. 10-12 cm); cele mai multe probleme puteau fi rezolvate rapid şi cu economie de mişcare. În comparaţie cu acesta modelul anterior părea greoi şi grosolan, mai ales datorită razei mai mari de acţiune (aprox. 20-25 cm).

În luna mai 1986 Microsoft a ieşit pe piaţă cu un mouse care era o versiune modificată a interfeţei busmouse, bazat pe aşa-numitul cip "Inport". Deoarece noul driver de mouse folosea sistemul de întreruperi programabil al acestui cip, viteza de reacţie a mouse-ului a putut fi ridicată vizibil.

Generaţia a treia

"Noul mouse Microsoft (care arată ca o bucată de săpun), cu noua sa bilă poziţionabilă şi cu mecanica îmbunătăţită este cu adevărat diferit de predecesorul său". afirma John C. Dvorak, în PC Magazine, 22.12.1987.

Mouse-ul celei de-a treia generaţii a fost prezentat de Microsoft în 1987. Corpul devenise mai suplu, mai îngust iar clicul tastelor mai plăcut. Arhitectura internă a acestui mouse tot se mai orienta încă după cea a mouse-ului cu taste gri. Cîteva modificări esenţiale au dus la o controlabilitate mai uşoară; de ex., poziţionarea bilei (care înainte se afla în spate) în faţa mouse-ului şi mărirea tastei din stînga faţă de cea din dreapta.

Generaţia a patra

Actualul mouse, prezentat în 1989, de culoare albă şi cu o rezoluţie de 400 dpi aparţine generaţiei a patra. Aspectul este similar cu cel din generaţia a treia, dar tehnologic se află pe un plan net superior. Importante sînt următoarele aspecte: rezoluţia net îmbunătăţită pe de o parte, iar pe de altă parte sensibilitatea şi viteza reglabilă. În plus în carcasa mouse-ului a fost introdusă electronica de control pentru mouse-ul serial PS/2, pentru a optimiza manipularea produsului.

"Odată cu apariţia soft-urilor tot mai complexe va trebui să avem tot mai multe aparate de introducere eficiente în lucrul cu calculatorul. În viitor va exista în mod cert un mouse". - spune Cary Lu, autorul cărţii "The Apple Macintosh Book", ediţia a treia, Microsoft Press.

Astăzi deja un soft cu suprafaţă utilizator grafică este mai degrabă o regulă decît o excepţie, şi în paralel cu această dezvoltare acceptarea mouse-ului se extinde continuu:
Bun venit în era şoarecilor !

(I. M.)

De la 80386 la 80486

PC-urile cu 80486 nu sînt chiar cele mai ieftine. Acest articol încearcă să explice de ce costă mai mult decît PC-urile cu 80386 şi care sînt avantajele noului procesor.

Pînă la apariţia lui 80486 fiecare nou procesor Intel (8086, 80286, 80386) era o veritabilă construcţie nouă. Comenzile fiecăruia menţineau compatibilitatea cu procesoarele anterioare, deci programele scrise pentru tipuri de procesoare mai vechi sînt funcţionale şi pe cele noi. Cu totul altfel se prezintă situaţia la procesorul 80486 care nu este o construcţie cu totul nouă: într-un singur cip el conţine toate componentele pe care le conţinea predecesorul său, procesorul 80386, coprocesorul 80387 şi controller-ul cache 82385.

80386 posedă, în comparaţie cu predecesorii săi, avantaje importante care în ciuda "folosirii sale îndelungate" (din 1985 şi pînă astăzi) nu au dus la necesitatea unei construcţii noi. El poate fi folosit la o frecvenţă de tact de 33 MHz şi se distinge prin viteza sa mare de calcul. Tactul maximal permis pe magistrală este de 66 MHz.

Lăţimea structurii componentelor realizate prin aşa-numitul procedeu CHMOS-III este de numai 1.5 μm, ceea ce înseamnă că fiecare conductor şi spaţiile dintre conductori au o lăţime de numai 0.0015 mm. Din acest motiv pentru producerea lor sînt necesare aparate foarte precise. Cipul conţine 275000 tranzistori. Carcasa, aşa-numită Pin-Grid, conţine 132 de pini şi are dimensiunile de 36 x 36 mm.

Procesorul i486 conţine aproape de patru ori mai multe tranzistoare (1.2 milioane) şi din acest motiv este fabricat prin procedeul 1-μm-Standard-CHMOS-IV (conductorii şi spaţiile dintre ei au o lăţime de doar 0.0001 mm). Carcasa din ceramică are dimensiunile de 44.8 x 44.8 mm şi posedă 168 pini de legătură în partea inferioară.

În concepţia procesorului 486 au fost preluate structurile de bază ale fiecărui procesor în parte (80386, 80387 şi 82385), dar ele au fost şi modificate pe alocuri. Astfel partea de procesor şi coprocesor au fost legate printr-o magistrală dublă de 32 de biţi, dispărînd astfel "locul îngust" existent pînă acum între cele două procesoare. Aşa se explică şi faptul datorită căruia un 80486 lucrează mai rapid decît un 80386 în tandem cu un 80387.

O particularitate a lui 80386 a fost păstrată şi la 80486. Unitatea internă de management a memoriei (MMU - Memory Management Unit), uşurează sub MS-DOS, dar mai ales sub Unix, lucrul cu pachete de programe mari. Ea controlează atît spaţiul de memorie fizic cît şi pe cel virtual.

Mărimea memoriei fizice atît a procesorului 80386 cît şi a lui 80486 este dată de cei 32 de conductori de adresare şi este de 4 GBytes. Spaţiul logic (virtual) de adrese este de 64 TBytes (1 Terabyte = 1000 Gigabytes). Nici 4 GBytes aproape că nu pot fi folosiţi, darămite 64 TBytes !

Spaţiul de memorie virtual se împarte în mai multe pagini de cîte 4 KBytes. Aceasta este cea mai mică unitate care poate fi transferată. Acest lucru se poate observa foarte bine cu sistemul de operare MS-DOS, un PC 386 şi Windows 3.0. Această suprafaţă utilizator rezervă pe harddisk o aşa-numită zonă de swapare ("Swap Area"), ce corespunde ca mărime cu mărimea memoriei virtuale. Cu cît este mai mare această zonă, cu atît este mai mare şi memoria virtuală care stă la dispoziţia programului.

Pentru a putea utiliza un 80386 la viteza sa maximă de lucru este necesar ca şi periferia să fie tot la fel de rapidă. Din această cauză se încearcă tot felul de trucuri, pentru a intercala cît mai puţine cicluri de aşteptare (waitstates). Unul dintre aceste trucuri este aşa-numitul "Pipelining", în cazul căruia unitatea de management a memoriei foloseşte semitacturile pentru a determina în timp util adresele care urmează. De obicei procesorul află adresa şi definiţia următorului ciclu de bus înainte de prelucrarea ciclului în curs, fapt care îi permite să trimită deja noile adrese pe bus (magistrală). Periferia are astfel timp suficient pentru prelucrare şi va preda datele abia după trei perioade de tact, fără a frîna procesul.

Utilizarea memoriei cache (tampon) la sistemele 80386 place în continuare. Nu este importantă numai mărimea ei ci şi modul de administrare. În această memorie se scriu datele care sînt necesare mai la urmă. Deoarece cel mai adesea un program nu "ţopăie sălbatic" prin tot domeniul de adrese, probabilitatea ca datele din memoria cache să fie necesare de mai multe ori este foarte mare. Această probabilitate, în funcţie şi de modul de administrare, este între 89% şi 97%.

În procesorul 80486 este integrată o memorie cache de 8 KBytes. Administrarea ei se face prin intermediul controller-ului de pe cip, formatul cache avînd chiar şi o lăţime de 128 biţi. Cu toate că cipul 80486 înglobează deja un coprocesor, unele plăci dau posibilitatea instalării unui coprocesor suplimentar Waitek 4167. Programele de calcul intensiv care recunosc acest coprocesor rulează de două, trei ori mai rapid.

Pentru măsurarea vitezei reale a unităţii centrale (CPU - Central Processor Unit) se foloseşte testul Dhrystone, iar pentru măsurarea vitezei cu un coprocesor se foloseşte testul Whetstone. Ambele sînt teste ("Benchmarks") standardizate şi conţin un spectru prestabilit de comenzi de procesare.

O comparaţie între rezultatele testelor pentru un 80486 şi un 80386 (cu 80387) ne arată că primul are performanţe mult mai ridicate, fapt datorat organizării sale optimizate.

(I. M.)

Să-l muşti ...

Macintosh înseamnă o adiere de rafinament şi confort ales (de utilizare a unui calculator) la preţuri medii, acceptabile. Citiţi care Mac vi s-ar potrivi cel mai bine.

Faptul că începătorii în ale calculatoarelor îşi încep cariera lor de utilizatori nu cu puţină teamă, la 10 ani de la introducerea calculatoarelor IBM-PC (1982), nu este fără motiv. Căci ca şi pînă acum cei mai mulţi utilizatori sînt nevoiţi să utilizeze sistemul de operare MS-DOS şi comenzi neplăcute de genul: "XCOPY /A/E/P", "GRAFTABL", sau "EXE2BIN".

De mult există însă o alternativă: deja în 1984 constructorul californian de calculatoare Apple a prezentat un concept de calculator cu un confort utilizator revoluţionar, şi anume Macintosh. Un posesor de Mac nu trebuie să facă risipă nici de acrobaţie mentală şi nici de răbdare pentru a descoperi secretele utilizării calculatorului, ci se poate concentra imediat după pornirea lui asupra muncii propriu-zise şi chiar şi cel mai neştiutor dintre toţi poate lucra după 1/2 oră cu un Macintosh.

Ceea ce face Macintosh-ul atît de atractiv, (Macintosh este şi numele unui sort de mere), este sistemul său grafic de manipulare şi anume "Finder". Indiferent că se doreşte lansarea în execuţie a unui program, ştergerea sau copierea unui fişier, totul se rezolvă printr-un simplu clic de mouse şi prin simboluri uşor de înţeles. Finder permite o utilizare nu numai uşoară ci şi unitară. Toate programele pot fi deservite după aceeaşi schemă. Dacă utilizatorul cunoaşte un program, atunci le cunoaşte pe toate.

Un alt atu: toate calculatoarele Macintosh stăpînesc aşa-numitul multitasking. "Multi" înseamnă "mai multe", "task" înseamnă "sarcină"; calculatoarele care permit multitasking-ul pot prelucra (rezolva) mai multe sarcini în paralel, mai multe programe putînd fiind rulate simultan.

Avantajul: cînd utilizatorul scrie, de ex., o scrisoare unui client şi are nevoie de informaţii despre partenerul de afaceri, poate comuta rapid într-un program bază de date, poate obţine datele despre client şi se poate întoarce în programul de prelucrare de texte - chiar şi schimbul de date între aplicaţii este posibil în acest mod. Cele mai multe PC-uri nu pot face acest lucru nici pînă în ziua de azi. (Odată cu apariţia suprafeţei utilizator MS-Windows 3.0, pe PC-uri lucrurile s-au mai schimbat.)

Pentru începători: Macintosh-Classic

Cine nu s-a născut ca şi copil al unor părinţi bogaţi, mulţi ani nu putea decît să privească cu jind la posesorii de Macintosh-uri: calculatoarele Apple erau pentru majoritatea cumpărătorilor mult prea scumpe.

Totuşi începînd din octombrie, anul trecut, odată cu apariţia noului Macintosh Classic situaţia s-a schimbat. Un Classic fără harddisk (1 MByte memorie principală şi un lector de dischete de 3,5") se poate obţine deja cu 2000 DM, iar unul cu un harddisk de 40 MBytes poate fi cumpărat cu 3000 DM (incluzînd 2 MBytes RAM şi un lector de dischete).

Noul Apple înlocuieşte Mac-urile compacte Macintosh SE şi Macintosh Plus. În exterior li se aseamănă amîndorura, tehnic însă este urmaşul lui SE: Apple construieşte ambele calculatoare constitutiv din aceleaşi componente. Totuşi placa de bază a lui Classic este numai pe jumătate atît de mare cît cea a lui SE - microminiaturizarea electronicii face acest lucru posibil.

Macintosh Classic este acţionat de un procesor Motorola 68000 cu un tact de 8 MHz. Puterea de calcul a lui Classic este suficientă pentru aplicaţii simple, pentru calcule complexe nu ajunge însă (cum ar fi de ex. pentru aplicaţii grafice profesionale sau pentru gestiunea bazelor de date mari).

Classic este o opţiune potrivită pentru începători, pentru lucrul la domiciliu, sau pentru lucrări care nu necesită nici un tempo de calcul intensiv şi nici o prezentare color. Punctele sale forte:

- procesarea de texte
- gestiunea bazelor de date mici (de exemplu cu adrese şi numere de telefon) şi
- calculul tabelar de dimensiuni mici

Numele de un milion de dolari

Atunci cînd şeful firmei Apple John Sculley a vrut să boteze noul Macintosh "Classic" a trebuit să constate că acest nume era protejat deja pentru o altă firmă. Un milion de dolari a trebuit să plătească Apple din această cauză filialei Mercedes "Modular Computer Systems" din însorita Florida, pentru a-şi putea boteza noul produs "Classic". Ironie a sorţii: Classic a fost (şi este în SUA) şi numele unui sort de Coca-Cola. Şeful firmei Apple John Sculley şi-a început cariera managerială la Pepsi-Cola şi şi-a făcut un bun nume în ramură ca expert în marketing.

- De asemenea Classic este indicat pentru a face cu el primii paşi în domeniul aplicaţiilor grafice (desen, pictură, publicistică).

Un avantaj enorm al lui Classic în comparaţie cu majoritatea calculatoarelor compatibile IBM este faptul că poate fi cuplat într-o reţea ("Appletalk"), fără ca pentru aceasta să fie necesare investiţii hard semnificative. Electronica necesară este deja construită de Apple în Classic (ca la fiecare Macintosh). Utilizatorul nu trebuie să cumpere în plus decît un cablu "Localtalk" (preţ: 160 DM) şi software-ul de comandă corespunzător.

Alte posibilităţi de conectare: micuţul Apple are (din nou ca toate Mac-urile) o interfaţă SCSI (Small Computers System Interface - un standard de interfaţare), la care pot fi conectate simultan pînă la şase periferice (de ex. un harddisk suplimentar şi unităţi de salvare/restaurare pe/de pe bandă magnetică).

Dacă lectorul de dischete încorporat (pentru dischete de 3,5" cu o capacitate de 1,44 MBytes) nu este suficient, Classic are o mufă la care mai poate fi conectat un lector de dischete extern.

Macintosh Classic este compact şi are dimensiuni reduse, avantajul: spre deosebire de majoritatea calculatoarelor MS-DOS el poate fi aşezat pe masa de scris şi mai rămîne loc şi pentru acte şi cărţi. Un alt avantaj: deoarece este atît de mic şi de uşor, şi întrucît are integrat în carcasă un mîner, poate fi foarte uşor transportat.

Dezavantaj: deoarece monitorul monocrom este înglobat în carcasă, trebuie să te descurci cu o diagonală a ecranului de doar 9" (23 cm) - pentru comparaţie: diagonala unui monitor normal măsoară cel puţin 12" (30 cm). Consolarea: cu toate că monitorul lui Classic este mic, textele şi imaginile sînt afişate cu rezoluţia ridicată de 342 de linii a cîte 512 puncte fiecare..

O altă lipsă a lui Classic: el nu are nici un slot (mufă de extensie) liber pentru plăci suplimentare cu ajutorul cărora i-ar putea fi extinse capacităţile (de ex. pentru grafică color).

Există un singur slot pentru extensia de memorie numit "systems memory slot". La Classic-urile care au memoria principală de doar 1 MByte, în acesta se poate introduce o cartelă RAM Apple specială care să ridice memoria la 2 MBytes (preţ: cca. 600 DM). Această cartelă mai conţine şi socluri pentru cipuri de memorie, aşa-numite SIMM (Single In line Memory Module) cu ajutorul cărora memoria poate fi extinsă pînă la 4 MBytes. (preţ pe MByte: cca. 150 DM).

Inginerii de la Apple au dotat Classic-ul şi cu o intrare audio la care poate fi conectat un microfon sau o instalaţie stereofonică, pentru a introduce în calculator mesaje verbale, muzică sau sunete, a le prelucra şi a le reda apoi.

Calculatorul: Macintosh SE/30

El este tot atît de mic ca şi modelul Classic, dar în interiorul lui mocneşte puterea unui uriaş: Macintosh SE/30 (preţ: începînd de la 7600 DM) lucrează cu rapidul procesor 68030 (momentan cel mai rapid procesor Motorola aflat pe piaţă) cu un tact de 16 MHz, căreia i se adaugă în cazul calculelor deosebit de complicate şi coprocesorul 68882. Programele de calcul tabelar care recunosc coprocesorul 68882 lucrează cu acesta de cca. 100 de ori mai rapid. El mai este echipat cu o memorie de 2 MBytes, un harddisk de 40 MBytes şi un slot (aşa numit "processor direct slot", mufă cu acces direct la procesor).

Mac SE/30 a fost primul calculator Apple cu Procesor Direct Slot. Plăcile de extensie introduse în acest slot pot accesa direct rapidul procesor 68030.

Modelul SE/30 este recomandat pentru utilizatorii care au nevoie de o putere de calcul ridicată, dar care se pot lipsi de un monitor mare şi de o reprezentare color. Se recomandă pentru:

- procesare de texte
- baze de date mari şi
- programe de calcul tabelar costisitoare, dar şi ca
- server (calculator principal) într-o reţea.

Cine doreşte să realizeze pe SE/30 grafică la nivel profesional

Operare uşoară: Finder

Calculatoarele Macintosh sînt cele mai prietenoase calculatoare din lume în privinţa operării şi acest lucru se datorează suprafeţei grafice Finder (se pronunţă Fainder). După un test al Institului Fraunhofer din Stuttgart, din punct de vedere al operării calculatoarele compatibile IBM nu pot da clasă unui Mac nici chiar cu suprafaţa utilizator Windows.

Inima operării sub Finder: mouse-ul. Dacă, de ex., cursorul este deplasat deasupra simbolului harddisk-ului şi se apasă de două ori tasta mouse-ului se deschide o fereastră cu directorul harddisk-ului.

Chiar dacă nu este la fel de uşor să foloseşti un IBM PC cum este de uşor să foloseşti un Mac, IBM PC prezintă totuşi un avantaj uriaş: pentru ele există cea mai bogată ofertă de software.

Atunci cînd utilizatorii de Mac-uri vor să aibă acces la această ofertă (pentru că au nevoie de ex. de o anumită aplicaţie) apare o problemă: Mac-urile nu pot prelucra programele scrise pentru MS-DOS. Motivul: calculatoarele Apple utilizează un alt sistem de operare (Finder) decît PC-urile compatibile IBM (MS-DOS).

O ieşire din impas există doar pentru posesorii de Mac-uri din puternica serie II (IIsi, IIci şi IIfx) şi de SE/30. Aceştia pot utiliza programul de emulare Soft-PC (900 DM) care emulează un PC/XT şi pot rula astfel toate programele MS-DOS.

va trebui să înlocuiască monitorul de 9" cu un monitor mai mare. Există producători care oferă plăci de extensie cu ajutorul cărora se pot conecta la SE/30 monitoare mari care să poată prezenta una sau chiar două pagini DIN A4.

SE/30 este simpatizat mai ales în SUA şi Japonia. Motivul: în ambele ţări birourile uriaşe sînt divizate în celule de lucru minuscule. Fiecare angajat dispune de un minibirou cu o suprafaţă de cca. doi pe doi metri şi de o micuţă masă de scris. Un PC normal nu prea se potriveşte înăuntru - în asemenea locuri este necesar un SE/30 care ocupă cel mult suprafaţa unei mape de scris DIN A4 şi care dispune totuşi de o putere de calcul maximală.

Curcubeu ieftin: Macintosh LC

Cu Macintosh LC Apple oferă în sfîrşit cumpărătorilor un model color plătibil ("LC" semnifică "Low cost Color", în româneşte "color ieftin", dar ieftin în sensul că merită banii). Preţul unui LC cu monitor de 12": 4900 DM.

Adaptorul grafic care comandă monitorul este înglobat pe placa de bază (motherboard). Cu monitorul Apple color de 12" (rezoluţie 384x512) LC poate reprezenta imagini în 256 de culori, cu un monitor color de 13" (rezoluţie 480x640) poate reprezenta imagini în 16 culori.

Mai multe culori pot fi obţinute doar prin adăugarea unei plăci de extensie suplimentare, o aşa-numită cartelă "Video-RAM", care conţine mai multă memorie pentru grafice; cu ajutorul ei pot fi obţinute 32000 de culori cu un monitor de 12" şi 256 de culori cu unul de 13".

Înafară de aceasta un LC mai conţine un procesor 68020 (tact 16 MHz), 2 MBytes RAM (maxim 10 MBytes) şi un harddisk de 40 MBytes.

Cu această dotare el poate prelucra texte, fişiere şi calcula tabele de două ori mai rapid decît un Classic, dar nu atinge decît cca. 60% din puterea unui SE/30.

La intrarea audio a lui Macintosh LC poate fi cuplat un microfon (livrat de asemenea de Apple) sau de exemplu un recorder CD (Compact Disc) şi astfel pot fi digitalizate vorbire, muzică şi zgomote, pot fi prelucrate şi apoi redate. De asemenea textele şi datele pot fi însoţite de comentarii vorbite, pentru a da de exemplu indicaţii colegilor în timpul lucrului cu un program.

Deoarece fişierele de sunete necesită foarte mult spaţiu, Apple oferă un program special de comprimare, cu ajutorul acestuia pe harddisk-ul de 40 MBytes pot fi memorate înregistrări de pînă la 30 de minute.

Dezavantajul lui LC: are un singur slot liber pentru extensii hard. Consolare: deoarece acest slot (mufă de extensie) este identic cu Processor Direct Slot al lui Mac SE/30 poate fi folosită întreaga gamă de cartele de extensie disponibilă pentru SE/30.

Macintosh Portable

Macintosh Portable este un calculator pentru călătorii. El dispune de o memorie de 2 MBytes şi de un harddisk de 40 MBytes. Mouse-ul tipic Mac este înlocuit cu o aşa-numită "trackball" înglobată în tastatură (o bilă prin mişcarea căreia poate fi deplasat cursorul pe ecran). Inginerii de la Apple sînt atenţi la detalii: pentru stîngaci bila poate fi montată în stînga tastaturii.

La fel de practic: cu puterea acumulatorului său, Mac Portable este economicos ca un scoţian - el poate funcţiona pînă la 10 ore independent.

Monitorul cu cristale lichide cu iluminare din fundal este de calitate ridicată: imaginile sînt construite rapid, contrastul este optim.

Cel mai mare neajuns este greutatea ridicată (7,2 kg).

Cu facilităţile sale color limitate şi viteza sa medie de calcul Macintosh LC este un calculator indicat pentru şcoală, universitate şi birou. Se recomandă pentru:

- procesare de texte
- baze de date
- calcul tabelar şi
- grafică de prezentare

A începe pentru a urca: Macintosh IIsi

Ele oferă un tempo de calcul ridicat şi o grafică excelentă - aşa-numita serie constructivă modulară (Macintosh IIsi, IIci şi IIfx). Aceste Mac-uri sînt modulare deoarece permit utilizatorului să şi le adapteze după necesităţi cu ajutorul unor plăci de extensie (module).

Cel mai mic calculator din seria modulară este Macintosh IIsi (cu monitor color de 13" începînd de la 10500 DM), calculator adus pe piaţă de Apple în octombrie.

IIsi lucrează ca şi SE/30 cu un procesor 68030. Tactul său este mai mare decît al lui SE/30 (20 MHz în loc de 16 MHz), dar trebuie să se descurce fără coprocesor şi din această cauză nu atinge decît cca. 90% din puterea acestuia.

El mai are înglobat un adaptor grafic (pînă la 256 de culori cu o rezoluţie de 384X512 pixeli), 2 MBytes RAM şi un harddisk de 40 MBytes. IIsi are un singur slot liber posibilităţile de extensie fiind astfel limitate. Totuşi utilizatorul poate utiliza fie plăci de extensie specifice pentru SE/30 fie cartele "Nubus". Trucul în cazul plăcilor de extensie: ele conţin în plus un coprocesor 68882. IIsi este ca o combinaţie între un Mac LC şi un SE/30:

Apple Story

Este o poveste tipic americană "de la spălător de vase, la milionar": Steve Jobs, un fan al electronicii tînăr şi excentric, împreună cu prietenul său "Woz" Wozniak, un electronist bricoleur genial, au înfiinţat în 1976 firma Apple Computer.

Într-un garaj au construit primul calculator personal din istorie, Apple I, care a avut un succes deosebit. Cu cîştigul obţinut Jobs şi Wozniak au finanţat la sfîrşitul anilor 70 construirea calculatorului Apple II, cu care s-au impus definitiv în lumea calculatoarelor personale. Apple II a fost provizoriu ultimul mare succes pentru Apple: atît urmaşul lui Apple III, cît şi precursoarea Macintosh-urilor Lisa, n-au avut priză la public şi au adus firma Apple în pragul falimentului. Abia cu Macintosh norocul s-a reîntors.

Părinţii Macintosh-ului: Steve Jobs (sus) şi Stephen Wozniak (mijloc). Jos: şeful firmei Apple John Sculley

Şeful firmei Apple Steve Jobs (azi 35 de ani şi şeful firmei Next care face concurenţă lui Apple) şi-a propus în momentul realizării modelului Macintosh să realizeze un calculator cu putere mare de calcul, confortabil în operare, fără compromisuri şi care să încapă pe masa de scris: pe scurt să construiască cel mai bun calculator personal.

După multe săptămîni de muncă de 70 de ore cu muzică rock, suc de fructe proaspăt şi jocuri pe video pentru destindere, constructorii au putut prezenta în 1984 primul calculator Macintosh: "Mac 128". Ceea ce deosebea Mac-ul de toate celelalte calculatoare era Finder: un sistem de operare cu mouse, iconuri (simboluri) şi meniuri. Modelul său: o suprafaţă utilizator pe care Jobs şi "piraţii" (aşa îşi spuneau proiectanţii înşişi) au văzut-o în Xerox PARC (PARC: Palo Alto Research Center).

Datorită faptului că permitea o operare prietenoasă şi că era fiabilă această suprafaţă utilizator a transformat Macintosh-ul într-o legendă vie. Totuşi ea n-a putut afecta PC-urile IBM şi ale altor concurenţi printre altele datorită politicii de preţuri ridicate practicată de Apple. În cele din urmă în 1983 cînd Apple a intrat într-o criză financiară, Steve Jobs la adus la Apple pe John Sculley, pînă în acel moment manager la Pepsi, pentru a repune firma pe picioare. Sculley şi-a dat repede seama de faptul că Apple se afla în criză şi datorită stilului incoerent de conducere al lui Jobs şi l-a deposedat de putere. Cătrănit Jobs a părăsit firma Apple şi a înfiinţat o nouă firmă, Next Computer care produce staţii de lucru (workstation).

John Sculley a renunţat la politica preţurilor ridicate abia în momentul în care în ultimul an procentul din piaţă şi cîştigurile firmei Apple au început din nou să scadă rapid ca urmare a scăderilor de preţuri la calculatoarele compatibile IBM şi a apariţiei suprafeţei utilizator Windows. În plus Apple a lansat pe piaţă trei noi Mac-uri ieftine (Classic, LC, IIsi) şi a scăzut şi preţurile la modelele rămase.

Măsurile luate par să aibă succes: după cum afirmă vînzătorii de Mac-uri, mai ales Classic se vinde ca pîinea caldă. Dar nici în viitor viaţa lui Sculley nu va fi mai liniştită: noua firmă a lui Steve Jobs Next a lansat pe piaţă calculatoare care sînt mai puternice decît calculatoarele Mac II şi care costă sensibil mai puţin.

Apple s-a hotărît să riposteze concurenţei cu un nou Macintosh II cu un procesor Motorola 68040, cu tactul de 25 MHz, de îndată ce acest procesor va fi la dispoziţie în cantităţi suficient de mari.

Se mai vorbeşte şi despre un posibil Macintosh cu procesor RISC, care ar putea fi de mai multe ori mai rapid chiar decît 68040.

Pentru Macintosh Portable va exista în curînd un înlocuitor: deja la finele anului 1990 Apple a încheiat un acord cu experţii japonezi în materie de Laptop-uri Sony şi Toshiba pentru realizarea unui Macintosh Laptop care ar trebui să apară pînă la sfîrşitul acestui an. Acesta nu va fi tot la fel de puternic ca şi Portable, dar va fi mult mai uşor.

un calculator rapid, color. Acestea îl recomandă pentru utilizarea în institute de învăţămînt superior şi la birou. Se recomandă pentru:

- procesare de texte
- gestiunea bazelor de date
- calcul tabelar
- grafică de prezentare
- aplicaţii grafice semiprofesionale (de ex. publicistică).

Artistul: Macintosh IIci

Fratele mijlociu din seria modulară Mac se numeşte Macintosh IIci (preţ: 15150 DM cu monitor color 13"). El este echipat asemănător cu fratele său mai mic, dar procesorul său 68030 lucrează la un tact de 25 MHz, în plus foloseşte un coprocesor 68882 şi are o memorie de 5 MBytes.

Şi în cazul lui "Mac IIci" adaptorul grafic este înglobat pe placa de bază (max. 256 culori). El dispune însă de trei sloturi de extensie.

Cu IIci începe lumea pentru toţi cei care doresc să facă publicistică profesională. Programe de grafică puternice memorează documentele paginate pe dischete astfel încît acestea pot fi preluate şi trimise direct în tipografie.

Condiţia pentru aceasta este o putere de calcul foarte mare. Atunci cînd utilizatorul doreşte imprimarea unei imagini color, calculatorul trebuie să preia aşanumita "separare a culorilor", pentru fiecare pixel al imaginii trebuind transmise culorile constitutive.

IIsi este o alegere potrivită pentru aplicaţii în mediul universitar, la birou sau în edituri. Se recomandă pentru:

- procesare de texte
- gestiunea bazelor de date mari
- grafică de prezentare şi
- aplicaţii grafice (pictură, desen, Desktop Publishing).

Dată fiind puterea sa de calcul el poate fi utilizat, ca şi SE/30, ca server într-o reţea.

Extraterestrul: Macintosh IIfx

Nava amiral a Armadei Apple este Macintosh IIfx (preţ 22900 DM, inclusiv adaptorul grafic şi un monitor color de 13"). Prescurtarea "fx" vine de la "fast" şi "extensible" (rapid şi extensibil), deoarece IIfx este Mac-ul cel mai rapid (procesor 68030 cu tactul de 40 MHz, plus coprocesor) şi cu cele mai multe sloturi de extensie (şase).

Cu un adaptor grafic de 24 de biţi poate reprezenta simultan peste 16 milioane de culori. Macintosh IIfx este indicat pentru toate aplicaţiile pentru care era recomandat şi predecesorul său; în plus el este singurul Mac (şi probabil singurul calculator personal, incluzînd chiar şi calculatoarele compatibile IBM PC 486) care este suficient de rapid pentru următoarele categorii de aplicaţii:

- desen tehnic pentru constructori, designeri şi arhitecţi (cuvînt cheie: CAD = Computer Aided Design, proiectare cu ajutorul calculatorului);
- animaţie
- aplicaţii multimedia
- simulări

(I. M. & R. M.)

Familia Macintosh dintr-o privire

Nume produs	Mac Classic	Mac SE/30	Mac LC	Mac IIsi	Mac IIci	Mac IIfx	Mac Portable
Preţ (DM)	3000	7650	4900	10500	15150	22900	9000
Procesor Motorola							
Procesor, Tact (MHz)	68000, 8	68030, 16	68020, 16	68030, 20	68030, 25	68030, 40	68HC00, 16
Coprocesor	-	68882			68882	68882	
Memorie							
Standard/Maxim (MBytes)	2/4	2/32	2/10	2/17	5/32	4/32	2/9
Lector de dischete							
Format	3,5"	3,5"	3,5"	3,5"	3,5"	3,5"	3,5"
Capacitate (MBytes)	1,44	1,44	1,44	1,44	1,44	1,44	1,44
Harddisk							
Capacitate (MBytes)	40	40	40	40	40	80	40
Sloturi							
Tip "Processor Direct Slot"	-	1	1	(1)	-	-	1
Tip "Nubus"	-	-	-	(1)	3	6 (5 libere)	-
Interfeţe							
Imprimantă	*	*	*	*	*	*	*
SCSI	*	*	*	*	*	*	*
Modem	*	*	*	*	*	*	*
Lector de dischete extern	*	*	-	*	*	-	*
Intrare/ieşire audio	nu/mono	nu/stereo	da/mono	da/stereo	nu/stereo	nu/stereo	mono
Monitor							
Diagonală	9"	9"	12"	13"	13"	13"	9,8"
Tip	a/n, integrat	a/n integrat	color	color	color	color	a/n-LCD, integrat
Rezoluţie	512x342	512x342	512x384	640x480	640x480	640x480	640x400
Nr. nuanţelor de gri, resp. al culorilor	2	2	256	256	256	>16,7 *10^6	2
Adaptor video integrat	*	*	*	*	*		*

Joystick

Cu calculatorul te poţi şi juca, dar pentru aceasta pentru anumite jocuri ai nevoie de un joystick. În cele ce urmează o să încercăm să vă explicăm ce este şi cum funcţionează un joystick.

Jocurile pe calculator sînt un mod plăcut de petrecere a timpului liber. Pentru a putea conduce vehicule spaţiale pe ecran, a simula pilotarea unui avion sau o cursă de maşini, nu este necesar doar un anumit program ci şi o configuraţie corespunzătoare. Un accesoriu important este joystick-ul, care transformă mişcările mîinii în informaţii pentru calculator. Prin intermediul lui jucătorul ia parte la acţiunea de pe ecranul calculatorului.

Cea mai importantă componentă a joystick-ului este **maneta**. Ea este mişcată cu mîna pentru a da calculatorului comenzile de direcţionare. Dacă se doreşte mişcarea figurii proprii de pe ecran în sus, se împinge maneta în sus. Doar prin simple împingeri se pot obţine în total opt sensuri de deplasare: sus, jos, stînga, dreapta, ca şi direcţiile diagonale (diagonal dreapta sus etc.).

Atunci cînd se mişcă maneta într-o anumită direcţie, mişcarea este înregistrată de cîteva componente electronice care se află în interiorul joystick-ului. În cel mai simplu caz cu maneta se apasă tasta unui aşa-numit **microîntrerupător**. Ca şi în cazul unui întrerupător pentru lumină, se închide un circuit şi un impuls de curent slab este transmis prin intermediul **cablului de legătură** prin **mufa joystick-ului** la calculator. Prin legătură programul de joacă poate afla care sînt valorile transmise de joystick. Dacă valoarea este "zero", înseamnă că tastele de direcţie nu au fost apăsate, dacă valoarea este "unu", înseamnă că o anumită tastă a fost apăsată. Majoritatea joystick-urilor dispun pentru sesizarea direcţiei de deplasare de patru microîntrerupătoare de la care pleacă cîte un fir la calculator. Cu ajutorul lor calculatorul poate afla şi dacă maneta a fost împinsă, de ex., diagonal stînga jos. În acest caz vor fi pe "unu" simultan valorile pentru "jos" şi "stînga".

Joystick-urile cu microîntrerupătoare sînt foarte convenabile (performanţe/preţ) şi din acest motiv sînt utilizate de o mulţime de jocuri pentru HC-uri (Home Computer). Ele pot indica doar direcţiile de deplasare, dar nu şi cît de departe a fost împinsă maneta într-o anumită direcţie. Pentru aceasta sînt necesare aşa-numitele joystick-uri analogice, care ştiu deosebi şi situaţii intermediare cum ar fi "50% dreapta" sau "95% dreapta". În timp ce celelalte joystick-uri asemenea unui întrerupător pentru lumină nu cunosc decît poziţiile "închis" şi "deschis", modelele analogice acţionează ca un reostat care permite ca lumina să scadă sau să crească în trepte. Avantajul: acolo unde contează sensibilitatea degetelor, de ex. la simularea zborului cu un avion cu reacţie, se poate conduce mai exact. Joystick-urile cu microîntrerupătoare sînt denumite, pentru deosebire, "joystick-uri digitale".

Joystick-urile digitale utilizează în locul microîntrerupătoarelor două "potenţiometre". Cu cît este deplasat cursorul unui potenţiometru, prin intermediul manetei, mai mult într-o anumită direcţie, cu atît mai mare sau mică este doza de curent care este transmisă prin intermediul cablului de legătură la aşa-numitul "convertor analog-digital", care transformă impulsurile de curent în valori digitale. În acest caz nu se face însă distincţie doar între "zero" şi "unu" ci pot fi deosebite pînă la 64 de valori intermediare. Valoarea "zero" semnifică, de ex., "la stînga de tot", iar valoarea "63" înseamnă "la dreapta de tot". Acelaşi lucru este valabil şi pentru cel de-al doilea potenţiometru, care semnalează poziţiile de sus pînă jos. Atunci cînd maneta este la mijloc ambele potenţiometre trebuie să indice valoarea "31".

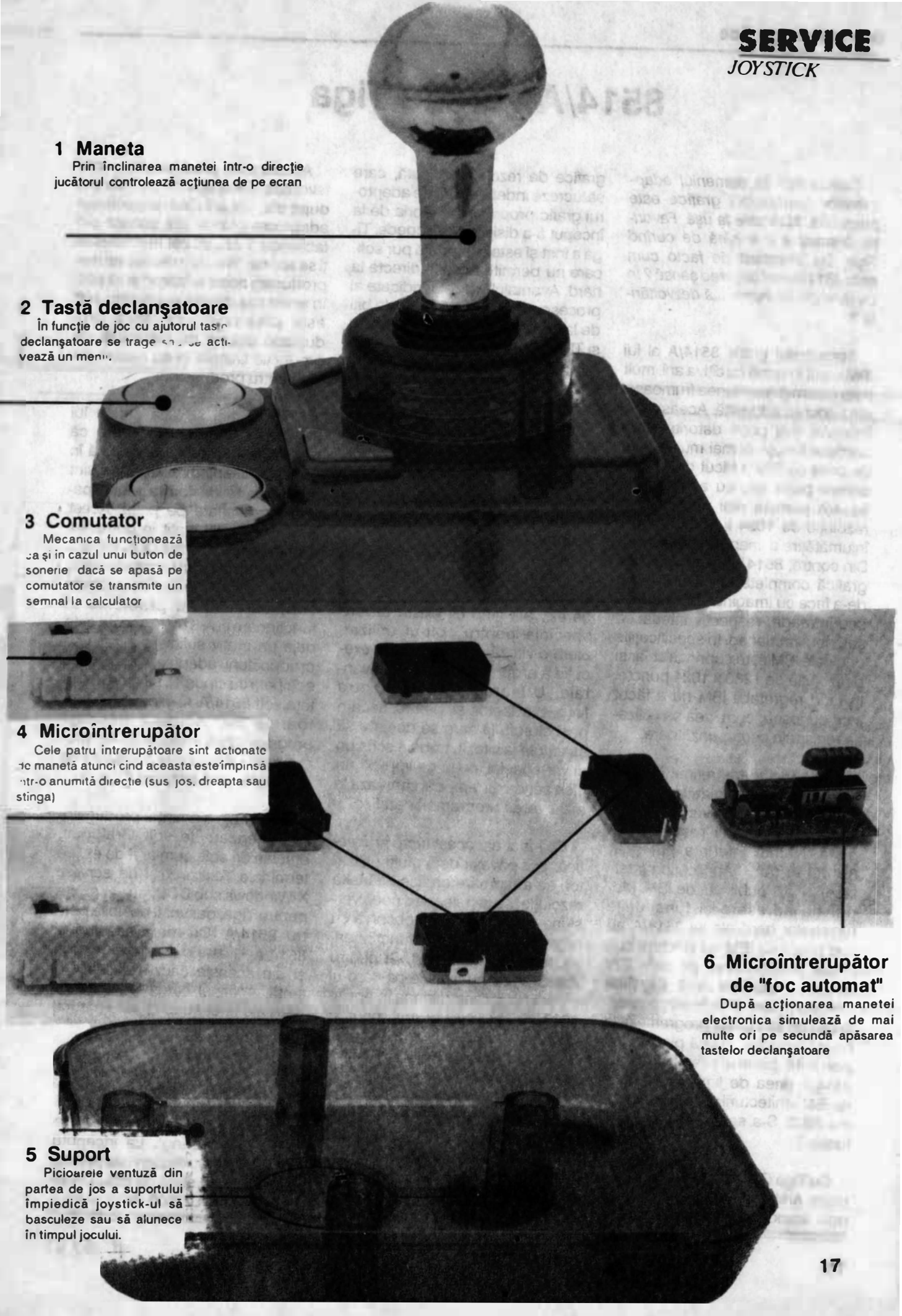
Joystick-urile analogice sînt utilizate mai ales pentru PC-uri. Acestea au nevoie în plus de un aşa-numit "port pentru jocuri" (Preţ: 50 DM plus 50 DM pentru joystick; joystick-urile digitale costă între 10 şi 40 DM). Acesta este de fapt o placă de extensie care conţine o interfaţă pentru joystick. Majoritatea jocurilor se pot lipsi însă de un joystick, operarea putîndu-se face de la tastatură sau cu un mouse.

Înafara manetei de comandă o altă componentă importantă a joystick-ului o reprezintă **tastele declanşatoare** (trăgaciul). Prin intermediul lor jucătorul poate transmite alte comenzi calculatorului, asemănător ca prin intermediul tastelor de pe mouse. Prin intermediul tastelor jucătorul poate trage asupra "inamicilor" sau poate selecta o opţiune dintr-un meniu.

Joystick-urile bune au şi o **tastă de "foc automat"**, care simulează apăsarea repetată a tastelor. Jucătorul se poate concentra astfel doar asupra conducerii.

La un joystick sînt foarte importante şi **picioarele de sprijin**, care să dea o stabilitate joystick-ului în timpul jocului. Din acest motiv multe modele dispun de picioare ventuză.

(R.M.)



1 Maneta

Prin înclinarea manetei într-o direcţie jucătorul controlează acţiunea de pe ecran

2 Tastă declanşatoare

În funcţie de joc cu ajutorul tastei declanşatoare se trage sau se activează un meniu.

3 Comutator

Mecanica funcţionează ca şi în cazul unui buton de sonerie dacă se apasă pe comutator se transmite un semnal la calculator

4 Microîntrerupător

Cele patru întrerupătoare sînt actionate de manetă atunci cînd aceasta este împinsă într-o anumită direcţie (sus jos, dreapta sau stînga)

6 Microîntrerupător de "foc automat"

După acţionarea manetei electronica simulează de mai multe ori pe secundă apăsarea tastelor declanşatoare

5 Suport

Picioarele ventuză din partea de jos a suportului împiedică joystick-ul să basculeze sau să alunece în timpul jocului.

8514/A versus Tiga

Concurenţa în domeniul adaptoarelor (cartelelor) grafice este puternică. XGA bate la uşă. Favoritul branşei a fost pînă de curînd Tiga. Un standard de facto cum este 8514/A mai are vreo şansă ? În ce direcţie se îndreaptă dezvoltările ?

Standardul grafic 8514/A al lui IBM creat în urmă cu cîţiva ani, mult timp a dormit asemenea frumoasei din pădurea adormită. Aceasta s-a întîmplat mai puţin datorită standardului însuşi, ci mai mult a ţinut de ceea ce IBM a făcut din el. Deoarece puţin ştiu cu adevărat că 8514/A permite mai mult decît o rezoluţie de 1024 x 768 în modul înjumătăţire a imaginii (interlaced). Din contră, 8514/A este o interfaţă grafică completă şi nu are nimic de-a face cu imaginea înjumătăţită sau întreagă, respectiv interlaced sau non-interlaced. În specificaţiile originale IBM este cuprinsă chiar şi o rezoluţie de 1280 x 1024 puncte. În mod regretabil IBM nu a făcut încă cunoscut cum vrea să utilizeze registrele corespunzătoare.

IBM a fost foarte reţinut în ceea ce priveşte informaţiile despre 8514/A, ceea ce a avut ca efect o acceptanţă relativ restrînsă pe piaţă. Numai interfaţa pur soft, 8514/A-AI (AI = Application Interface), a fost publicată de IBM. Modul de adresare şi funcţiunile registrelor hard ale lui 8514/A au fost ţinute de IBM cu stricteţe sub cheie. Direcţionarea pe care IBM dorea s-o dea era clară: Big Blue dorea să determine, în sfîrşit, casele de soft să nu mai programeze la nivel hardware. O altă piedică importantă pentru răspîndirea lui 8514/A ţinea de limitările impuse de IBM arhitecturii microcanal a seriei PS/2. S-a schimbat cumva situaţia ?

Cu Tiga (Texas Instruments Graphics Architecture) s-a realizat un nou standard pentru adaptoare grafice de rezoluţie ridicată, care să lucreze independent de adaptorul grafic propriu zis. Euforia de la început s-a disipat însă repede. Tiga a fost şi este o interfaţă pur soft, care nu permite accese directe la hard. Avantajul vitezelor ridicate al procesoarelor grafice pe 32 de biţi de la Texas Instruments (TI-34010 şi TI-34020) se pierde din această cauză.

Acest lucru a fost observat de către cîţiva dintre producătorii de adaptoare grafice cu procesoare TI, şi aceştia nu mai oferă chiar nici o interfaţă Tiga. Să scrii drivere unitare pentru astfel de adaptoare este imposibil deoarece fiecare adaptor grafic are un alt design hard. Astfel există adaptoare care sprijină în mod special funcţiile pan şi zoom sau alte caracteristici, care de ex., atît timp cît există drivere speciale pentru soft-ul utilizat, oferă o viteză mai ridicată de execuţie a programelor sau alte avantaje. Urmarea este că pentru fiecare adaptor şi pentru fiecare pachet soft, la care se doreşte să se umble la viteză, trebuie scris un driver special, ceea ce implică un efort serios. Şi tempoul contează la majoritatea programelor grafice.

Ce efect are acest lucru în practică, pot povesti deja mulţi utilizatori de adaptoare grafice de înaltă rezoluţie: se instalează o nouă versiune a unui soft şi se observă cu regret că programul nu funcţionează pe PC-ul disponibil. Şi pentru aceasta nici nu trebuie ca hard-ul să fie vechi. Cine deţinea în aprilie 1990, de ex., ce era mai "nou" în domeniul adaptoarelor grafice, s-a trezit nu mai departe decît în iunie, la instalarea lui Windows 3.0, că se loveşte de aceleaşi probleme de care se lovise şi cu un an înainte la actualizarea versiunii Windows 2.03 cu mai noua versiune 2.1. Trebuiau aşteptate drivere noi, şi aceasta de regulă durează mai multe luni.

Acesta este, privit relativ, un caz favorabil, deoarece ce se întîmplă după doi, trei ani cînd respectivele adaptoare grafice sînt scoase din fabricaţie ? Atunci cel mai adesea ţi se spune: "Ne pare rău dar nu mai producem acest adaptor" şi să scrii în acest caz drivere noi pentru soft este prea costisitor. Mulţi producători dispun de capacităţi de producţie limitate şi au nevoie de ele pentru produse noi.

Tocmai aici se află şansa lui 8514/A. Deoarece cu toate că adaptoarele 8514/A nu au pînă în prezent o răspîndire largă, ele sînt recunoscute de foarte multe pachete soft aflate pe piaţă. Acest lucru este valabil atît în domeniul sistemelor de operare DOS şi OS/2 cît şi pentru Unix. În totalitate există astăzi mai mult de 300 de pachete soft, care nu au probleme în funcţionare cu 8514/A. Acesta este deja un motiv suficient pentru ca producătorii adaptoarelor grafice echipate cu cipuri TI, să ofere interfaţa soft 8514/A-AI şi pentru adaptoarele lor. Această interfaţă program oferă modalităţi de programare sensibil mai apropiate de hardware decît interfaţa Tiga.

După cum a anunţat producătorul american de soft Grafpoint, emulări de soft, cum ar fi de ex. de terminale Tektronix şi de servere X-Windows sub DOS, nu vor exista pentru Tiga, dar vor fi posibile pentru 8514/A. Cu toate acestea şi 8514/A-AI este o interfaţă pur soft şi din această cauză este relativ lentă. Accesul direct la registrele hard ale unui adaptor este natural mai rapid.

În 1989 producătorului american Western Digital i-a reuşit să citească conţinutul registrelor cipului IBM-8514/A prin tehnica "Reverse-Engineering". La începutul acestui an erau disponibile primele adaptoare cu setul de cipuri Western Digital WD-9500. În locul origi-

nalelor au început să apară clone interesante ca preţ. Şi aceste clone sînt disponibile nu numai pentru microcanal ci şi pentru magistralele normale ale AT-urilor. Copiile au o rezoluţie de 1024 x 768 puncte la 16 şi respectiv 256 de culori şi lucrează în plus în modul imagine întreagă (non-interlaced).

Această situaţie - după ce registrele hard ale lui 8514/A au devenit cunoscute - face acest adaptor încă şi mai interesant pentru ca producătorii mari de soft să-l trateze în pachetele lor. Microsoft a făcut acest lucru cu Windows 3.0, acelaşi lucru făcîndu-l şi Autodesk cu Autocad 11.0. Tocmai succesul uriaş al lui Windows 3.0 face ca 8514/A să apară ca un standard ideal pentru adaptoare grafice. Se poate lucra cu el fără probleme nu numai cu bogata paletă de aplicaţii Windows, ci şi cu programe CAD cum ar fi Autocad, PC-Draft sau cu soft-uri asemănătoare.

Nici o grijă pentru soft

Şi în domeniul Unix, 8514/A se scrie cu litere mari. X-Windows funcţionează pe 8514/A, acelaşi lucru întîmplîndu-se şi cu programele producătorilor Interactive şi SCO (Santa Cruz Operation). Servere DOS pentru X-Windows cu acest standard vor fi disponibile cel mai tîrziu în august 1991.

Aceste produse se află momentan în faza de beta-testare, deci sînt aproape de lansarea pe piaţă. Criteriul important şi hotărîtor al întregii situaţii este că utilizatorul nu trebuie să-şi facă probleme pentru modul în care va fi sprijinit soft-ul. Pentru aceasta se angajează în competiţie tot mai mulţi producători de adaptoare grafice.

Efortul de elaborare al unui driver per adaptor grafic şi pachet soft se situează între 20000 şi 100000 DM. O investiţie care într-un fel sau altul trebuie plătită de utilizatori şi nu prezintă nici o garanţie de viitor.

Testele comparative între Tiga şi clonele 8514/A au produs o surpriză în favoarea lui 8514/A. Adaptorul 8514/A echipat cu un set de cipuri WD-9500 la un tact de 60 MHz a fost în medie de trei ori mai rapid decît un adaptor echipat cu procesorul TI-34010, la acelaşi tact.

Clonele sosesc

Alţi producători se află deja în preajma lansării pe piaţă a unor seturi de cipuri 8514/A. Astfel un producător american de adaptoare grafice a anunţat pentru această primăvară un clon 8514/A, cu un tact de 80 MHz, bazat pe cipuri de la firma Chips & Technologies (C&T). Cu acestea ar trebui să se atingă o viteză care să echivaleze puterea adaptoarelor grafice echipate cu procesorul grafic TI-34020, dar la preţuri sensibil mai scăzute.

Preţul final de utilizare al acestui adaptor se va situa sub 1800 DM. Acesta ar fi cu cca. 60% mai ieftin decît un adaptor cu TI-34020. Pentru acesta ar fi posibilă o rezoluţie de 1024 x 768 la 256 de culori şi în cazul unei echipări corespunzătoare chiar de 1280 x 1024 la 256 de culori.

Premisa de funcţionare a unui adaptor grafic 8514/A este utilizarea unui monitor multi-frecvenţă, deoarece unele soft-uri accesează direct registrele hard. Aceasta poate însemna în cazuri concrete o rezoluţie ecran de 640 x 480 puncte, 800 x 600 puncte, sau pînă la 1024 x 768 interlaced, respectiv 1024 x 768 non-interlaced. Aceasta este dependentă doar de soft şi nu poate fi influenţată din afară.

Totuşi nu este cunoscut nici un soft care să utilizeze rezoluţia de 800 x 600 după specificaţiile 8514/A, iar multe programe lucrează în modul 1024 x 768 non-interlaced.

În dezvoltarea tehnică a adaptoarelor grafice sînt de aşteptat cîteva noutăţi interesante. Astfel vor exista adaptoare cu seturi de cipuri VGA "on-board". Prin două ieşiri separate pentru semnale VGA şi 8514/A şi o "Pass-Through" (trecere) internă a semnalelor VGA peste ieşirea de înaltă rezoluţie, va exista posibilitatea de a utiliza unul sau două monitoare cu un singur adaptor grafic. Avantajul imediat: pentru întregul sistem grafic va fi utilizat un singur conector (slot) la magistrală.

Şi la capitolul ergonomie adaptoarele 8514/A au un avans: monitoarele cu o frecvenţă de împrospătare a imaginii de 70 Hz aparţin deja standardului. Elaborarea unor adaptoare cu o frecvenţă pe verticală peste 80 de Hz este perfectată. Problema va fi doar să se găsească acele monitoare care să poată lucra la o astfel de frecvenţă.

Pînă acum i-a reuşit unui singur producător "să-i sufle" IBM-ului un standard grafic de pe piaţă, este vorba despre Hercules. Aceasta s-a întîmplat însă numai pentru un timp deoarece astăzi Hercules se retrage: noile adaptoare grafice Hercules sînt compatibile cu toate standardele grafice IBM, inclusiv 8514/A.

Nu numai în domeniul impunerii unor standarde grafice a fost IBM cel mai mare. Egal, cît de bune sau cît de proaste au fost aceste standarde, toţi ceilalţi producători au urmat, mai devreme sau mai tîrziu, direcţia de marş a uriaşului albastru. De îndată ce avantajele celui mai bun standard, lansat pînă acum de IBM, vor deveni cunoscute, nu mai există nici o îndoială: cu sau fără sprijinul lui IBM 8514/A se va impune alături de XGA.

(R. M.)

Merită să mizezi pe XGA ?

Cu anunţarea de către IBM a lui XGA s-a deschis un nou capitol în problema standardelor grafice, o sentinţă rămîne însă: din punct de vedere tehnic sînt puţine noutăţi de semnalat. Clienţii şi distribuitorii sînt încurcaţi. Ce este XGA, ce avantaje aduc specificaţiile ? Se vor impune ?

IBM a anunţat un nou standard - şi întreaga branşă este tulburată. Extended Graphics Array (XGA) este conceptul care înfierbîntă multe sentimente în şi aşa destul de agitata lume a afacerilor cu grafică. "Interfaţa grafică XGA este o implementare busmaster, care oferă o nouă dimensiune în grafică", afirmă bineînţeles Big Blue. Totuşi părerile celor din branşă sînt discordante.

În urma unui chestionar fulger dat de revista PC Magazin mai multor distribuitori şi producători de adaptoare grafice a rezultat că: unii livrează în continuare din stocurile de adaptoare 8514/A şi consideră că acestea au încă şanse bune să se menţină, în timp ce alţii sînt entuziasmaţi de XGA. Un comentariu exemplificator va lămuri îndoiala unor cunoscători ai branşei: "marfă veche în amabalaj nou, atrăgător - foarte iscusit din parte lui IBM".

Cum poate fi înţeles acest lucru ? Vechi este XGA deoarece rezoluţia de 1024 x 768 dpi (dot per inch) exista deja de mult, şi chiar din casa IBM-ului (8514/A). Nou este ambalajul: XGA este compatibil soft cu VGA, 8514/A, Windows, Presentation Manager şi Autocad. Aplicaţiile corespunzătoare trebuie deci să funcţioneze fără probleme.

Oferta de standarde în segmentul de piaţă al rezoluţiilor oferite de VGA Plus, Super-VGA şi 8514/A era încîlcită. Texas Instruments proclamase interfaţa soft Tiga, pentru adaptoare grafice de rezoluţie ridicată echipate cu procesoare TI din familia TMS 340, ca "inteligentă". Vesa specificase un standard propriu, iar producători ca Miro sau Spea pătrunseseră pe piaţă cu rezoluţii de 2048 x 2048 pixeli şi cu drivere proprii. IBM s-a ţinut cu tărie de 8514/A şi a aşteptat să vadă dacă va exista necesitatea unor rezoluţii mai mari sau a mai multor culori.

O lipsă cunoscută era că introducerea unei plăci grafice de rezoluţie ridicată, cel mai adesea scumpă, era dependentă de existenţa driverelor corespunzătoare pentru soft. Dacă XGA se va impune, atunci se va termina cu aşteptarea după drivere speciale, parţial specifice chiar adaptorului utilizat. Istoria PC-urilor a arătat de mai multe ori că standardele stabilite de IBM s-au impus cu repeziciune. În cazul XGA acest fenomen s-ar putea repeta.

Ce avantaje oferă XGA ? Big Blue promite o creştere sensibilă a puterii sistemelor cu microcanal, deoarece adaptorul ca busmaster preia anumite funcţiuni ale procesorului principal degrevîndul pe acesta şi lăsîndu-l să se ocupe între timp de alte probleme. IBM vorbeşte despre adaptorul XGA ca despre un subsistem grafic. Pînă acum este cunoscut că adaptorul lucrează cu un procesor RISC (Reduced Instruction Set Computer), al cărui set redus de instrucţiuni permite o creştere a vitezei. Suplimentar, utilizînd un adaptor XGA în modul standard VGA sînt posibile mai multe culori la o aceeaşi rezoluţie.

XGA ca busmaster

Extended Graphics Array a fost folosit mai întîi în sistemele microcanal PS/2 90 şi 95. În modelul 90 XGA este integrat pe placa de bază ca interfaţă monitor. În 95 XP 486, XGA este prevăzut standard ca adaptor busmaster. Acelaşi adaptor grafic XGA poate fi instalat, la dorinţă, în toate modelele PS/2 cu procesoare 80386SX, 80386 şi 80486. Într-un sistem PS/2 vor fi sprijinite mai multe astfel de adaptoare. Alte caracteristici: setul tehnologic VRAM (Video Random Access Memory), un cursor hard propriu şi drivere pentru aplicaţii CAD (Computer Aided Design) speciale.

Ce rezoluţii sînt posibile ? Posesorii de modele PS/2 sau de PC-uri cu microcanal pot fi liniştiţi, nu au nevoie de un nou monitor. XGA se bazează în mare pe standardele 8514/A, şi respectiv pe Super VGA şi partajează cu acestea rezoluţia de 1024 x 768 pixeli la 256 de culori. În modul VGA, subsistemul de regiştri XGA este compatibil cu standardul VGA. În mod text XGA permite afişarea pe ecran a 132 de caractere, a cîte 8 pixeli. În cazul rezoluţiei de 640 x 480 pixeli pot fi reprezentate 65536 de culori.

Branşa a privit pozitiv intrarea lui IBM în VESA (Video Electronics Standard Association). Acest pas indică cu claritate renunţarea lui IBM la rolul "călăreţului singuratec" şi lasă speranţe pentru consolidarea pieţei de grafică. Cunoscătorii afirmă că deşi specificaţiile XGA au fost date publicităţii vor trece 9 pînă la 12 luni pînă cînd vor putea fi dezvoltate adaptoare noi. Greutăţile vor fi produse mai ales de interfaţa de 32 de biţi, ca şi de cipurile care sînt mai complexe la XGA decît la VGA. După IBM este posibilă o creştere a rezoluţiei de pînă la 4096 x 4096 pixeli.

Dataquest a criticat totuşi faptul că acest lucru nu este definit în mod clar, deoarece spaţiul stabilit de IBM ar putea duce la incompatibilităţi. Un lucru este clar însă: IBM doreşte să sprijine producătorii de adaptoare. Din această cauză XGA are şanse bune de a se impune. Doi "stropi de vermut" există totuşi: concurenţa din propria casă, 8514/A, care tocmai a început să se impună, şi adaptorul Tiga care pînă acum a fost favorizat.

(R. M.)

Standarde grafice

Anul introducerii	Standard grafic	Rezoluţie	Mod	Culori	Matricea de pixeli / caracter	Moduri compatibile hard	Frecvenţa verticală (Hz)	Frecvenţa orizontală (KHz)
1981	**MDA** (Color Graphics Adapter)	720x350	Text	1	9x14	-	50	18.43
1981	**CGA** (Color Graphics Adapter)	640x200	Text	16	8x8	-	60	15.75
		320x200	Text	16	8x8	-	60	15.75
		160x200	Grafic	16	-	-	60	15.75
		320x200	Grafic	4	-	-	60	15.75
		640x200	Grafic	2	-	-	60	15.75
1982	**MGA** (Hercules Monochrome Graphics Adapter)	720x350	Text	1	9x14	MDA	50	18.10
		720x348	Grafic	1	-	MDA	50	18.10
1984	**EGA** (Enhanced Graphics Adapter)	640x350	Text	16	8x14	CGA,MDA	60	21.85
		720x350	Text	4	9x14	CGA,MDA	60	21.85
		640x350	Grafic	16	-	CGA,MDA	60	21.85
		320x200	Grafic	16	-	CGA,MDA	60	21.85
		640x200	Grafic	16	-	CGA,MDA	60	21.85
		640x350	Grafic	16	-	CGA,MDA	60	21.85
1984	**PGA** (Professional Graphics Adapter)	640x480	Grafic	256	-	CGA	60	30.50
1987	**VGA** (Video Graphics Adapter)	720x400	Text	16	9x16	CGA,EGA	70	31.50
		360x400	Text	16	9x16	CGA,EGA	70	31.50
		640x480	Grafic	16	-	CGA,EGA	60	31.50
		640x480	Grafic	2	-	CGA,EGA	60	31.50
		320x200	Grafic	256	-	CGA,EGA	70	31.50
1987	**MCGA** (Memory Controller Gate Array)	320x400	Text	4	8x16	CGA,EGA	70	31.50
		640x400	Text	2	8x16	CGA,EGA	70	31.50
		640x480	Grafic	2	-	CGA,EGA	60	31.50
		320x200	Grafic	256	-	CGA,EGA	70	31.50
1989	**Super VGA** (recomandări VESA)	800x600	Grafic	16	-	VGA,CGA, EGA	56, 60 sau 72	35.00, 37.60 sau 48.00
1987	**8514/A**	1024x768	Grafic	16	-	VGA şi compatibilele în jos	43.48	35.52
		640x480	Grafic	256	-		43.48	35.52
		1024x768	Grafic	256	-		43.48	35.52
1990	**XGA** (Extended Graphics Array)	640x480	Grafic	256	-	VGA	43.48	35.52
		1024x768	Grafic	256	-	VGA	43.48	35.52
		640x480	Grafic	65536	-	VGA	43.48	35.52
		1056x400	Text	16	8x16	VGA	43.48	35.52

(Sursa: PC Magazin 4/91)

Evaluarea performanţelor harddisk-urilor

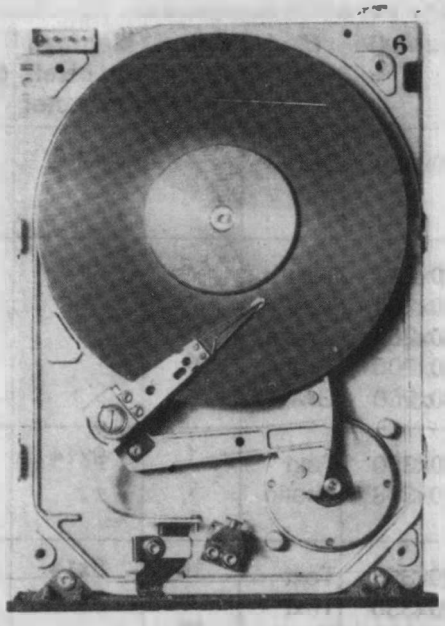
Cumpărătorul unui PC, chiar dacă de cele mai multe ori preferă soluţia "de-a gata", din ce în ce mai des are posibilitatea opţiunii la configurarea sistemului cu care va lucra. Cîteva criterii de decizie pentru alegerea harddisk-ului sînt explicitate în acest articol.

Trei sînt criteriile cele mai importante: preţul, capacitatea şi "performanţa" unităţii de disc, "performanţa" fiind o sinteză a mai mulţi parametri, ca de exemplu timpul de răspuns (acces), viteza de transfer, tratarea erorilor, fiabilitatea ş.a.m.d.

Sistemele de operare precum VMS, OS, Unix ca şi toate programele moderne de aplicaţii (proiectare 3D, sisteme complexe de baze de date) lucrează foarte intens cu memoria externă - unităţile de disc. Deseori, la sistemele multiutilizator, motivul principal pentru încetineala exasperantă a calculatorului nu-l constituie puterea insuficientă a unităţii centrale, ci performanţa redusă a discului. Fenomenul survine în special la aplicaţii intensive în privinţa datelor, atît comerciale cît şi ştiinţifice. (Mie mi s-a întîmplat chiar pe un VAX, cînd proiectam cu editorul grafic "Princess").

Costul

Depinde desigur de firma producătoare şi tipul unităţii de harddisk (vezi tabelul). Trebuie avută în vedere minimizarea costului total, nu pur şi simplu cumpărarea celui mai ieftin disc disponibil la capacitatea dată. Trebuiesc avute în vedere mai multe considerente: interfaţa, controller-ul şi funcţiile suplimentare ale acestuia, compatibilitatea cu restul echipamentului (nu doar ca interfaţă sau magistrală!), funcţiile integrate (precum Cache Look-Ahead sau tratarea "transparentă", în timp real, a defectelor), posibilităţile de extindere ulterioară şi fiabilitatea. Mai ales în condiţiile noastre, unde depanarea ridică probleme, ar trebui să ezităm a cumpăra un harddisk garantat doar 6 luni sau un an şi cu MTBF (Mean Time Between Failures - timpul mediu de defectare) nedepăşind 25.000 ore, chiar dacă-i substanţial mai ieftin.



Totodată, nu trebuie exagerat - pentru aplicaţiile uzuale, Hewlett Packard oferă discuri performante, cu MTBF de 150 şi chiar 175.000 ore, iar Seagate dă la modelele de vîrf 150.000 ore în mediul unui birou obişnuit şi 250 000 în sală calculator climatizată (clasa A). Ceea ce înseamnă 30 de ani la exploatarea non-stop în 3 schimburi şi peste 100 de ani la utilizarea "casnică". Chiar în România, durata de viaţă morală a unui sistem nu depăşeşte 25 de ani!

Capacitatea

Ce semnifică cele două cifre din tabel ? Capacitatea brută este capacitatea totală, fizică, produsul dintre nr. total de piste x nr. de octeţi ce pot fi scrişi pe o pistă. Capacitatea netă este cea utilă şi este semnificativ mai mică - nr. octeţi/sector x nr. sectoare/pistă x nr. piste/faţă disc x nr. feţe disc. Diferenţa se datorează în principal spaţiului dintre sectoare, informaţiilor adiţionale înscrise în antetul şi "coada" fiecărui sector, spaţiului (eventual) rezervat pentru compensarea erorilor. Factorul de corecţie poate fi de 10-20%. Unele firme dau capacitatea brută, fiindcă livrează discul neformatat, deci utilizatorul poate să şi-l organizeze cum doreşte, altele specifică ambele valori sau numai pe ultima - capacitatea netă - dacă livrează discul complet formatat.

Ce capacitate este necesară?

Cel mai simplu este de estimat în cazul aplicaţiilor de tip prelucrare text: o pagină A4 este echivalentă cu circa 3 ko, deci pe un disc de 40 Mo se pot înmagazina circa 12.000 de pagini de text dactilografiat. De fapt, valoarea reală poate fi considerabil mai mică: informaţia în realitate nu-i compactă, ci cu mai multe "auxiliare", versiuni anterioare, fişiere provizorii etc. Sistemul de lucru ocupă de asemenea spaţiu pe disc. Programele de procesare de texte ocupă adesea 2 Mo sau mai mult, iar aplicaţii specifice de editare - tehnoredactare, precum Ventura Publisher sau Pagemaker, au nevoie de 3 - 10 Mo sau chiar mai mult, cînd se lucrează cu imprimante laser sau postscript.

Ţinînd cont şi de versiunile multiple, chiar şi un disc de capacitate mare se umple repede. În afară de aceasta, totdeauna trebuie rezervat loc pentru date neprevăzute: altfel, se poate ajunge la situaţii de blocaj.

Trebuie estimat optimul compromisului capacitate - preţ: discul să nu fie supradimensionat, dar nici subdimensionat.

Uneori, cumpărarea unui harddisk fără rezerve prea mari de spaţiu şi înlocuirea lui ulterioară cu unul de capacitate mai mare poate fi o soluţie mai ieftină decît cumpărarea directă a modelului mai scump, dacă timpul dintre achiziţia iniţială şi schimb nu-i prea mare. Deci, se va avea în vedere un harddisk suficient pentru următorii

1 - 3 ani.

Observaţie: relaţia capacitate - preţ nu e liniară; modelele mai mari au costul specific mai redus. În plus, şi timpul de acces este mai mic decît la modelele mai mici, ieftine.

Performanţa

Un criteriu deosebit de important pentru performanţă este timpul în care (în medie statistică) subsistemul de disc duce la îndeplinire o comandă de scriere sau citire a datelor. Acest timp se însumează cu timpii necesari pentru diversele operaţii pregătitoare pentru execuţia comenzii. Pentru evaluarea performanţei harddiskului nu este relevant timpul de overhead ("regie") al sistemului de operare şi cel al controller-ului de disc, deoarece aceştia sînt relativ independenţi de tipul unităţii de disc folosite. Pentru simplificare, putem considera - în primă aproximaţie - că unitatea de disc este legată direct la calculator, care-i sursa de date. Cînd unitatea de disc primeşte o comandă de citire sau scriere, trebuie ca mai întîi blocul capetelor să se poziţioneze pe cilindrul unde există primul bloc de transferat. Acest timp se numeşte timp de poziţionare. Mărimea sa depinde evident de locul unde se face poziţionarea, astfel că se ia, în mod normal, timpul în care capetele parcurg 1/3 din lungimea maximă (30% din cursă). Se poate demonstra matematic că, la blocuri de date necorelate, avem statistic L poziţ. medie = 1/3 L poziţ. max., relaţia fiind valabilă şi pentru timpi. Valori extreme: <= 10 ms la discurile cele mai rapide, 20 - 30 ms în medie, 100 ms sau chiar mai mult la dischete sau discuri optice. Pentru siguranţă, firmele serioase dau, în afara timpului mediu de poziţionare, şi pe cel maxim (pentru 100% din cursa capetelor). Unele discuri mai necesită un timp de poziţionare mecanică, pentru stabilizarea la locaţia respectivă. Acest timp de stabilizare (settling) se specifică de obicei doar la unităţile "mici" de discuri, de 5 1/4" diametru.

După poziţionare, trebuie aşteptat pînă ce suprafaţa discului, care se roteşte, aduce blocul de date dorit în dreptul capetelor de citire/scriere. Acesta este timpul de latenţă. Majoritatea unităţilor de disc rotesc discurile cu 3600 rpm (rotaţii pe minut), astfel că timpul necesar pentru o jumătate de rotaţie este de cca 8ms. Fiind egal la toate discurile, nu-i un criteriu pentru departajare. Dar atenţie! Unităţile de disc contemporane care vor să asigure performanţe superioare utilizează turaţii mai mari (lucru imposibil la discurile vechi, de diametru mare, >= 8"). Lucrînd la peste 4000 sau chiar 5600 r.p.m., timpul de latenţă scade la cca 5 ms - şi atunci poate fi criteriu de selecţie.

Urmează transferul datelor de pe disc la controller sau invers. Dacă nu se transferă cantităţi uriaşe de date la o singură cerere de citire sau scriere, şi acest timp poate fi dat factor comun şi neglijat. Sistemele de operare precum VMS şi OS, ca şi majoritatea aplicaţiilor (sisteme de baze de date, procesare de text, etc.) transferă în general puţine blocuri de date la o cerere, tipic numai 4-8 blocuri/cerere de citire/scriere, astfel că timpul de transmisie pentru respectiva cerere este de 1-3 ms la majoritatea unităţilor de disc.

Deci, în mod normal, timpul total de acces (Zugriff în germană) este suma timpilor de: poziţionare, stabilizare, latenţă, şi transfer, deci este cu un ordin de mărime superior timpului de transfer, care şi el poate fi neglijat.

De altfel, o viteză foarte mare de transfer a datelor între disc şi controller nu-i utilă dacă acesta din urmă nu poate să debiteze datele pe magistrală la fel de repede.

De aceea, actualmente majoritatea controllerelor de harddisk-uri contemporane au buffere (memorii-tampon) SMB, de adaptare a vitezelor. Sarcina SMB este de a "potrivi" viteza de transfer ridicată dintre disk şi controller cu cea mai lentă, dintre controller şi memoria calculatorului, memorînd temporar datele pînă ce sînt transferate. Altfel pentru a evita riscul pierderii de date, transferul de pe disc la controller ar fi trebuit să se facă mai lent decît posibilităţile reale, intrinseci.

Deci, rata de transfer a datelor între disk şi controller este edificatoare mai curînd pentru performanţele şi puterea întregului sistem, decît ale unităţii de disc propriu-zise.

Mic dicţionar - harddisk-uri

Tratarea automată a defectelor

Funcţia de tratare automată a defectelor ("defect management") este integrată, de exemplu, la seria ProDrive a firmei Quantum. Este un sistem de tratare automată a defectelor integrat sub formă de firmware în unitatea de disc. Se elimină astfel necesitatea ca utilizatorul să ridice hărţi ale defectelor (clasica tabelă cu "bad blocks") şi să scrie soft utilitar dedicat evitării folosirii acestor zone.

La fabrică, suprafeţele de date sînt baleiate pentru detectarea sectoarelor defecte. Orice sector defect este dez-alocat înainte de a expedia unitatea de disc. Se realizează automat o hartă inline (ILDM) a defectelor, care conţine locaţiile tuturor sectoarelor cu defecte găsite pe disc. Pentru protecţie la erori, harta este duplicată pe mai multe piste. În plus, în timpul utilizării normale, unitatea de harddisk continuă să compenseze (detecteze + ocolească) orice noi sectoare cu defecte apărute pe disc. Toate defectele găsite la utilizator sînt cartate direct în sectoarele de înlocuire. Toată această tratare a defectelor este complet transparentă pentru utilizator, care n-are nici o grijă !

Valori tipice: rata de transfer "internă" poate ajunge la 10 şi chiar 28 Mo/s, în timp ce transferul "extern" (disc <-> memoria calculatorului) se face doar cu 1-5 Mo/s. Aceste valori diferă, chiar la acelaşi echipament, în funcţie de modul de lucru selectat:

- sincron 2-5 Mo/s, asincron cu o viteză redusă la jumătate;
- în regim de "salvă", "rafală" 4-5 Mo/s, în regim continuu, susţinut (cînd bufferul SMB nu mai poate acţiona) doar 1-2 Mo/s.

În concluzie, criteriile importante pentru stabilirea performanţelor unei unităţi de disc sînt:

-timpul de poziţionare ;
-timpul de latenţă;
-timpul de transfer al datelor în întregul sistem (dat de interfaţa sau magistrala folosită).

În plus, la unele aplicaţii speciale ce necesită transferul unor volume foarte mari de date la o singură cerere de citire/scriere, trebuie considerat şi timpul de transfer "interior", între disc şi controller.

Există o întreagă gamă de posibilităţi de a spori performanţele unui subsistem de harddisk.

Optimizarea tranzacţiilor

Controller-ul poate optimiza timpul de acces dacă adună toate cererile de scriere şi citire pentru discul respectiv într-un şir de aşteptare (da, şi cererile stau la coadă!), şi lucrează apoi într-o ordine sortată optimal, astfel ca întotdeauna după o cerere să o satisfacă pe cea următoare în sensul distanţei minime pe disc. Deci ordinea nu mai este cea "naturală", în care au fost emise cererile, ci a dispunerii pe pistele suprafeţei de disc folosite în acel moment, a apropierii pe disc.

Un astfel de controller poate aduce, în cazul optim, cînd şirul de aşteptare este suficient de lung, o îmbunătăţire de cca 20-30 % a timpului de acces.

Cazul optimal implică un şir destul de lung de aşteptare a accesului la disc, şir deja format pentru unitatea de harddisk respectivă. Dacă nu-i cazul (de exemplu, dacă nu-s multe cereri, deci şirul de aşteptare este foarte scurt) îmbunătăţirea este insesizabilă.

Pe lingă optimizarea prin sortare după distanţa pe disc, aproape toate controllerele MSCP fac optimizare la cereri de mai multe blocuri, acestea nu vor fi transmise în succesiunea lor naturală, ci în acea ordine care permite harddisk-ului să citească tot în jumătatea interioară a pistei respective, făcînd deci economie de o jumătate de rotaţie. Cum această metodă de optimizare a performanţei este independentă de tipul unităţii de disc, nu este un criteriu de alegere a unui anumit tip de unitate de disc, ci doar o indicaţie (controllerul incorporat să "ştie" acest mod de optimizare).

Pentru a spori performanţa sistemului, se poate "instala" logic un disc mai mare, format din mai multe discuri fizice mai mici. Este inversul partiţionării unui disc în mai multe discuri logice mai mici. Discul logic mare aduce două îmbunătăţiri importante performanţei:

- în primul rînd, unităţile de disc separate se pot poziţiona suprapus;
-apoi, poziţionarea se desfăşoară pe de-a-ntregul atunci cînd capetele de citire/scriere sînt încă pe pista dorită. Probabilitatea respectivă este mai mare la mai multe discuri mici decît la unul mare.

Fragmentarea nu trebuie să fie excesivă - unităţile de disc mici, de 20-40 Mo, au nu numai costul specific mai mare, dar şi timpul de acces mai prost decît un harddisk mai mare, de 120-300 Mo. Se înlocuieşte deci parţial selectarea mecanică a pistelor cu selectarea electronică a unităţilor de disc, iar fiabilitatea sistemului creşte: în caz de pană la un disc, nu sînt afectate toate datele, ci doar cele de pe unitatea respectivă.

Atunci cînd capacitatea maximă de transfer a unui controller sau a unei magistrale de date este depăşită, se poate face o împărţire (repartiţie) a discurilor pe mai multe controllere sau mai multe magistrale de sistem, sau într-un "cluster" cu mai multe calculatoare.

În mod normal, o asemenea împărţire/separare aduce avantaje de viteză, dar mai mici decît se speră.

Cache

O altă posibilitate, ulterioară, de creştere a performanţei harddiskului este folosirea unui controller cu "cache", memorie-tampon de "prindere". ("Cache" se pronunţă "caş" şi vine din franceză: cacher = a ascunde, cache = ascunzătoare).

Mic dicţionar - harddiskuri

Multisegmentare

Unităţile clasice de disc de masă aveau un număr fix de sectoare pe pistă, indiferent de poziţia pistei. Ori, pistele fiind dispuse concentric pe disc, este evident că pe cele exterioare este spaţiu mai mult; cu cît pista este mai aproape de centrul de rotaţie, cu atît mai "strîmt" devine spaţiul pentru sectoarele ce trebuie să încapă pe pistă. Restricţia cu numărul sectoarelor se datora controllerelor - realizate pe atunci cu circuite cu grad de integrare redus (TTL-uri MSI). Nevoia presantă de capacităţi de memorare mai mari - dublată de progresele uriaşe în domeniul circuitelor VLSI, ULSI şi ASIC - a dus la apariţia unor controllere cu funcţii deosebit de complexe, dispunînd de algoritmi sofisticaţi de gestiune a spaţiului fizic al discului. Controllerele cu firmware de multisegmentare împart pistele într-un număr variabil de sectoare, în funcţie de poziţia pistei - mai multe spre exterior, mai puţine spre interior.

Fiecare cerere de scriere pe disc va fi procesată exact ca şi la un controller fără cache: fiecare din blocurile de înregistrat este trimis la capetele de citire/scriere ale unităţii de disc cuplate. În plus însă, controllerul îşi "notează" adresele blocurilor şi conţinutul lor într-o memorie internă, denumită cache. Memoria aceasta are o capacitate variabilă în funcţie de tipul unităţii de harddisk. Astfel, la controllerul QD-35 al firmei Emulex (pentru pînă la patru unităţi de harddisk cu interfaţă SMB, viteza de transfer de max. 3 Mo/s) memoria cache pentru stocarea intermediară a blocurilor de date şi adrese are 1 Mo.

Cînd calculatorul vrea să citească un bloc, controllerul verifică mai întîi dacă blocul respectiv nu se află cumva în memoria cache. Dacă da, va fi transmis imediat calculatorului, fără nici o întîrziere datorită poziţionării, latenţei, etc. Dacă nu se află în cache, va fi citit de pe disc şi transmis, ca în cazul anterior. Pe deasupra, QD-35 reţine şi în această situaţie adresa şi conţinutul blocului.

Cînd memoria-tampon (bufferul-cache) s-a umplut, se va suprascrie, începînd cu blocul care n-a mai fost utilizat/solicitat de cel mai mult timp - o procedură gen FIFO. Se defineşte un coeficient de reuşită, "hit-rate", a "prinderii" datelor în cache. Cu un cache-controller performant, coeficientul de reuşită poate ajunge chiar la 100%: adică, totdeauna cînd calculatorul are nevoie de date de pe disc, ele sînt deja disponibile în cache. Cifra variază în funcţie de împrejurări, de aplicaţia concretă.

Într-adevăr, un controller cu cache este deosebit de recomandabil atunci cînd sînt îndeplinite două sau trei din următoarele condiţii:

-calculatorul are memoria principală relativ redusă;

-unitatea de harddisk primeşte un mare număr de cereri (ca şi la "şirul de aşteptare" - dacă nu sînt multe cereri de citire, nu are ce să optimizeze);

-numărul tipic de blocuri per cerere de citire/scriere este redus;

-se folosesc aplicaţii care, în mod repetat, citesc şi scriu în acelaşi fişier, cu schimbul.

Controllere pentru discuri SMB

Harddisk-urile cu interfaţă SMB sînt performante (capacitate mare, debit ridicat) şi destul de scumpe, ceea ce explică de ce controllerele lor beneficiază de cele mai multe perfecţionări. Emulează protocolul MSCP şi au în consecinţă multe avantaje. Spre deosebire de alte emulări, MSCP lucrează independent de mărimea discului, de structura sa fizică. Softul calculatorului "întreabă" controllerul despre geometria/organizarea harddiskului conectat. La MSCP, sistemul vede doar blocuri numerotate în ordine crescătoare, nu piste, cilindrii, sectoare sau feţe de disc - toate aceste detalii fizice sînt "transparente". De asemenea, calculatorul nu mai are treabă cu blocurile cu defecte, bad blocks, tratarea on-line (în timp real) a defectelor fiind şi ea transparentă.

Minimizarea timpului de poziţionare dintre două cereri de citire succesive, menţionată anterior, se obţine cu ajutorul unui aşa numit "algoritm de ascensor". Întodeauna cea mai apropiată cerere din şirul de aşteptare va fi rezolvată - în ordinea distanţei de poziţionare pe disc. Pentru ca o anumită cerere să nu rămînă nerezolvată un timp foarte lung, există un indicator de "tratare cinstită", Fairness Count, şi un şir de aşteptare expres, care asigură soluţionarea rapidă a cererii corespunzătoare.

Controllerele MSCP au şi alte caracteristici avansate, cum ar fi:

-FHS (Fast Head Select). Din moment ce organizarea fizică a discului nu contează, controllerul are posibilitatea de a comuta capetele după necesităţi, pe timpul transferului de date, astfel încît să nu mai fie necesară o rotaţie suplimentară a discului;

- RPS (Rotational Position Sensing), aduce în primul rînd avantajul că, atunci cînd controllerul este legat la mai mult de o unitate de disc, va da întotdeauna cale liberă mai întîi datelor de la/pentru harddisk-ul ale cărui capete au de parcurs drumul minim pînă la poziţionarea corectă, ş. a. m. d.

-A-DMA, accesul direct adaptiv la memorie, reglează eliberarea magistralei de sistem (în general

Noutăţi

Harddisk-urile de 2,5" fiind ultimele apărute, beneficiază de perfecţionări în domeniul algoritmilor specifici precum şi-n hard-ul aferent (circuite VLSI specializate):

- multisegmentare
- cache adaptiv, ce evaluează modul în care datele sînt cerute de sistem şi adoptează automat algoritmul cel mai eficient
- cache cu predicţie, care minimizează timpul de acces la transferuri lungi
- consum redus de energie (2-3 W şi chiar mai puţin în "standby")
- corecţie automată a erorilor (ECC - cod polinomial de 48 biţi, permite corecţia unui sector cu "salvă" de pînă la 11 biţi eronaţi; rata erorilor scade de 1000 de ori (!)
- firmware integrat de autodiagnostic, foarte complet, care identifică şi semnalează prompt condiţiile de eroare
- robusteţe mecanică f. bună - rezistă la şocuri de 100 G cînd nu e în stare de funcţionare - datorită dimensiunilor reduse şi a sistemului automat de "parcare" a capetelor în zone special destinate
- sistem de servopoziţionare hibrid, folosind suprafaţa discului pentru a reduce timpul de acces la valori de ordinul a 10 ms şi servo înglobat (pistă specială cu semnale servo) pentru îmbunătăţirea preciziei de poziţionare a capetelor

Q-bus) cu prioritate mai mică sau mai mare în funcţie de transferul de date dintre harddisk şi calculator.

Fiecare din aceste caracteristici avansate optimizează cîte unul din aspectele performanţei discului : FHS elimină în mare parte timpul de latenţă, RPS îl reduce pe cel de poziţionare, A-DMA diminuează pierderile de timp la transferurile de date.

Funcţia denumită "optimizarea căutării" asigură, împreună cu bufferul de comenzi (ce poate stoca pînă la 13 comenzi succesive), procesarea cererilor de citire/scriere în ordine logică, şi nu pur şi simplu în ordinea apariţiei lor.

În plus controllerul conţine în EPROM (firmware) programe de diagnostic (autotest), de formatare a harddisk-ului şi de tratare a erorilor, astfel că discul poate fi formatat şi verificat fără programe speciale, încărcate de pe dischetă.

Controllerul QD-35 are, în afara funcţiilor descrise mai sus şi a memoriei cache, un procesor intern foarte rapid: un 68020 în capsulă ceramică Quadflatpack. Interesant este cum se face "prinderea" datelor - la eficienţa unei memorii cache contribuind nu numai mărimea acesteia, ci algoritmii utilizaţi. Firma Emulex declară că la proiectarea lui QD-35 a avut în vedere în mod special găsirea unei modalităţi eficiente de "prindere", fără influenţele negative ce se întîlnesc la alte controllere cu cache asupra performanţelor la lucrul cu mai multe unităţi de disc (MDP) sau la sortarea comenzilor (CS).

La QD-35, pentru prima dată, memoria cache este configurabilă după dorinţă. Utilizatorul are posibilitatea de a stabili ce domeniu al discului să fie "prins"şi care nu. În acest scop, la QD-35 (care este un veritabil minisistem) se leagă un terminal, prin intermediul unui cablu plat. Monitorul incorporat lui QD-35 permite setarea şi reglarea parametrilor, supraveghează tranzacţiile cu harddisk-ul şi dă informaţii statistice asupra valorilor timpilor de acces şi a eficacităţii "prinderii", permiţînd astfel stabilirea parametrilor optimi. Atenţie! O memorie cache aşa de mare poate prezenta şi dezavantaje : dacă alimentarea sistemului de calcul nu este asigurată de o sursă neîntreruptibilă (UPS - Uninterruptible Power Supply), în caz de pană de curent toate informaţiile din memoria controllerului se pierd ...

Interfaţa

Evident, controllerul şi unitatea de disc trebuie să aibă aceeaşi interfaţă. Actualmente, majoritatea harddisk-urilor uzuale (de 3,5" şi 5,25", capacităţi pînă în 300 Mo) au controllerul incorporat ("imbedded" - se vorbeşte atunci de interfaţă AT sau IDE - Intelligent Drive Electronics). Sînt mai răspîndite 3 tipuri de interfeţe:

ST506

Interfaţa controllerului original de pe IBM-XT, pentru discuri de capacitate nu prea mare (pînă în 150 Mo). Are două variante, corespunzătoare modului de înregistrare pe disc: MFM (Modified Frequency Modulation) şi RLL (Run Length Limited). Cu RLL, datele sînt procesate suplimentar: prin intermediul unei logici complicate, se obţin coduri în care pot exista între 1 şi 7 zerouri succesive (RLL 1.7) sau între două şi şapte zerouri succesive (RLL 2.7). Se obţine astfel o îmbunătăţire suplimentară a performanţelor, viteza de transfer poate creşte cu 50%. (încă o dată trebuie subliniat că este vorba de transferul între harddisk şi controller. Discul nu trebuie să dea date controllerului cu o viteză mai mare decît cea cu care le poate prelua memoria. Bufferul controllerului nu trebuie să se umple complet într-una sau mai multe rotaţii ale discului. O rată de transfer internă mai ridicată nu are nici un efect dacă viteza de transfer pe magistrala calculatorului este inferioară). Trebuie însă verificat că unitatea de harddisk acceptă şi modul RLL al controllerului: toate unităţile de disc cu ST506 acceptă modul MFM, dar nu toate pot lucra şi în RLL. Cablele sînt identice şi uneori un controller RLL pare să se înţeleagă cu un disc MFM. Dar, în astfel de cazuri, datele Dvs. sînt în mare pericol!

ESDI

Se foloseşte la capacităţi şi viteze mai mari. Viteza maximă de transfer este de 15 Mo/s. Atenţie! Cablurile şi mufele ESDI (Enhanced Small Device Interface) şi ST506 sînt identice ca aspect. Aceasta creează premise de eroare: se poate cupla din greşeală ieşirea unui harddisk ESDI la intrarea unui controller ST506, ceea ce poate duce la defectarea chiar a ambelor componente: se leagă împreună căi cu diferite funcţii.

SCSI

Cel mai nou standard este SCSI (Small Computer System Interface). Spre deosebire de celelalte,

Controllerele - tabel comparativ

	ST506		ESDI	SCSI
	MFM	RLL		asincron/sincron
rata maximală de transfer (interleave 1:1) la -interfaţa serială (Mbit/s)	5	7,5	10	12/40
-unitate (Mbit/s)			10/15	10/15
-sistem (ko/s)	450	650	850/900-1200	850/1200
busul de date	serial	serial	serial	8 (16) biţi
BIOS integrat	nu	da	da	da

nu este o simplă interfaţă de harddisk, ci o veritabilă magistrală de intrare/ieşire. Pe magistrală SCSI se pot lega în paralel mximum 8 surse de date, 8 noduri. Cele 8 noduri pot fi un calculator şi mai multe unităţi de harddisk, sau alte periferice: scanner, unitate de bandă magnetică, imprimantă, disc magneto optic, etc.

Magistrala SCSI "dă" mai multă inteligenţă perifericelor şi cere mai puţin plăcii de legătură a calculatorului. Discurile sub SCSI pot lucra în paralel, dacă sistemul de operare acceptă mai mult de o singură cerere de harddisk la un moment dat, adică numai cu sisteme de operare multitasking precum Unix, OS/2 sau Novell, nu MS-DOS. Şi cu asta se trece şi la capitolul următor:

Sistemul de operare

Iniţial, am neglijat timpul de overhead al sistemului de operare, fiind acelaşi pentru toate tipurile de harddisk-uri. Totuşi, sistemul de operare afectează modul de lucru cu discul, este un criteriu important de decizie în alegerea tipului de harddisk. Astfel, putem avea un harddisk de capacitate foarte mare, dar sistemul de operare să nu se potrivească. De exemplu, MS-DOS "consideră" că discurile au maximum 1024 de cilindri. Această limitare a sistemului de operare poate fi depăşită chiar dacă harddisk-ul are mai mult de o mie de cilindri (piste per faţă de disc): cineva, de regulă controllerul, va masca realitatea de pe disc, disponibilizînd însă mai multe feţe de disc (false) sau mai multe sectoare pe o pistă (false). În această tehnică lucrează controllerele din seria ACB23 a lui Adaptec sau WD1007Val lui Western Digital.

MS-DOS este un sistem de operare single-task; dar sub VMS, OS, Novell şi Unix se poate lucra în multitasking, cu mai multe unităţi de disc mici - cum s-a arătat mai înainte, pînă ce unul îşi poziţionează capetele, ne adresăm altuia pentru transferul de date.

Transportabilitatea

Este un criteriu de selecţie esenţial atunci cînd se impune ca datele să poată fi transportate rapid şi uşor de la un calculator la altul, sau cînd sînt date sensibile, ce trebuiesc protejate.

La harddisk-urile amovibile, unitatea este într-o cutie-sertar cu conectori, cutie care poate fi uşor detaşată pentru transport. Unitatea de bază este un şasiu, care conţine sursa de alimentare şi conectorii, şi poate accepta unul sau două containere cu discuri. Containerele, etanşe, cuprind un mediu controlat, clasă 100 (sub 100 de particule de praf / ft^3), în care funcţionează discurile.

Cum fiecare container cuprinde unitatea de harddisk completă, costul este ridicat, atît pentru subsistemul propriu-zis, cît şi pentru suportul de date. O soluţie mult mai economică o oferă discurile-cartuş (cartridge), gen Bernoulli-Box, la care se schimbă numai suportul de date, nu toată unitatea.

Totuşi, harddisk-urile amovibile în container se utilizează pe larg, avînd două avantaje majore:

-sînt cele mai bune atunci cînd este necesar să se transporte cantităţi mari de date, dar nu există timpul pentru a le capta pe bandă sau casetă magnetică;

-sînt foarte bune şi atunci cînd datele unui sistem trebuiesc puse în siguranţă, în întregime şi în mod rapid - de exemplu containerele se încuie peste noapte în seif.

Faţă de transportul sau asigurarea (back-up) datelor prin copierea pe bandă, harddisk-urile amovile prezintă un timp de acces mult mai redus, iar probabilitatea de apariţie a unor erori la copiere este considerabil diminuată. Un alt suport de date interschimbabil ce le-ar putea face concurenţă este discul optic reînregistrabil (disc magneto-optic), dar acesta prezintă actualmente inconveniente ca:

-deşi suportul de date este ieftin pentru capacitatea sa, unitatea de disc magneto-optică este foarte scumpă;

-timpii de acces sînt mari, de 4 ori mai mari la citire şi de 10 ori la scriere, în timp ce discurile amovibile au timpi de răspuns comparabili cu cei ai harddisk-urilor Winchester normale;

-şi rata de transfer este mai scăzută, 0,5-1 Mo/s faţă de 2-5 Mo/s la harddisk-urile normale.

Alte criterii

Actualmente, cele mai răspîndite sînt harddisk-urile de 5,25" (fi 13 cm), care există în două înălţimi diferite ale cutiei:

-înălţime întreagă (FH - full height, un model de 10,5 cm), cu capacităţi de peste 300 Mo, ajungînd pînă la 1200 Mo;

-semi-înălţime (HH - half height, 5,25 cm), relativ răspîndită la discurile pînă în 200 Mo.

Discurile de 8" (20 cm), cu interfeţe tip SMB sau SCSI, au capacităţi de pînă la 2000-3200 Mo.

Cele mai mari unităţi de disc au diametrul de 9", dar sînt puţin răspîndite, nefiind competitive nici ca performanţe, nici ca preţ (costisitoare, dimensiuni, consum şi greutate foarte ridicate).

ing. Cristian Mulide, I.C.C.E.

Filecard

Este numele sub care sînt cunoscute harddisk-urile ce se montează direct într-un slot liber al PC-ului, fiind solidare cu cartela care se introduce şi pe care sînt montate. De regulă sînt mai scumpe decît harddisk-urile clasice; sînt o alternativă bună cînd trebuie extinsă capacitatea disc dar controllerul este deja "epuizat" sau nu mai este loc prevăzut pentru o unitate suplimentară de harddisk.

Firma	Model	Preţ (DM)	Formă	Capacitate (Mo)	Timp de acces (ms)	Rata de transfer (Mo/s)	Tip interfaţă	Mod de înregistrare	Cache integrat (ko)	Cilindri/capete	MTBF (mii ore)
Adcomp	AL DTurbo-Disk		ext. 2 x 5,25" FH	/330	10,5	5	SCSI	Zoom-bit recording	32		
Conner	P-3044	581	3,5"-Winchester	43/40	25	1,5	AT	RLL (2,7)			
	CP-3104	1026	-||-	/105	25	1,25	AT	RLL(2,7)			
	CP-3204F	1676	-||-	/212	16	1,5	AT	RLL(2,7)			
	HOPI CP30104		3,5"	/120	16	1,5/4,0	AT/EISA	RLL	64		
	RAMBO CP3204F		3,5"	/212	16	1,5/5,0	AT	RLL	64		
Data General	6557	2428	3,5"	/104	25	1,25	IDE	RLL			
Daimler International	Legacy Mass Tower I/380	9151	subsistem extern	/380	>15	2,7	SCSI	RLL			
Fujitsu	M2611SA		3,5" HH	/45	25	1,5	SCSI	RLL (1,7)	24	1334/2	50
	M2614ESA	2438	3,5"	/180	20	1,25	SCSI	RLL (1,7)	24	1334/4	
	M2266SA	9776	5,25"	1079	14,5	4	SCSI	RLL (1,7)	128	1658/15+1	200
	M2263SA	7085	5,25"	672	16	4	SCSI	RLL (1,7)	64	1658/15+1	200
	M2246SA	1.850		172 141	25		SCSI				
	M2622SA	2.630	3,5" FH	/330	12,5	3	SCSI-2	RLL (1,7)	4 x 60	1435/7+1	75
	M2623SA	3.050	3,5" FH	/425	12	3	SCSI-2	RLL (1,7)	4 x 60	1435/9+1	
	M2624SA	3.480	3,5" FH	/520	12	3	SCSI-2	RLL (1,7)	4 x 60	1435/11+1	
Hewlett-Packard	HP C2233S		3,5"	/234	13	2,5	SCSI-2		64	2172/15	
Hitachi	DK516C-16		5,25"	1652/	13,5		SCSI				
IBM	Modell 60/80	1.250	5,25"	44	40		ST-506				
	PS/2	2.880	3,5"	60	27		ESDI				
	PS/2	2095	3,5"	60	23		SCSI				
	PS/2	12.860	5,25"	314	23		ESDI				
Kyocera	KC40GA		3,5"	/40	28		ST506	RLL		1069/2	
Maxtor	EM 8051A3		3,5"	/42	28	1,0	IDE-AT	RLL	32		
	XT-2190		5,25"	160	29	5	ST506	MFM			
	XT-4380		5,25"	/338	16	4,8	SCSI ESDI	RLL (2,7)	45/SCSI		
Micropolis	1528-15		5,25"	/1354	14	1,6/5,0	SCSI-2	RLL	256	2106/15	
Mitsubishi Electric	MR 335		3,5"	40	20	5	ST506	MFM		612/5	
NCR		1.960	3,5"	/42	25	1,25	AT	RLL			
NEC	D3761		3,5"	/733	20		AT	RLL (2,7)		733/8	
Plus Development	Filecard80	1.664	AT 3,5" Winchester	/80	28	0,7	EISA	RLL			
Procom Technology	PhD 45	1.450	3,5"	/44,7	25	5,0	IBM PS/2	MFM			
	Pira 50	1.505	3,5" Winchester	/45	25	0,625	ST-506	MFM			
	Si 200	3.595	3,5"	/209	18	12	Compaq Deskpro	RLL/SCSI			
Quantum	LPS 52S	718	3,5"	/52	< 17	2	SCSI		64	1219/2	
	LPS 52 AT	798	3,5"	/52	< 17	4	AT	RLL(1,7)	64	1219/2	
	LPS 105 AT	1.398	3,5"	105	< 17	0,8	IDE	RLL (1,7)	64	1219/2	
	LPS 105 S	1.173	3,5"	/105	< 17	2/4	SCSI		64	1219/2	
	Pro Drive 40S	618	3,5"	/42	19	2/4	SCSI	RLL (2,7)		834/3	
	Pro Drive 80S	927	3,5"	/84	19	2/4	SCSI	RLL(2,7)		834/6	
	Pro Drive 120AT		3,5"	/120	< 15	4	AT	RLL (1,7)		1123/5	
	Pro Drive 210AT		3,5"	/210	< 15	4	AT	RLL (1,7)		1156/7	
Rodime	RO 3000 AT		3,5"	45-210	18	max 2,5	AT (IDE)	RLL(2,7)	32/64		
Seagate	ST 325N		3,5"	/21,4	28		SCSI	RLL(2,7)			
	ST125N		3,5"	/21,5	28	1,5	SCSI	RLL(2,7)			
	ST138N		3,5"	/32,2		1,5	SCSI	RLL(2,7)			
	ST157N		3,5"	/48,6	20	1,5	SCSI	RLL (2,7)			
	ST177N		3,5"	/60,8	20	1,5	SCSI	RLL(2,7)			
	ST1096N		3,5"	/83,9	15	1,5	SCSI	RLL (2,7)			
	ST1126N		3,5"	/107,0	15	1,5	SCSI	RLL (2,7)			
	ST1133N		3,5"	/113,4	15	1,25	SCSI	RLL(2,7)			
	ST1162N		3,5"	/137,5	15	5	SCSI	RLL (2,7)			
	ST1186N		3,5"	/158,8	15	5	SCSI	RLL (2,7)			
	ST1201N		3,5"	171,9	15	5	SCSI	RLL (2,7)			
	ST1239N		3,5"	204,2	15	5	SCSI	RLL (2,7)			
Toshiba	MK 134FA		3,5"	/40	25	5	ST-506				
Western Digital	Filecard 40	752	3,5"	/40	29	0,9	ISA	RLL			

Rutine care clasifică

Aceste rutine clasifică caracterele ASCII în litere, caractere de control, semne de punctuaţie, majuscule, etc.

	isalnum(c)	<ctype.h>
	isalpha(c)	<ctype.h>
	isascii(c)	<ctype.h>
	iscntrl(c)	<ctype.h>
	isdigit(c)	<ctype.h>
	isgraph(c)	<ctype.h>
	islower(c)	<ctype.h>
	isprint(c)	<ctype.h>
	ispunct(c)	<ctype.h>
	isspace(c)	<ctype.h>
	isupper(c)	<ctype.h>
	isxdigit(c)	<ctype.h>

Rutine de conversie

Aceste rutine convertesc şiruri alfanumerice în diferite reprezentări numerice şi invers, din majuscule în minuscule şi invers

double	atof(const char *s)	<stdlib.h>
int	atoi(const char *s)	<stdlib.h>
long	atol(const char *s)	<stdlib.h>
char	*ecvt(double value, int ndig, int *dec, int *sign)	<stdlib.h>
char	*fcvt(double value, int ndig, int *dec, int *sign)	<stdlib.h>
char	*gcvt(doublevalue, int ndec, char *buf)	<stdlib.h>
char	itoa(int value, char *string, int radix)	<stdlib.h>
char	ltoa(long value, char *string, int radix)	<stdlib.h>
double	strtod(const char *s, char **endptr)	<stdlib.h>
long	strtol(const char *s, char **endptr, int radix)	<stdlib.h>
unsigned long	strtoul(const char *s, char **endptr, int radix)	<stdlib.h>
	toascii(c)	<ctype.h>
	_tolower(c)	<ctype.h>
int	tolower(int ch)	<ctype.h>
	_toupper(c)	<ctype.h>
int	toupper(int ch)	<ctype.h>
char	ultoa(unsigned long value, char *string, int radix)	<stdlib.h>

Rutine de control al directoarelor

Aceste rutine manipulează directoare şi nume de căi

int	chdir(const char *path)	<dir.h>
int	findfirst(const char *pathname, struct ffblk *ffblk, int attrib)	<dir.h>
int	findnext(struct ffblk *ffblk)	<dir.h>
void	fnmerge(char *path, const char *drive, const char *dir, const char *name, const char *ext)	<dir.h>
int	fnsplit(char *path, const char *drive, const char *dir, const char *name, const char *ext)	<dir.h>
int	getcurdir(int drive, char *directory)	<dir.h>
char	*getcwd(char *buf, int buflen)	<dir.h>
int	getdisk(void)	<dir.h>
int	mkdir(const char *path)	<dir.h>
char	*mktemp(char *template)	<dir.h>
int	rmdir(const char *path)	<dir.h>
char	*searchpath(const char *file)	<dir.h>
int	setdisk(int drive)	<dir.h>

Rutine de diagnosticare

Aceste rutine asigură capacitatea de depanare înglobată în programe

void	assert(int test)	<assert.h>
int	matherr(struct exception *e)	<math.h>

int	freemem(unsigned segx)	<dos.h>
void	geninterrupt(int intr_num)	<dos.h>
int	getcbrk(void)	<dos.h>
void	getdfree(unsigned char drive, struct dfree *dtable)	<dos.h>
char far	*getdta(void)	<dos.h>
void	getfat(unsigned char drive, struct fatinfo *dtable)	<dos.h>
void	getfatd(struct fatinfo *dtable)	<dos.h>
unsigned	getpsp(void)	<dos.h>
void	interrupt (*getvect(int interrupno)) ()	<dos.h>
int	getverify(void)	<dos.h>
void	harderr(int (*handler) ())	<dos.h>
void	hardresume(int axret)	<dos.h>
void	hardretn(int retn)	<dos.h>
int	inport(int portid)	<dos.h>
unsigned char	inportb(int portid)	<dos.h>
int	int86(int intno, union REGS *inregs, union REGS *outregs)	<dos.h>
int	int86x(int intno, union REGS *inregs, union REGS *outregs, struct SREGS *segregs)	<dos.h>
int	intdos(union REGS *inregs, union REGS *outregs)	<dos.h>
int	intdosx(union REGS *inregs, union REGS *outregs, struct SREGS *segregs)	<dos.h>
void	intr(int intno, struct REGPACK *preg)	<dos.h>
void	keep(unsigned char status, unsigned size)	<dos.h>
void far *	MK_FP(unsigned seg, insigned ofs)	<dos.h>
void	outport(int portid, int value)	<dos.h>
void	outportb(int portid, unsigned charvalue)	<dos.h>
char	*parsfnm(const char *cmdline. struct fcb *fcb, int opt)	<dos.h>
int	peek(unsigned segment, unsigned offset)	<dos.h>
char	peekb(unsigned segment, unsigned offset)	<dos.h>
void	poke(unsigned segment, unsigned offset, int value)	<dos.h>
void	pokeb(unsigned segment, unsigned offset, char value)	<dos.h>
int	randbrd(struct fcb *fcb, int rcnt)	<dos.h>
int	randbwr(struct fcb *fcb, int rcnt)	<dos.h>
void	segread(struct SREGS *segp)	<dos.h>
int	setcbrk(int cbrkvalue)	<dos.h>
void	setdta(char far *dta)	<dos.h>
void	setvect(int interruptno, void interrupt (*isr) ( ))	<dos.h>
void	setverify(int value)	<dos.h>
void	sleep(unsigned seconds)	<dos.h>
int	unlink(const char *filename)	<dos.h, io.h, stdio.h>

Rutine de manipulare şiruri şi zone de memorie

void	*memccpy(void *dest, const void *src, int c, size_t n)	<mem.h, string.h>
void	memchr(const void *s, int c, size_t n)	<mem.h, string.h>
int	memcmp(const void *s1, const void *s2, size_t n)	<mem.h, string.h>
void	*memcpy(void *dest, const void *src, size_t n)	<mem.h, string.h>
void	*memmove(void *dest, const void *src, size_t n)	<mem.h, string.h>
void	*memset(void *s, int c, size_t n)	<mem.h, string.h>
void	movedata(unsigned srcseg, unsigned srcoff, unsigned destseg, unsigned destoff, size_t n)	<mem.h, string.h>
void	movmem(void *src, void *dest, unsigned length)	<mem.h, string.h>
void	setmem(void *dest, int length, char value)	<mem.h>
char	*stpcpy(char *dest, const char *src)	<string.h>
char	*strcat(char *dest, const char *src)	<string.h>
char	*strchr(const char *s, int c)	<string.h>
int	strcmp(const char *s1, const char *s2)	<string.h>
int	strcoll(char *s1, char *s2)	<string.h>
char	*strcpy(char *dest, const char *src)	<string.h>
size_t	strcspn(const char *s1, const char *s2)	<string.h>
char	*strdup(const char *s)	<string.h>
char	*strerror(int errnum)	<string.h>
size_t	strlen(const char *s)	<string.h>
char	*strlwr(char *s)	<string.h>
char	*strncat(char *dest, const char *src, size_t maxlen)	<string.h>
int	strncmp(const char *s1, const char *s2, size_t maxlen)	<string.h>
char	*strncpy(char *dest, const char *src, size_t maxlen)	<string.h>
char	*strnset(int *s, int ch, size_t n)	<string.h>
char	*strpbrk(const char *s1, const char *s2)	<string.h>
char	*strrchr(const char *s, int c)	<string.h>
char	*strrev(char *s)	<string.h>
char	*strset(char *s, int ch)	<string.h>
size_t	strspn(const char *s1, const char *s2)	<string.h>
char	*strstr(const char *s1, const char *s2)	<string.h>
char	*strtok(char *s1, const char *s2)	<string.h>

Turbo C++ 1.0
Memento - funcţii

```
void      perror(const char *s)                                   <errno.h>
```

Rutine grafice

Aceste rutine permit realizarea de grafică şi text pe ecran

```
void far  arc(int x, int y, int stangle, int endangle, int radius)          <graphics.h>
void far  bar(int left, int top, int right, int bottom)                     <graphics.h>
void far  bar3d(int left, int top, int right, int bottom,
                int depth, int topflag)                                     <graphics.h>
void far  circle(int x, int y, int radius)                                  <graphics.h>
void far  cleardevice(void)                                                 <graphics.h>
void far  clearviewport(void)                                               <graphics.h>
void far  closegraph(void)                                                  <graphics.h>
void far  detectgraph(int far *graphdriver, int far *graphmode)             <graphics.h>
void far  drawpoly(int numpoints, int far *polypoints)                      <graphics.h>
void far  ellipse(int x, int y, int stangle, int endangle,
                int xradius, int yradius)                                   <graphics.h>
void far  fillellipse(int x, int y, int xradius, int yradius)               <graphics.h>
void far  fillpoly(int numpoints, int far *polypoints[])                    <graphics.h>
void far  floodfill(int x, int y, int border)                               <graphics.h>
void far  getarccoords(struct arccoordstype, far *arccords)                 <graphics.h>
void far  getaspectratio(int far *xasp, int far *yasp)                      <graphics.h>
int far   getbkcolor(void)                                                  <graphics.h>
int far   getcolor(void)                                                    <graphics.h>
struct palettetype *far  getdefaultpalette(void)                            <graphics.h>
char *far getdrivername(void)                                               <graphics.h>
void far  getfillpattern(char far *pattern)                                 <graphics.h>
void far  getfillsettings(struct fillsettingstype far *fillinfo)            <graphics.h>
int far   getgraphmode(void)                                                <graphics.h>
void far  getimage(int left, int top, int right, int bottom,
                void far *bitmap)                                           <graphics.h>
void far  getlinesettings(struct linesettingstype far *lineinfo)            <graphics.h>
int far   getmaxcolor(void)                                                 <graphics.h>
int far   getmaxmode(void)                                                  <graphics.h>
int far   getmaxx(void)                                                     <graphics.h>
int far   getmaxy(void)                                                     <graphics.h>
char * far getmodename(int mode_number)                                     <graphics.h>
void far  getmoderange(int graphdriver, int far *lomode,
                int far *himode)                                            <graphics.h>
void far  getpalette(struct palettetype far *palette)                       <graphics.h>
int far   getpalettesize(void)                                              <graphics.h>
unsigned far  getpixel(int x, int y)                                        <graphics.h>
void far  gettextsettings(struct textsettingstype far *
                texttypeinfo)                                               <graphics.h>
void far  getviewsettings(struct viewporttype far *viewport)                <graphics.h>
int far   getx(void)                                                        <graphics.h>
int far   gety(void)                                                        <graphics.h>
void far  graphdefaults(void)                                               <graphics.h>
char *far grapherrormsg(int errorcode)                                      <graphics.h>
void far  _graphfreemem(void far *ptr, unsigned size)                       <graphics.h>
void far * far _graphgetmem(unsigned size)                                  <graphics.h>
int far   graphresult(void)                                                 <graphics.h>
unsigned far  imagesize(int left, int top, int right, int bottom)           <graphics.h>
void far  initgraph(int far *graphdriver, int far *graphmode,
                char far *pathtodriver)                                     <graphics.h>
int far   installuserdriver(char far *name,
                int huge (*detect) (void))                                  <graphics.h>
int far   installuserfont(char far *name)                                   <graphics.h>
void far  line(int x1, int y1, int x2, int y2)                              <graphics.h>
void far  linerel(int dx, int dy)                                           <graphics.h>
void far  lineto(int x, int y)                                              <graphics.h>
void far  moverel(int dx, int dy)                                           <graphics.h>
void far  moveto(int x, int y)                                              <graphics.h>
void far  outtext(char far *textstring)                                     <graphics.h>
void far  outtextxy(int x, int y, char far *textstring)                     <graphics.h>
void far  pieslice(int x, int y, int stangle, int endangle, int radius)     <graphics.h>
void far  putimage(int left, int top, void far *bitmap, int op)             <graphics.h>
void far  putpixel(int x, int y, int color)                                 <graphics.h>
void far  rectangle(int left, int top, int right, int bottom)               <graphics.h>
int       registerbgidriver(void (*driver) (void))                          <graphics.h>
int       registerbgifont(void (*font)(void))                               <graphics.h>
void far  restorecrtmode(void)                                              <graphics.h>
```


Turbo C++ 1.0
Memento - funcţii


```
char      *strupr(char *s)                                          <string.h>
size_t    strxfrm(char *s1, char *s2, size_t n)                     <string.h>
```

Rutine matematice

```
[int | double] abs([int | complex] x)                    <math.h, complex.h, stdlib.h>
[double | complex] acos([double | complex] x)                    <complex.h, math.h>
double    arg(complex z)                                                <complex.h>
[double | complex] asin([double | complex] x)                    <complex.h, math.h>
[double | complex] atan([double | complex] x)                    <complex.h, math.h>
double    atan2(double y, double x)                                     <complex.h, math.h>
double    atof(const char *s)                                   <stdlib.h, math.h>
int       atoi(const char *s)                                           <stdlib.h>
long      atol(const char *s)                                           <stdlib.h>
bcd       bcd(int x)                                                      <bcd.h>
bcd       bcd(double x)                                                   <bcd.h>
bcd       bcd(double x, int decimals)                                     <bcd.h>
double    cabs(struct complex z)                                        <math.h>
double    ceil(double x)                                                <math.h>
unsigned int  _clear87(void)                                            <float.h>
complex   complex(double real, double imag)                             <complex.h>
double    conj(complex z)                                               <complex.h>
unsigned int  _control87(unsigned int new, unsigned int mask)           <float.h>
[double | complex] cos([double | complex] z)            <complex.h, math.h>
[double | complex] cosh([double | complex] z)           <complex.h, math.h>
div_t     div(int numer, int denom)                                     <math.h>
char      *ecvt(double value, int ndig, int *dec, int *sign)            <stdlib.h>
[double | complex] exp([double | complex] z)                          <math.h>
double    fabs(double x)                                                <math.h>
char      *fcvt(double value, int ndig, int *dec, int *sign)            <stdlib.h>
double    floor(double x)                                               <math.h>
double    fmod(double x, doubley)                                       <math.h>
void      _fpreset(void)                                                <float.h>
double    frexp(double x, int *exponent)                                <math.h>
char      *gcvt(double value, int ndec, char *buf)                      <stdlib.h>
double    hypot(double x, doubley)                                      <math.h>
double    imag(complex z)                                               <complex.h>
char      *itoa(long value, char *string, int radix)                    <stdlib.h>
long int  labs(long int x)                                              <stdlib.h>
double    ldexp(double x, int exp)                                      <math.h>
ldiv_t    ldiv(long int numer, long int denom)                          <math.h>
[double | complex] log([double | complex] x)            <complex.h, math.h>
double    log10(double x)                                 <complex.h, math.h>
unsigned long  _lrotl(unsigned long val, int count)                     <stdlib.h>
unsigned long  _lrotr(unsigned long val, int count)                     <stdlib.h>
          _matherr(enumerated type)                                     <math.h>
int       matherr(struct exception *e)                                  <math.h>
double    modf(double x, double *ipart)                                 <math.h>
double    norm(complex x)                                               <complex.h>
complex   polar(double mag, double angle)                               <complex.h>
double    poly(double x, int degree, double coeffs [])                  <math.h>
[double | complex] pow([double | complex] x,
                [double | complex] y)                   <complex.h, math.h>
double    pow10(int p)                                                  <math.h>
int       rand(void)                                                    <stdlib.h>
int       random(int num)                                               <stdlib.h>
void      randomize(void)                                               <stdlib.h>
double    real(complex x)                                               <complex.h>
unsigned  _rotl(unsigned val, int count)                                <stdlib.h>
unsigned  _rotr(unsigned val, int count)                                <stdlib.h>
[double | complex] sin([double | complex] z)            <complex.h, math.h>
[double | complex] sinh([double | complex] z)           <complex.h, math.h>
[double | complex] sqrt([double | complex] x)           <complex.h, math.h>
void      srand(unsigned seed)                                          <stdlib.h>
unsigned int  _status87(void)                                           <float.h>
double    strtod(const char *s, char **endptr)                          <stdlib.h>
long      strtol(const char *s, char **endptr, int radix)               <stdlib.h>
unsigned long  strtoul(const char *s, char **endptr, int radix)         <stdlib.h>
[double | complex] tan([double | complex] x)            <complex.h, math.h>
[double | complex] tanh([double | complex] x)           <complex.h, math.h>
char      *ultoa(unsigned long value, char *string, int radix)          <stdlib.h>
```

Rutine de gestiune a memoriei

```
int       allocmem(unsigned size, unsigned *segp)                       <dos.h>
int       brk(void *addr)                                               <alloc.h>
```

```
void far    sector(int x, int y, int stangle, int endangle,
                   int xradius, int yradius)                    <graphics.h>
void far    setactivepage(int page)                             <graphics.h>
void far    setallpalette(struct palettetype far *palette)      <graphics.h>
void far    setaspectratio(int xasp, int yasp)                  <graphics.h>
void far    setbkcolor(int color)                               <graphics.h>
void far    setcolor(int color)                                 <graphics.h>
void far    setfillpattern(char far *upattern, int color)       <graphics.h>
void far    setfillstyle(int pattern, int color)                <graphics.h>
unsigned far setgraphbufsize(unsigned bufsize)                  <graphics.h>
void far    setgraphmode(int mode)                              <graphics.h>
void far    setlinestyle(int linestyle, unsigned upattern, int thickness) <graphics.h>
void far    setpalette(int colornum, int color)                 <graphics.h>
void far    setrgbpalette(int colornum, int red, int green, int blue) <graphics.h>
void far    settextjustify(int horiz, int vert)                 <graphics.h>
void far    settextstyle(int font, int direction, int charsize) <graphics.h>
void far    setusercharsize(int multx, int divx, int multy, int divy) <graphics.h>
void far    setviewport(int left, int top, int right, int bottom, int clip) <graphics.h>
void far    setvisualpage(int page)                             <graphics.h>
void far    setwritemode(int mode)                              <graphics.h>
int far     textheight(char far *textstring)                    <graphics.h>
int far     textwidth(char far *textstring)                     <graphics.h>
```

Rutine de intrare/ieşire

Aceste rutine asigură posibilităţi de intrare /ieşire la nivel stream şi DOS

```
int     access(const char *filename, int amode)                 <io.h>
char    *cgets(char *str)                                       <conio.h>
int     _chmod(const char *path, int func [, int attrib])       <io.h>
int     chmod(const char *path, int amode)                      <io.h>
int     chsize(int handle, long size)                           <io.h>
void    clearerr(FILE *stream)                                  <stdio.h>
int     _close(int handle)                                      <io.h>
int     close(int handle)                                       <io.h>
int     cprintf(const char *format [, argument,...])            <conio.h>
int     cputs(const char *str)                                  <conio.h>
int     _creat(const char *path, int attrib)                    <io.h>
int     creat(const char *path, int amode)                      <io.h>
int     creatnew(const char *path, int mode)                    <io.h>
int     creattemp(char *path, int attrib)                       <io.h>
int     cscanf(char *format [, address, ... ])                  <conio.h>
int     dup(int handle)                                         <io.h>
int     dup2(int oldhandle, int newhandle)                      <io.h>
int     eof(int handle)                                         <io.h>
int     fclose(FILE *stream)                                    <stdio.h>
int     fcloseall(void)                                         <stdio.h>
FILE    *fdopen(int handle, char *type)                         <stdio.h>
int     feof(FILE *stream)                                      <stdio.h>
int     ferror(FILE *stream)                                    <stdio.h>
int     fflush(FILE *stream)                                    <stdio.h>
int     fgetc(FILE *stream)                                     <stdio.h>
int     fgetchar(void)                                          <stdio.h>
int     fgetpos(FILE *stream, fpos_t *pos)                      <stdio.h>
char    *fgets(char *s, int n, FILE *stream)                    <stdio.h>
long    filelength(int handle)                                  <io.h>
int     fileno(FILE *stream)                                    <stdio.h>
int     flushall(void)                                          <stdio.h>
FILE    *fopen(const char *filename, const char *mode)          <stdio.h>
int     fprintf(FILE *stream, const char *format [, argument, ... ]) <stdio.h>
int     fputc(int c, FILE *stream)                              <stdio.h>
int     fputchar(int c)                                         <stdio.h>
int     fputs(const char *s, FILE *stream)                      <stdio.h>
size_t  fread(void *ptr, size_t size, size_t n, FILE *stream)   <stdio.h>
FILE    *freopen(const char *filename, const char *mode,
                 FILE *stream)                                  <stdio.h>
int     fscanf(FILE *stream, const char *format [, address, ... ]) <stdio.h>
int     fseek(FILE *stream, long offset, int whence)            <stdio.h>
int     fsetpos(FILE *stream, const fpos_t *pos)                <stdio.h>
int     fstat(int handle, struct stat *statbuf)                 <sys\stat.h>
long    ftell(FILE *stream)                                     <stdio.h>
size_t  fwrite(const void *ptr, size_t size, size_t n, FILE *stream) <stdio.h>
```



```
void        *calloc(size_t nitems, size_t size)                 <alloc.h>
unsigned [long] coreleft(void)                                  <alloc.h>
void far    *farcalloc(unsigned long nunits, unsigned long unitsz) <alloc.h>
unsigned long farcoreleft(void)                                 <alloc.h>
void        farfree(void far *block)                            <alloc.h>
int         farheapcheck(void)                                  <alloc.h>
int         farheapcheckfree(unsigned int fillvalue)            <alloc.h>
int         farheapchecknode(void *node)                        <alloc.h>
int         farheapfillfree(unsigned int fillvalue)             <alloc.h>
int         farheapwalk(struct farheapinfo *hi)                 <alloc.h>
void far    *farmalloc(unsigned long nbytes)                    <alloc.h>
void far    *farrealloc(void far *oldblock, unsigned long nbytes) <alloc.h>
void        free(void *block)                                   <alloc.h, stdlib.h>
int         heapcheck(void)                                     <alloc.h>
int         heapcheckfree(unsigned int fillvalue)               <alloc.h>
int         heapchecknode(void *node)                           <alloc.h>
int         heapwalk(struct heapinfo *hi)                       <alloc.h>
void        *malloc(size_t size)                                <alloc.h, stdlib.h>
void        *realloc(void *block, size_t size)                  <alloc.h, stdlib.h>
void        *sbrk(int incr)                                     <alloc.h>
int         setblock(unsigned segx, unsigned newsize)           <dos.h>
```

Alte rutine

```
void        delay(unsigned milliseconds)                        <dos.h>
struct lconv *localeconv(void)                                  <locale.h>
void        longjmp(jmp_buf jmpb, int retval)                   <setjmp.h>
void        nosound(void)                                       <dos.h>
int         setjmp(jmp_buf jmpb)                                <setjmp.h>
char        *setlocale(int category, char *locale)              <locale.h>
void        sound(unsigned frequency)                           <dos.h>
```

Rutine de control procese

Aceste rutine pornesc şi opresc procese noi dinăuntrul altor procese

```
void    abort(void)                                             <stdlib.h, process.h>
int     execl(char *path, char *arg0, ..., NULL)                <process.h>
int     execle(char *path, char *arg0, ..., NULL, char **env)   <process.h>
int     execlp(char *path, char *arg0, ..)                      <process.h>
int     execlpe(char *path, char *arg0, ..., NULL, char **env)  <process.h>
int     execv(char *path, char *argv[])                         <process.h>
int     execve(char *path, char *argv[],char **env)             <process.h>
int     execvp(char *path, char *argv[])                        <process.h>
int     execvpe(char *path, char *argv[], char **env)           <process.h>
void    _exit(int status)                                       <process.h, stdlib.h>
void    exit(int status)                                        <process.h, stdlib.h>
        getpid                                                  <process.h>
int     raise(int sig)                                          <signal.h>
void    (*signal(int sig, void (*func) int sig[, int subcode]))(int)  <signal.h>
int     spawnl(int mode, char *path, char *arg0, ..., NULL)     <process.h>
int     spawnle(int mode, char *path, char *arg0, ..., NULL,
                char *envp[])                                   <process.h>
int     spawnlp(int mode, char *path, char *arg0, ..., NULL)    <process.h>
int     spawnlpe(int mode, char *path, char *arg0, ..., NULL,
                 char *envp[])                                  <process.h>
int     spawnv(int mode, char *path, char *argv[])              <process.h>
int     spawnve(int mode, char *path, char *argv[],
                char *envp[])                                   <process.h>
int     spawnvp(int mode, char *path, char *argv[])             <process.h>
int     spawnvpe(int mode, char *path, char *argv[], char *envp[]) <process.h>
```

Rutine standard

```
void    abort(void)                                             <stdlib.h>
int     abs(int x)                                              <math.h, stdlib.h>
double  abs(complex x)                                          <complex.h>
int     atexit(atexit_t func)                                   <stdlib.h>
double  atof(const char *s)                                     <math.h, stdlib.h>
int     atoi(const char *s)                                     <stdlib.h>
long    atol(const char *s)                                     <stdlib.h>
void    *bsearch(const void *key, const void *base, size_t *nelem,
                 size_t width, int (*fcmp)(const void*, const void*)) <stdlib.h>
void    *calloc(size_t nitems, size_t size)                     <alloc.h, stdlib.h>
```

Turbo C++ 1.0
Memento - funcţii

```
int      getc(FILE *stream)                                          <stdio.h>
int      getch(void)                                                 <conio.h>
int      getchar(void)                                               <stdio.h>
int      getche(void)                                                <conio.h>
int      getftime(int handle, struct ftime *ftimep)                  <io.h>
char     *getpass(const char *prompt)                                <conio.h>
char     *gets(char *string)                                         <stdio.h>
int      getw(FILE *stream)                                          <stdio.h>
int      ioctl(int handle, int func [, void *argdx, int argcx ])     <io.h>
int      isatty(int handle)                                          <io.h>
int      kbhit(void)                                                 <conio.h>
int      lock(int handle, long offset, long length)                  <io.h>
long     lseek(int handle, long offset, int fromwhere)               <io.h>
int      _open(const char *filename, int oflags)                     <io.h>
int      open(const char *path, int access [ , unsigned mode ])      <io.h>
void     perror(const char *s)                                       <stdio.h>
int      printf(const char *format [, argument, ... ])               <stdio.h>
int      putc(int c, FILE *stream)                                   <stdio.h>
int      putch(int ch)                                               <conio.h>
int      putchar(int c)                                              <stdio.h>
int      puts(const char *s)                                         <stdio.h>
int      putw(int w, FILE *stream)                                   <stdio.h>
int      _read(int handle, void *buf, unsigned len)                  <io.h>
int      read(int handle, void *buf, unsigned len)                   <io.h>
int      remove(const char *filename)                                <stdio.h>
int      rename(const char *oldname, const char *newname)            <stdio.h>
void     rewind(FILE *stream)                                        <stdio.h>
int      scanf(const char *format [,... ])                           <stdio.h>
void     setbuf(FILE *stream, char *buf)                             <stdio.h>
int      setftime(int handle, struct ftime *ftimep)                  <io.h>
int      setmode(int handle, int amode)                              <io.h>
int      setvbuf(FILE *stream, char *buf, int type, size_t size)     <stdio.h>
         sopen(path, access, shflag, mode)                           <io.h>
int      sprintf(char *buffer, const char *format [, argument, ... ])  <stdio.h>
int      sscanf(const char *buffer, const char *format [, address, ... ])  <stdio.h>
int      stat(char *path, struct stat *statbuf)                      <sys\stat.h>
char     *_strerror(const char *s)                                   <string.h, stdio.h>
char     *strerror(int errnum)                                       <stdio.h>
long     tell(int handle)                                            <io.h>
FILE     *tmpfile(void)                                              <stdio.h>
char     *tmpnam(char *sptr)                                         <stdio.h>
int      ungetc(int c, FILE *stream)                                 <stdio.h>
int      ungetch(int ch)                                             <conio.h>
int      unlock(int handle, long offset, long length)                <io.h>
int      vfprintf(FILE *fp, const char *format, va_list arglist)     <stdio.h>
int      vfscanf(FILE *stream, const char *format, va_list arglist)  <stdio.h>
int      vprintf(const char *format, va_list arglist)                <stdio.h>
int      vscanf(const char *format, va_list arglist)                 <stdio.h>
int      vsprintf(char *buffer, const char *format, va_list arglist) <stdio.h>
int      vsscanf(const char *buffer, const char *format, va_list arglist)  <io.h>
int      _write(int handle, void *buf, unsigned len)                 <io.h>
```

Rutine de interfaţare (DOS, BIOS, 8086)

```
int      absread(int drive, int nsects, int lsect, void *buffer)     <dos.h>
int      abswrite(int drive, int nsects, int lsect, void *buffer)    <dos.h>
int      bdos(int dosfun, unsigned dosdx, unsigned dosal)            <dos.h>
int      bdosptr(int dosfun, void *argument, unsigned dosal)         <dos.h>
int      bioscom(int cmd, char abyte, int port)                      <bios.h>
int      biosdisk(int cmd, int drive, int head, int track, int sector,
                  int nsects, void *buffer)                          <bios.h>
int      biosequip(void)                                             <bios.h>
int      bioskey(int cmd)                                            <bios.h>
int      biosmemory(void)                                            <bios.h>
int      biosprint(int cmd, int abyte, int port)                     <bios.h>
long     biostime(int cmd, long newtime)                             <bios.h>
void     ctrlbrk(int (*handler)(void))                               <dos.h>
void     disable(void)                                               <dos.h>
int      doserrerr(struct DOSERROR *eblkp)                           <dos.h>
void     enable(void)                                                <dos.h>
unsigned FP_OFF(void far *p)                                         <dos.h>
unsigned FP_SEG(void far *p)                                         <dos.h>
```


Turbo C++ 1.0
Memento - funcţii


```
char     *ecvt(doublevalue, int ndig, int *dec, int *sign)           <stdlib.h>
void     _exit(int status)                                           <process.h, stdlib.h>
void     exit(int status)                                            <process.h, stdlib.h>
char     *fcvt(double value, int ndig, int *dec, int *sign)          <stdlib.h>
void     free(void *block)                                           <alloc.h, stdlib.h>
char     *gcvt(double value, int ndec, char *buf)                    <stdlib.h>
char     *getenv(const char *name)                                   <stdlib.h>
char     *itoa(int value, char *string, int radix)                   <stdlib.h>
long int labs(long int x)                                            <math.h, stdlib.h>
void     *lfind(const void *key, const void *base, size_t *num,
                size_t width, int (*fcmp(const void *, const void *))  <stdlib.h>
void     *lsearch(const void *key, const void *base, size_t *num,
                size_t width, int (*fcmp(const void *, const void *))  <stdlib.h>
char     *ltoa(longvalue, char *string, int radix)                   <stdlib.h>
void     *malloc(size_t size)                                        <alloc.h, stdlib.h>
int      putenv(cont char *name)                                     <stdlib.h>
void     qsort(void *base, size_t nelem, size_t width,
                int (*fcmp)(const void *, const void *))             <stdlib.h>
int      rand(void)                                                  <stdlib.h>
void     *realloc(void *block, size_t size)                          <stdlib.h>
void     srand(unsigned seed)                                        <stdlib.h>
double   strtod(const char *s, char **endptr)                        <stdlib.h>
long     strtol(const char *s, char **endptr, int radix)             <stdlib.h>
void     swab(char *from, char *to, int nbytes)                      <stdlib.b>
int      system(const char *command)                                 <process.h, stdlib.h>
```

Rutine de afişare text în ferestre

```
void     clreol(void)                                                <conio.h>
void     clrscr(void)                                                <conio.h>
void     delline(void)                                               <conio.h>
int      gettext(int left, int top, int right, int bottom, void *destin)  <conio.h>
void     gettextinfo(struct text_info *r)                            <conio.h>
void     gotoxy(int x, int y)                                        <conio.h>
void     highvideo(void)                                             <conio.h>
void     insline(void)                                               <conio.h>
void     lowvideo(void)                                              <conio.h>
int      movetext(int left, int top, int right, int bottom,
                  int destleft, int desttop)                         <conio.h>
void     normvideo(void)                                             <conio.h>
int      puttext(int left, int top, int right, int bottom, void *source)  <conio.h>
void     textattr(int newattr)                                       <conio.h>
void     textbackground(int newcolor)                                <conio.h>
void     textcolor(int newcolor)                                     <conio.h>
void     textmode(int newmode)                                       <conio.h>
int      wherex(void)                                                <conio.h>
int      wherey(void)                                                <conio.h>
void     window(int left, int top, int right, int bottom)            <conio.h>
```

Rutine pentru dată şi oră

```
char     *asctime(const struct tm *tblock)                           <time.h>
char     *ctime(const time_t *time)                                  <time.h>
double   difftime(time_t time2, time_t time1)                        <time.h>
long     dostounix(struct date *d, struct time *t)                   <dos.h>
void     ftime(struct timeb *buf)                                    <sys\timeb.h>
void     getdate(struct date *datep)                                 <dos.h>
void     gettime(struct time *timep)                                 <dos.h>
struct tm *gmtime(const time_t *timer)                               <time.h>
struct tm *localtime(const time_t *timer)                            <time.h>
time_t   mktime(struct tm *t)                                        <time.h>
void     setdate(struct date *datep)                                 <dos.h>
void     settime(struct time *timep)                                 <dos.h>
int      stime(time_t *tp)                                           <time.h>
size_t   _cdecl strftime(char *s, size_t maxsize, const char *fmt,
                const struct tm *t)                                  <time.h>
time_t   time(time_t *timer)                                         <time.h>
void     tzset(void)                                                 <time.h>
void     unixtodos(long time, struct date *d, struct time *t)        <dos.h>
```

Rutine de lucru pentru liste de argumente variabile

```
type     va_arg(va_list ap, type)                                    <stdarg.h>
void     va_end(va_list ap)                                          <stdarg.h>
void     va_start(va_list ap, lastfix)                               <stdarg.h>
```

DR-DOS 5.0 vs. MS-DOS 5.0

Mai multă memorie, utilitare mai bune - astfel se explică preferinţele utilizatorilor pentru DR-DOS. Cu noua versiune MS-DOS 5.0, Microsoft vrea să recupereze.

De mai mult de un an deja, Digital Research are motive de bucurie: DR-DOS e din ce în ce mai bine primit, concurenţa de la Microsoft pare depăşită: tot mai mulţi vînzători de PC-uri oferă sistemul de operare DR-DOS ca dotare de bază.

Şi este de înţeles de ce MS-DOS 4.01 nu poate concura cu DR-DOS 5.0: utilizarea optimizată a memoriei disponibile, utilitarele foarte bune din cel de-al doilea nu-i lasă practic nici o şansă primului. Pentru a recîştiga o parte din piaţă - sau măcar pentru a nu o pierde de tot, dat fiind că "bătrînul" DOS va determina încă şi în continuare destinele multor PC-uri, Microsoft a "ieşit" cu o versiune nouă, refăcută, MS-DOS 5.0. Ceea ce vă putem relata despre această versiune se referă la versiunea "beta", deci versiunea încă neoficială, lansată pentru testări. Se poate întîmpla deci ca în versiunea finală să mai apară modificări faţă de ceea ce vă prezentăm. Vă vom face cunoscută apariţia versiunii finale de cum vom şti mai mult. (Gurile rele spun că nici nu are sens să porneşti cu 5.0, fiind mai prudent să aştepţi direct 5.01... Vorba rea are oarecare temei: la lansarea versiunii 4.0, au fost inacceptabil de multe erori şi anomalii în sistemul lansat. Chiar dacă ele au fost destul de rapid eliminate prin lansarea versiunii 4.01, imaginea Microsoft a avut mult de suferit. Se încearcă acum o refacere a bunului nume al Microsoft-ului, printr-o fază de testare neaşteptat de lungă - cea mai completă şi mai costisitoare de pînă acum, dacă ar fi să dăm crezare unor declaraţii făcute în acest sens de către reprezentanţi ai Microsoft).

Avantajul major al DR-DOS era pînă acum modul de gestionare al memoriei. Astfel, în DR-DOS, este posibilă "strămutarea" anumitor părţi din sistemul de operare în zona de high-memory. Astfel că dintr-o dată în strîmta zonă a celor 640 kbyte este mai mult loc - utilizatorul are disponibili pentru aplicaţii pînă la 620 kbyte şi chiar mai mult. Un progres imens, avînd în vedere faptul că aplicaţiile moderne sînt dependente de orice bit de memorie operativă. Şi nu numai sistemul de operare poate fi mutat, ci şi drivere, necesare de exemplu pentru utilizarea unui scanner sau a unei cartele fax. Desigur, premisa pentru acest lucru este o memorie extinsă care să corespundă specificaţiilor LIM.

Posibilităţile de încărcare a memoriei superioare în avantajul celei operative sînt aproape nelimitate: drivere, sistemul de operare şi chiar programe rezidente ca "Sidekick" sînt ţinute astfel în memoria principală, fără ca acest lucru să atragă după sine restricţiile uzuale. Niciodată deci, Word 5.0 de exemplu nu mai spune "Insufficient memory ...": doar sînt disponibili peste 600 kbytes de memorie.

Şi totuşi, nu numai acest lucru constituie avantajul DR-DOS. Cei care l-au creat au lucrat cu multă grijă pentru detalii: astfel, s-au născut utilitare care sînt complet compatibile (în jos) cu modelele MS-DOS, dar oferă în plus numeroase posibilităţi şi dispun suplimentar de cîte o funcţie de help proprie (care poate afişa, la cerere, lista parametrilor şi semnificaţiile lor). Ceea ce arată că pînă şi un sistem de operare orientat pe comenzi poate fi "prietenos".

Din furnitură face parte şi o suprafaţă utilizator grafică, care cuprinde aceleaşi funcţii ca cea din MS-DOS 4.01. Deoarece Digital Research dispune oricum de soft comparabil, suprafaţa grafică DOS nu a mai trebuit special făcută: s-a recurs la deja consacratul "GEM", rebotezîndu-l "ViewMax". Acesta nu are toate funcţiile din "GEM", dar în ce priveşte forma de prezentare şi modul de operare nu sînt mari deosebiri. Altfel este la MS-DOS 4.01, unde Microsoft a respectat cu stricteţe prescripţiile SAA ale IBM, creînd o suprafaţă complet nouă (- dar uitînd cîteva funcţii importante pentru ea).

Acest lucru ar urma să se schimbe acum, în MS-DOS 5.0: ca suprafaţă, se zvonea că va fi ceva asemănator cu Windows, (poate chiar o versiune mai veche de Windows), iar în plus, vor fi atîtea utilitare încît producătorii "clasici" de utilitare ca "PC Tools" sau Peter Norton vor fi nevoiţi să-şi schimbe meseria, la apariţia unor programe ca "undelete", "unformat" sau "diskdoctor". Realitatea însă se prezintă mult mai modest: suprafaţa a rămas aceeaşi, cu cîteva inovaţii desigur, iar utilitare noi sînt puţine: "unformat" şi un program de compresie - decompresie. Şi totuşi: chiar dacă nu se vede din prima privire, verificarea capacităţii de memorie disponibile arată ceva mai mult de 630 kbyte! Pînă acum, erau în cel mai bun caz 580 kbyte. Deci aici Microsoft s-a decis să urmeze exemplul concurenţei: se foloseşte memoria suplimentară pentru a descărca memoria operativă de drivere şi părţi ale sistemului de operare.

În afara acestor performanţe îmbunătăţite relative la memoria suplimentară, mare lucru nu s-a schimbat. Nici măcar n-au fost dotate cu funcţii de help diversele utilitare. În continuare lipseşte o protecţie efectivă prin parolă a fişierelor sau directoarelor. În timp ce în DR-DOS datele pot fi protejate la nivelul sistemului de operare, în MS-DOS 5.0 acest lucru este posibil numai în cadrul "shell" - ului, suprafaţa utilizator. La acest lucru se adaugă faptul că parola este înscrisă într-un fişier în care sînt trecuţi şi ceilalţi parametri ai "shell" - ului. La prima vedere, combinaţia de litere este cifrată, dar o privire mai atentă relevă imediat faptul că este vorba de o simplă translaţie în cadrul codului ASCII. Deci nici un fel de piedică pentru profesionişti.

La acest lucru se adaugă faptul că în afara suprafeţei utilizator, protecţia datelor nu funcţionează de loc, datele putînd fi astfel deteriorate sau văzute chiar şi de către începători. La DR-DOS, protecţia este generalizată la nivelul sistemului.

Această listă poate fi cu uşurinţă continuată: cu comanda "type" de exemplu. MS-DOS 5.0 permite afişarea unui singur fişier la un moment dat, pe cînd DR-DOS permite folosirea de specificatori generici (wildcards) (*,?). Pe lîngă aceasta, o "pauză" după fiecare ecran se obţine cu parametrul "/p", pe cînd MS-DOS cere lansarea unei a doua comenzi: "type nume | more". Cine vrea să schimbe configuraţia MS-DOS, trebuie să treacă prin dificultăţile editării fişierelor "autoexec.bat" şi "config.sys". DR-DOS în schimb oferă pentru aceasta un program special de setup, care permite definirea comodă a modului de folosire a memoriei, defineşte cartela grafică în uz sau configurează drivere pentru mouse-ul folosit.

În ce priveşte "depanările", Digital Research a renunţat la un echivalent al "debug"-ului de la Microsoft, scoţind în schimb la lumină o relicvă de pe vremea CP/M-ului: "sid", adică "symbolic instruction debugger", care îndeplineşte în esenţă aceleaşi funcţii ca şi "debug".

Din furnitura de livrare a DR-DOS face parte şi un program cache, care permite creşterea aparentă a vitezei de lucru a harddisk-ului. MS-DOS nu are nimic în acest sens.

Şi un alt program a suferit modificările necesare: sub DR-DOS, cu "format"harddisk-ul nu mai poate fi formatat. Funcţia de formatare e preluată de "fdisk", care în MS-DOS răspunde numai de partiţionarea harddisk-ului.

Avantaj DR-DOS şi la comenzile de ştergere: cu "delq" este posibilă ştergerea selectivă a unor fişiere. Pentru fiecare fişier, programul pune o întrebare suplimentară la consolă pentru a se asigura că fişierul trebuie într-adevăr şters. Iar "xdel" permite chiar ştergerea de directoare şi subdirectoare.

Comparaţia celor două suprafeţe utilizator grafice nu relevă nici un fel de asemănări. Ambele răspund însă la două necesităţi importante: pe de o parte permit administrarea harddisk-ului, iar pe de altă parte permit lansarea fără probleme a unor aplicaţii DOS. Programele trebuiesc configurate înainte de lansare. Multe din ele, în special utilitare, au nevoie de preluarea unor parametri înainte de lansare. Ambele suprafeţe permit astfel de aranjamente; se poate face ca parametri să fie predaţi în mod automat, sau se solicită o "fereastră" înainte de orice lansare. Pur vizual, Viewmax oferă mai mult decît Shell. în timp ce Shell atribuie oricărui program nou primit în listă un mereu acelaşi simbol grafic, Viewmax oferă o varietate de "icons". Astfel, programele pot fi repede clasate pe grupe, folosind de exemplu totdeauna o pană de gîscă pentru programele de prelucrare a textelor. MS-DOS Shell a mers pe altă idee: Diversele programe nu sînt lansate direct din director, ca la Viewmax, ci după configurare sînt trecute într-o listă de programe, care poate fi împărţită apoi în grupe şi subgrupe. Utilizatorul îşi poate astfel alcătui "dosare" proprii pentru fiecare categorie de programe. Astfel, toate programele ce ţin de contabilitate pot fi grupate în "dosarul contabilitate", în timp ce utilitarele pot fi grupate în "tools".

Nu prea sînt dificultăţi cu nici un program, la lansarea lui de la nivelul suprafeţei, deoarece ambii concurenţi "se retrag" aproape total din memoria principală la lansarea unei aplicaţii DOS, revenind abia la terminarea programului.

Ambele sisteme de operare dispun de editoare orientate ecran. Editorul MS-DOS, care acum în sfîrşit ar putea înlocui de mult învechitul editor "edlin", orientat pe rînduri, lucrează chiar cu meniuri moderne tip "pull-down" şi ferestre. Funcţiile disponibile sînt însă limitate la căutări, înlocuiri precum şi unele funcţii pe blocuri de text. Pe cînd ecranul poate fi configurat după gust de fiecare utilizator. Ceea ce nu înseamnă sfîrşitul pentru "edlin" - cine îl doreşte, îl găseşte în continuare în furnitură.

Editorul din DR-DOS nu are astfel de "briz-brizuri". Cine lucrează în mod curent cu Wordstar nu va avea probleme, deoarece cele cîteva funcţii pe care le oferă pot fi lansate numai prin combinaţii de taste. Pentru răsfăţaţii obişnuiţi cu ferestre şi SAA, acest lucru cere acomodare şi învăţare.

În afară de editor, MS-DOS a mai fost dotat cu un program nou. Dacă mult timp interpretorul de Basic GW-Basic se livra împreună cu sistemul de operare, acum în furnitura de livrare a sistemului (deci fără alţi bani) se găseşte sistemul de programare "Quick Basic" în versiunea 1.0. Este practic un mediu de dezvoltare Basic, destul de performant. Păcat că lipseşte opţiunea de compilare: e vorba deci în continuare de un interpretor "pur-sînge". În schimb, are un debugger aproape profesional şi un help online la introducere. Subprograme pot fi găsite rapid cu ajutorul unor ferestre suplimentare, deasemenea funcţii pot fi inserate via ferestre. Astfel încît cine are nevoie de un program rapid, vrea să se iniţieze în programare sau pur şi simplu are nevoie de un program pentru el, are tot ce-i trebuie în Quick Basic, chiar şi în această versiune.

Relativ la compatibilitate, chiar şi versiunea "de probă" MS-DOS s-a dovedit surprinzător de stabilă: n-au fost probleme cu nici una din aplicaţiile testate, de la "Word" şi "dBase" pînă la "Norton Utilities", deşi aceste programe apelează deseori funcţiile critice pentru lucrul cu harddisk-ul. E încă vie în amintirea utilizatorilor dezamăgirea provocată de MS-DOS 4.01, care, datorită modificărilor în administrarea harddisk-ului, făcuse inutilizabile versiunile curente ale unor utilitare ca "PC Tools" sau "Norton Utilities". Se pare că de data aceasta Microsoft a mizat pe un grad mult mai mare de compatibilitate.

(continuare în pag. 34)

OOP - Modă sau ... ?

Limbajele de programare procedurale sînt înlocuite de cele orientate obiect. E acesta "leacul miraculos", soluţia tuturor problemelor ? Sau trebuie ţinut seama de anumite premise, obligatorii ? Şi de ce OOP ?

Puţini sînt termenii ce fac atîta vîlvă în ultima vreme. OOP - conceptul viitorului, soluţia pentru toate problemele pe care le implică limbajele procedurale ? Sau doar furtună într-un pahar cu apă ?

În mod cert, niciuna din poziţii nu este necondiţionat corectă sau incorectă. Programele dezvoltate după metodele clasice au un dezavantaj major: întreţinerea. Studii efectuate în această direcţie au relevat faptul că pentru întreţinerea unui program se cheltuieşte 60 - 80 % din efortul total depus. Şi nu este singurul dezavantaj: programele dezvoltate după metode clasice se modifică extrem de greu: deseori, pentru a rezolva sarcini uşor modificate, programele trebuiesc rescrise. Apoi, şi gradul de refolosire al modulelor este destul de redus - lucru evident prin lipsa de productivitate în soft faţă de ritmul de dezvoltare al hardului. O soluţie pentru toate acestea se speră a fi limbajele de programare orientate obiect.

Şi, într-adevăr, filozofia ce stă la baza OOP - programatorul să se orienteze după date, nu după prelucrările lor - convinge repede. Căci paralela între OOP şi logica gîndirii umane este desigur tentantă:

Omul gîndeşte prin noţiuni, care desemnează obiecte - iar în OOP se folosesc "clase" care se concretizează în obiecte. Comunicarea umană e folosită ca model şi în OOP - obiectele îşi trimit unul altuia "mesaje".

Obiectul - noţiunea de bază a filozofiei OOP - reprezintă o entitate care conţine atît un set de date cît şi descrierea metodelor pentru manipularea lor.

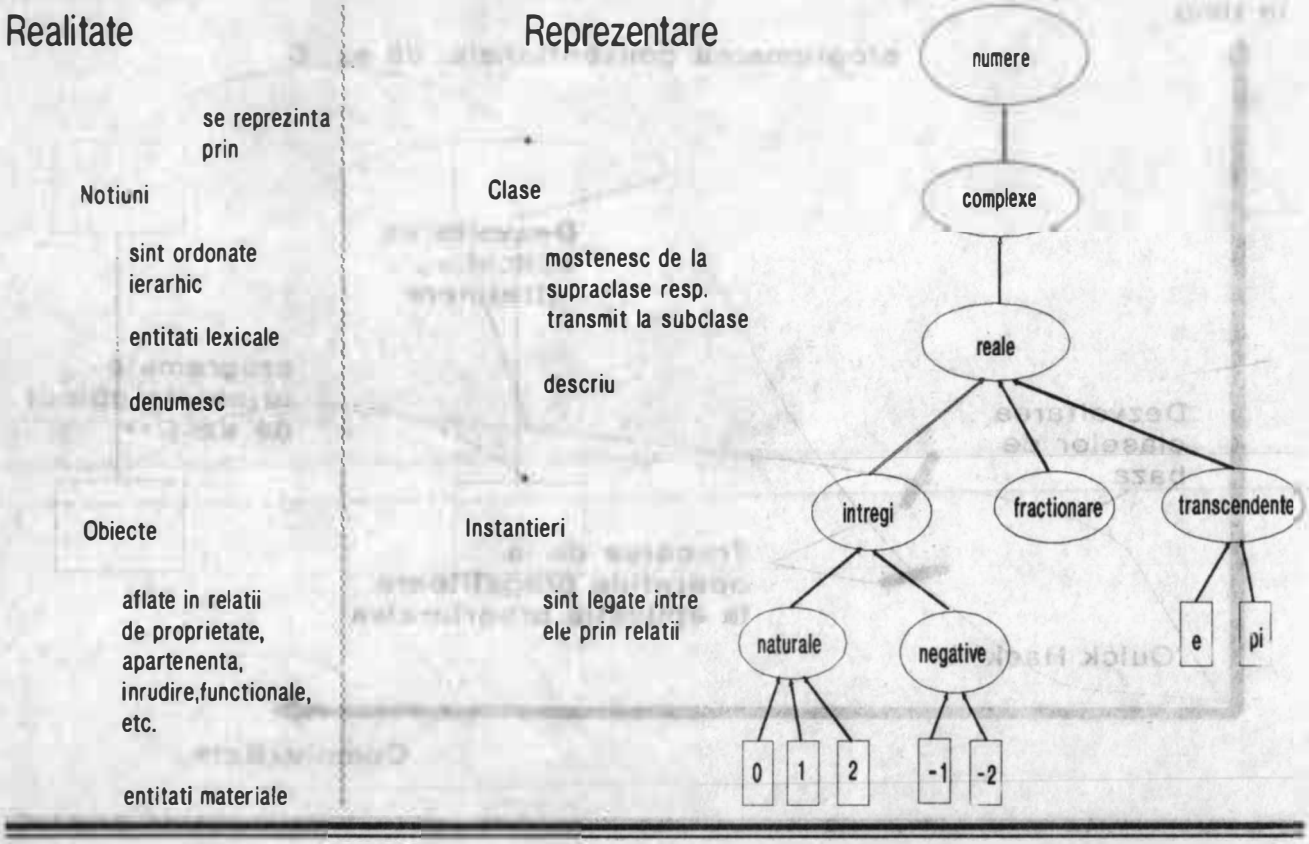
Pentru că obiectele conţin funcţiile necesare manipulării datelor, e suficient să li se adreseze un mesaj corespunzător. Astfel de ex. obiectului "dreptunghi" e suficient să i se trimită mesajul "desenează-te" sau "întinde-te" sau "măreşte-te" - etc. Mesajul nu este deci altceva decît o specificare a unei prelucrări asupra obiectului, care trebuie să precizeze:

-obiectul căruia i se adresează

-operaţia pe care obiectul trebuie să o execute

-parametri actuali necesari operaţiei

Obiectul reacţionează univoc la mesaj; despre polimorfism se vorbeşte cînd obiecte diferite reacţionează în mod diferit la un acelaşi



mesaj. Obiectele sînt descrise în "clase" - este ceea ce gîndirea umană face prin categorisire. Metoda e simplă - proprietăţile comune mai multor obiecte se sintetizează într-o clasă nouă, care conţine toate caracteristicile abstracte ale claselor subordonate. Astfel se constituie supraclasele. Astfel de ex. clasele "autoturism" şi camion ar fi reunite în clasa de bază "maşină".

În afară de aceasta, există instanţierile - obiectele descrise de o anumită clasă. Clasa ar putea să fie "întreg natural" - instanţierile ar fi 1, 2, 3, ...

Avantajele OOP - independenţa reprezentării de implementare, încapsularea, uşurinţa operării modificărilor (de ex. a structurii datelor) etc. - se apropie astfel din ce în ce mai mult de "cipul" soft. Conform acestui concept, munca redundantă (repetată) dispare odată pentru totdeauna, astfel încît rămîn de făcut numai lucrurile realmente noi - ceea ce înseamnă că activitatea programatorului ar consta în a construi aplicaţia do ită din obiecte existente. Ceea ce înseamnă într-adevăr o revoluţie pentru programarea clasică.

Procedeul pare foarte simplu - şi aşa şi este. Totuşi. cîteva dezavantaje majore ale OOP nu pot fi trecute pur şi simplu cu vederea.

Programele OOP nu sînt capabile de multitasking; utilizarea de cod uzual este dificilă; pentru aplicaţii în timp real, există probleme serioase legate de eficienţa codului; programele consumă multă memorie, iar pentru cei implicaţi în proiectare e necesar un efort semnificativ de şcolarizare şi/sau acomodare .

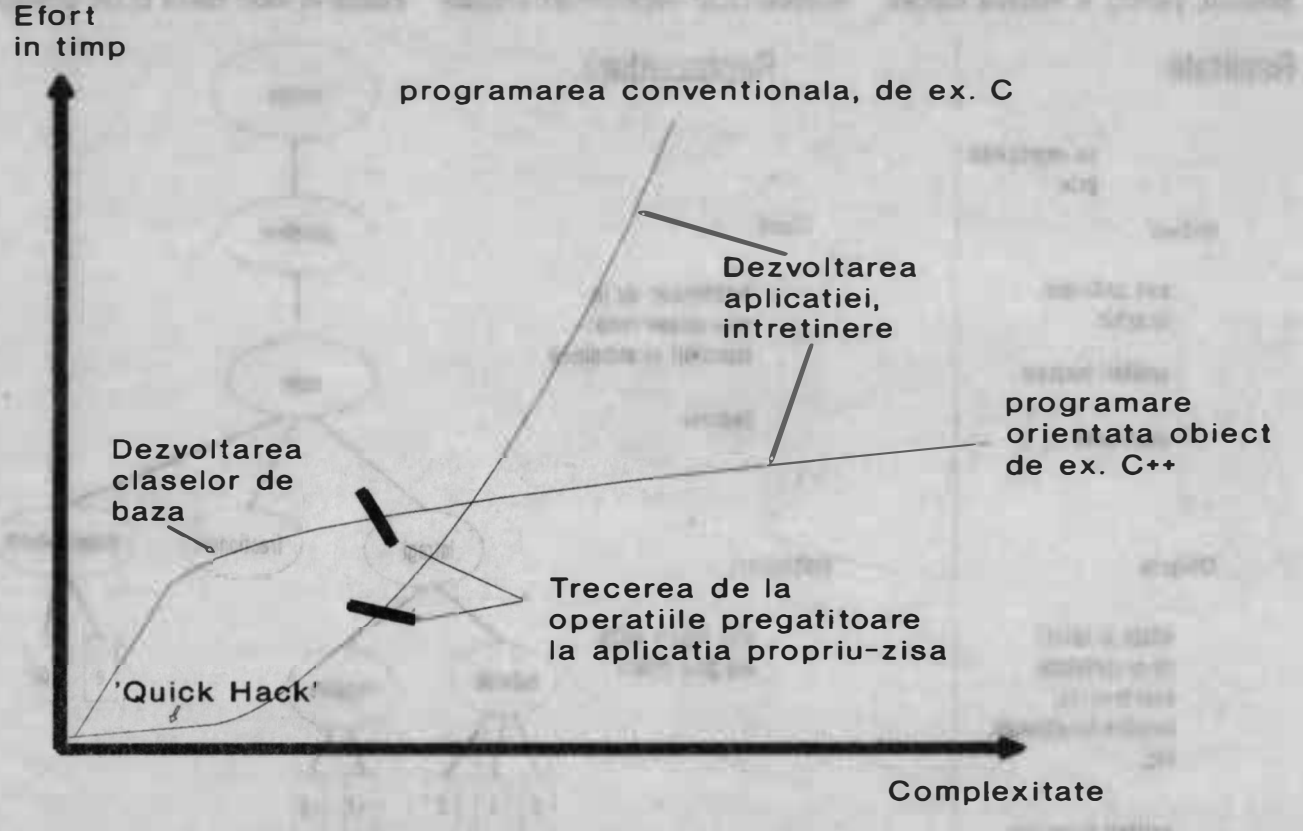
Dezavantajul major este însă în mod evident numai temporar: există încă puţine "cutii de scule" (toolboxes) şi clase gata definite, ceea ce înseamnă multă muncă în faza de proiectare.

Dacă însă clasele de bază ("cipurile soft") sînt definite, se pot într-adevăr selecta componentele potrivite pentru a construi aplicaţia dorită. Citîndu-l pe Georg Heeg: "oricum ar fi, este o mare diferenţă între a învăţa atît la elefant cît şi la pisică sau girafă că gîtul este format din 7 vertebre şi a învăţa acest lucru o singură dată, la nivelul "mamifere".

Pentru că, acum, principala muncă o constituie definirea claselor de bază. Se speră ca în viitor pruductivitatea faţă de programarea clasică să crească între 10 şi 100 de ori. E de ţinut seama de acest lucru; mai ales pentru echipe de proiectare mai mari, investiţia în viitor merită.

Experţii consideră că principalele domenii în care se vor utiliza OOP vor fi suprafeţele utilizator grafice, aplicaţiile tehnice şi sistemele expert, mai puţin bazele de date, procesele numerice şi aplicaţiile în timp real.

(Iosif Fettich)



"C" - Cuceritorul

Ce anume determină atracţia crescîndă exercitată de acest limbaj de programare ? De ce tocmai "C" ?

Cine nu a văzut un calculator ? PC sau mini, pe masă sau în birou - toate au ceva în comun: nu produc nimic dacă nu sînt programate. Ceea ce , desigur, nu vă deranjează, dacă achiziţionaţi o aplicaţie gata scrisă (programată) şi o folosiţi ca atare.

Dar dacă vă prinde microbul ? Vreţi să ştiţi cum anume lucrează calculatorul, aveţi idei noi pentru aplicaţia cu care lucraţi, vreţi să vă scrieţi singur alta - pe scurt, vreţi să programaţi. Primele semne de întrebare: în ce limbă să comunicaţi cu calculatorul ? Primele cercetări vă abandonează într-o adevărată junglă de denumiri: Cobol, Fortran, Algol, Pascal, Assembler, Modula, Lisp, PL/I, C, Ada etc. Ce să alegeţi - şi de ce ?

Vi se recomandă Basic, ca fiind extrem de uşor de învăţat. Încercaţi - şi într-adevăr, primele succese nu se lasă aşteptate. Repede aveţi o serie întreagă de programe care fac ceea ce v-aţi propus.

Pretenţiile vă cresc acum, scopurile pe care vi le-aţi propus devin mai complexe, începeţi să nu aveţi răbdare cînd calculatorul "gîndeşte" prea mult. Entuziasmul iniţial pentru Basic scade pe zi ce trece: constataţi că programele mai complicate devin imposibil de descifrat la cîteva luni după ce leaţi terminat - şi doar era totul aşa de clar cînd aţi scris rutina asta ! - că unele rutine durează insuportabil de mult, că programul care merge impecabil pe calculatorul D-voastră refuză să facă orice altceva decît să blocheze calculatorul prietenului Dvs., căruia vroiaţi să-i faceţi o demonstraţie - ş.a.m.d. Dezamăgit, căutaţi un limbaj de programare mai bun - de data aceasta un compilator, pentru a obţine viteze de execuţie convenabile.

Cu 3 cerinţe mai clar formulate acum (structurare, viteză de execuţie, portabilitate) intraţi din nou în "pădurea" limbajelor de programare. Sub marcajul "structurat" găsiţi o cărare care vă duce pe lîngă Pascal, Ada, Oberon, Modula-2 şi C. Insistînd asupra vitezei de execuţie a codului produs, rămîn doar Modula-2, Assembler şi C. Iar dacă ţineţi ca programele Dvs. să fie şi portabile - i.e., să poată fi rulate fără modificări majore pe orice calculator, atunci căutarea s-a terminat. Tot ce v-a rămas în plasă este "C"-ul.

Gîndit iniţial pentru calculatoare mari, "C" s-a născut în 1972. Între timp, s-a răspîndit atît de mult încît practic nu are concurent şi aproape orice tip de calculator are (cel puţin) un compilator de C disponibil.

Mulţi din producătorii de soft comercial - destinat vînzării - nu mai folosesc alte limbaje de programare, pentru că astfel nu numai că viteza de execuţie a programelor creşte, dar şi din punct de vedere comercial apar avantaje: la transpunerea pe un alt tip de calculator, un program C poate fi transpus unu la unu, adică fără modificări. Trebuiesc eventual adaptate numai caracteristicile speciale - grafica, intrări/ieşiri. Astfel încît nu este de mirare că un produs-program apare pe piaţă simultan pentru mai multe tipuri de calculatoare. În afară de aceasta, compilatoarele moderne de C generează un cod atît de bun şi de eficient încît în foarte rare cazuri este necesară intervenţia la nivelul limbajuluio de asamblare; pentru a "grăbi" vreo rutină mai înceată.

Entuziasmul peren cu care a fost primit "C"-ul a determinat Institutul American pentru Standardizare (ANSI - American National Standard Institute) să încerce o standardizare a limbajului, pentru a spori portabilitatea şi eficienţa lui. Standardul - terminat în 1988 - se referă atît la caracteristici ce ţin de tehnica programării cît şi la caracteristici sintactice. Cel mai important dintre avantajele folosirii lui - e "standard", deci nu există probleme de portabilitate; cel mai important dintre dezavantaje: anumite "trucuri", care produc cod eficient sau utilizează foarte rafinat anumite structuri specifice unui sistem de calcul, nu sînt permise.

Este remarcabilă "libertatea de exprimare" lăsată programatorului în "C". Rămîne la latitudinea lui să-şi scrie programele structurat, clar, sau să scrie programe supereficiente, dar ilizibile. Deoarece există multe programe şi din a doua categorie, criticii spun că "C" nu este un limbaj de programare de nivel înalt. Ceea ce, desigur, poate fi adevărat dacă programatorul este "nedisciplinat". Dar se poate şi altfel - şi programatorul e liber să-şi urmeze propriile opţiuni. Flexibilitatea deosebită a "C"-ului se datorează şi simplităţii nucleului limbajului. Pentru cine are noţiuni de programare, învăţarea "C"-ului nu ridică probleme - instrucţiunile şi cuvintele cheie sînt foarte puţine. Chiar atît de puţine încît foarte repede poate să apară întrebarea: "bine, bine, dar cum să programez ceva într-un limbaj care nu are nici măcar nişte instrucţiuni de intrare/ieşire ?"

Secretul îl constituie filozofia "C"-ului - toate funcţiile sînt externe limbajului şi sînt accesibile prin biblioteci. Cele mai des folosite funcţii (apeluri sistem) sînt adunate într-o bibliotecă standard care se ataşează tuturor compilatoarelor C. Ea conţine rutine pentru intrări/ieşiri generale, pentru gestionarea memoriei şi tratarea şirurilor. Conţinutul bibliotecii standard e prescris cu stricteţe de standardul ANSI, astfel încît orice compilator "C" care corespunde standardului dispune de funcţii identice (cel puţin ca dotare de bază).

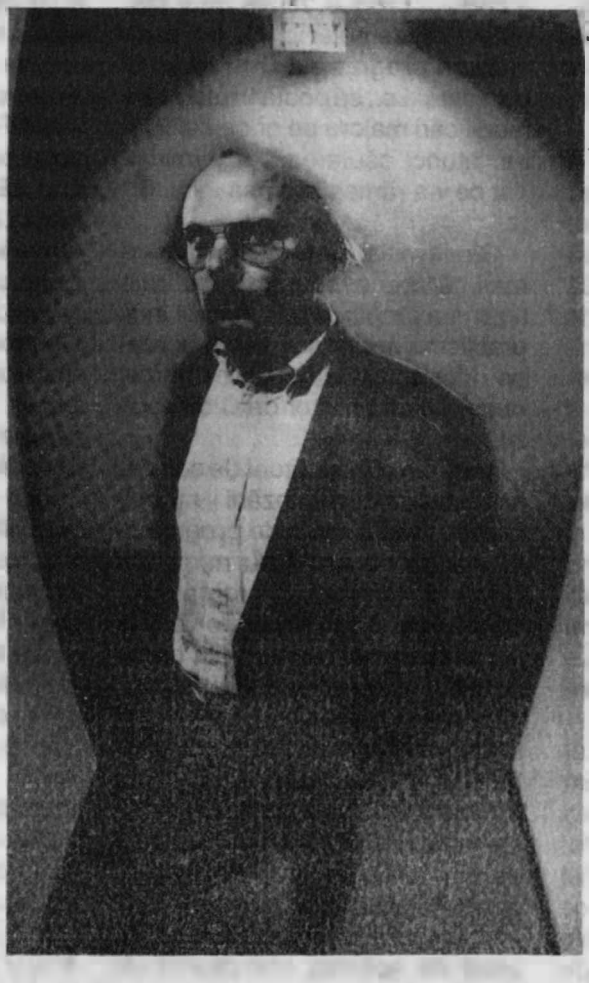
Principiul bibliotecilor de funcţii este foarte avantajos pentru "traducerea" unui program în cod exe-

cutabil (compilarea), deoarece compilatorul nu este încărcat cu numeroase cuvinte-cheie ale limbajului. Operaţiunea de compilare se desfăşoară de aceea semnificativ mai repede la un program C decît la limbajele de programare comparabile. Compilatorul de Turbo Pascal de ex. trebuie să administreze circa 200 de comenzi şi proceduri standard, pe cînd un compilator de C "are de lucru" numai cu 32 cuvinte-cheie şi cîţiva operatori.

Pentru "compunerea" unui program executabil din mai multe module compilate este necesar un alt utilitar: link-editorul. Sigur că şi el are mult de lucru (astfel că o parte din timpul "cîştigat" de C la compilare este pierdut în această fază). Link-editorul, în conformitate cu informaţiile şi tabelele stocate în modulele obiect compilate, caută funcţiile necesare în bibliotecile corespunzătoare şi le adună într-un "tot", un program executabil, de regulă mult mai compact decît la alte compilatoare).

Colecţiile externe de funcţii mai au un avantaj: deoarece o astfel de bibliotecă constă doar dintr-o înşiruire de funcţii C compilate - bineînţeles că şi funcţiile bibliotecii standard sînt formulate în C! - este foarte uşor a extinde acest set de funcţii cu funcţii noi sau a înlocui unele funcţii cu altele mai bune. Astfel că tot "vocabularul" unui sistem C poate fi cu uşurinţă adaptat propriilor nevoi sau dorinţe.

În afară de aceasta, prin "scoaterea" funcţiilordin limbaj, dependenţa de maşină a compilatorului dispare aproape complet, deoarece această dependenţă rămîne limitată la cîteva funcţii de bază. În conformitate cu filozofia Unix, după care orice operaţiune de intrare/ieşire (fie ea la ecran, imprimantă, tastatură sau altceva) este echivalată unei citiri/scrieri într-un fişier, aceste funcţii de bază constau de regulă din cîteva mici rutine pentru a asigura intări/ieşiri la / de la diverse echipamente periferice. Astfel încît la portarea unui compilator C pe un alt calculator nu trebuiesc adaptate decît aceste rutinuţe respectivului sistem de calcul. Prin structura deschisă a "sistemului C", nu numai programele scrise în C sînt uşor portabile, ci şi sistemul în ansamblu poate fi foarte uşor transpus pe alte calculatoare.



Bjarne Stroustrup, "tatăl" C++ - ului

Prin foarte multe din caracteristicile sale, "C"-ul este cel mai "de viitor" limbaj pentru toate tipurile de calculatoare şi, fără nici un dubiu, cea mai bună alegere dacă vreţi să învăţaţi un nou limbaj de programare.

Viitorul, pentru "C", a început deja: criticii au acuzat "C"-ul că lucrează mai ales cu numere şi valori, dar nu sprijină direct prelucrarea obiectelor. Orientarea obiect - care deja este implementată în Pascal şi în Oberon - va duce la o programare completamente abstractă şi independentă de structurile concrete din calculator. Diverse limbaje speciale (mai ales pentru inteligenţă artificială - Prolog, Lisp etc.) urmăresc de mai multă vreme acest obiectiv.

Îmbinarea avantajelor structurale ale "C"-ului cu OOP au născut C++, care se răspîndeşte într-un ritm incredibil. Rămîne de văzut în cît timp C++ va reuşi să detroneze "C" - liderul indubitabil al limbajelor de programare folosite în prezent.

(Iosif Fettich)

DR-DOS 5.0 vs. MS-DOS 5.0

(continuare din pag. 30)

În încheiere, se pune desigur întrebarea care DOS să rămînă pe viitor stăpîn pe valorile depozitate pe harddisk? Privind dinspre confortul utilizatorului, DR-DOS e clar cîştigător. Dar cine nu vrea să renunţe la creaţiile unei din cele mai mari case de soft din lume şi vrea să rămînă într-adevăr 100% compatibil, va lucra şi pe viitor cu MS-DOS. Decizia rămîne de fapt mai degrabă a producătorilor de hardware, dat fiind că ambele sisteme de operare se comercializează numai în domeniul OEM (original equipment manufacturer).

(I.F.)

Dosar: Turbo C++ 1.0

Unul din multele produse foarte reuşite ale firmei Borland, noul mediu de dezvoltare se prezintă cu multe noutăţi. Cele mai importante dintre ele vi le prezentăm în continuare.

În primul rînd, desigur, ++. Corespunzînd în mare măsură "standardului" (neoficial) pentru compilatoarele C++ - este vorba de norma AT&T 2.0 (dar totodată celui mai nou standard ANSI pentru C, cel din decembrie 1988) - Turbo C++ permite programarea cu obiecte "ca la carte". Relativ uşoară este trecerea de la C la C++ nu numai pentru că C++ este un supraset adevărat al C-ului, dar şi pentru că propunerile AT&T au fost preluate între timp de mai toate compilatoarele existente pe piaţă şi disponibilizate în biblioteci adiţionale.

Miezul C++-ului îl constituie noţiunea de "clasă". Cele trei aspecte fundamentale ale programării orientate obiect - încapsularea, moştenirea şi polimorfismul - sînt realizabile, de altfel, şi cu mijloace clasice, dar numai cu preţul unui efort deosebit şi într-un mod destul de "fragil" - e deajuns ca un programator să nu respecte convenţiile stabilite, ca tot "OOP"-ul să nu fie decît o frumoasă faţadă. Nu aşa se întîmplă în C++. Definiţiile de clase permit subdivizări consecvente ale funcţiilor şi datelor, care nu pot fi modificate decît de funcţii interne claselor. Ceea ce duce nu numai la o modularitate mai ridicată, dar uşurează foarte mult depanarea, deoarece erorile de programare care apar pot fi repede urmărite pînă în clasa în care se manifestă.

"Ereditatea" ca principiu fundamental al OOP Înseamnă că din clasele existente se pot genera alte clase, care moştenesc în mod automat toate domeniile de date şi funcţiile clasei vechi, care în mod automat devine parte a clasei noi. Şi în definirea convenţională de structuri se pot concepe construcţii ierarhizate de tipuri de date; în OOP însă, încapsularea se păstrează, mai mult, funcţiile îşi păstrează valabilitatea şi permit în continuare prelucrarea datelor clasei. În C-ul convenţional, ar fi necesare o serie întreagă de "acrobaţii" lingvistice pentru a putea prelucra cu o singură funcţie toate tipurile derivate de date.

Ca o consecinţă directă a "moştenirii" apare polimorfismul funcţiilor. "Polimorfism" vine din greacă şi înseamnă "multe forme": este proprietatea de a da unei funcţii un nume care va fi folosit uniform în toată ierarhia de obiecte, fiecare obiect fiind însă liber să-şi implementeze funcţia în modul cel mai adecvat. Astfel, "moştenirea" duce la preluarea numelui funcţiei, nu neapărat însă şi la cea a implementării. La nevoie, aceasta poate fi refăcută şi duce atunci la înlocuirea codului moştenit. Pe de altă parte, se poate păstra şi vechea versiune a funcţiei. Compilatorul de C++ rezolvă aceste referinţe complexe la momentul execuţiei, parcurgînd ierarhia de obiecte de la obiectul actual şi pînă (cel mult) la rădăcină, pentru a găsi o implementare a funcţiei necesitate. Cel tîrziu la rădăcină funcţia este găsită - altfel, ea nu ar fi putut fi moştenită.

Dar nu toate inovaţiile aduse de C++ sînt legate de programarea orientată obiect. Şi domeniul convenţional este restucturat. Astfel de exemplu biblioteca standard stdio.h este înlocuită cu stream.h. Comentarii de un singur rînd pot fi marcate prin "//". Parametri funcţiilor pot fi transmişi după dorinţă ca valoare sau prin adresă. Etc.

Noutăţi la Turbo-C++ 1.0

- C++ conform specificaţiilor AT&T 2.0
- aritmetică BCD şi numere complexe
- overlay-uri în tehnică VROOM
- mediu de dezvoltare îmbunătăţit
- TEML - macrolimbaj pentru configurarea editorului
- suprafaţă de utilizare operabilă cu mouse
- debugger mult îmbunătăţit
- profiler

Condiţionat de lucrul în C++, Turbo-C introduce în sfîrşit şi calculele cu numere complexe şi aritmetica BCD (cu o precizie de 17 cifre semnificative, necesară în calcule financiar-contabile).

Noutatea care "sare în ochi", pentru cei familiarizaţi cu versiunile mai vechi ale compilatorului, este mediul de dezvoltare, numit IDE (Integrated Development Environment). Noua suprafaţă corespunde standardului SAA şi poate fi folosit (în sfîrşit) şi prin intermediul mouse-ului. Dar cine nu are mouse sau preferă să lucreze cu tastatura, poate să o facă la fel de comod. În afara unor "jucării" atît de plăcute ca ferestrele cu umbre, IDE - ul a suferit numeroase îmbunătăţiri interne. Astfel, pot fi deschise mai multe ferestre simultan, mărimea lor se poate defini, poziţia lor pe ecran se poate fixa după dorinţă. Texte marcate pot fi copiate dintr-o fereastră într-alta prin intermediul clipboard-ului.

Editorul incorporat aduce şi el cîteva noutăţi: se pot edita acum şi surse mai mari de 64 kbyte, sau se pot edita deodată (în paralel) mai multe surse distincte; singura restricţie: dimensiunea totală a surselor editate să nu depăşească 8 Mbyte.

O altă noutate o constituie "TEML" (Turbo Editor Macro Language), care permite reconfigurarea sau modificarea editorului şi a tastelor sau combinaţiilor de taste asociate anumitor comenzi. Adică: dacă în locul editorului standard preferaţi să lucraţi cu un editor ca "Emacs" sau "Vi", prin intermediul macrocomenzilor se poate modifica editorul în aşa fel încît acesta să se prezinte ca cel cu care sînteţi obişnuit. Destul de puternic, acest macrolimbaj permite asocierea unor mici "programe de editare" la anumite taste sau combinaţii de taste, fără a ajunge totuşi la ca-

litățile unui editor specializat (nu există variabile de ex.) Există remediu și pentru acest lucru: editorul cu care lucrați de regulă poate fi înglobat în mediul Turbo, astfel încît să lucreze în ferestrele normale cu care operează acesta, păstrînd în felul acesta modul consecvent în care mediul Turbo-C apare utilizatorului.

Prin intermediul comenzii "asm", se poate insera cod în asamblor, fără a părăsi pentru aceasta mediul C. Pentru aceasta, este necesară însă prezența unui asamblor extern (ca "tasm" sau "masm"). Inserarea de cod în limbaj de asamblare este mult ușurată în felul acesta, deasemenea se simplifică mult predarea parametrilor între funcțiile C și cele scrise în limbaj de asamblare.

Și link-editorul pentru overlay-uri, care disponibilizează tehnica VROOM (Virtuell Runtime Object Oriented Memory Management) în C++, este nou. Altfel decît la tehnicile uzuale, care "descarcă" modulele nefolosite, lăsînd însă programatorului responsabilitățile legate de interdependența modulelor, VROOM-ul Borland-ului folosește un "dynamic segment swapping" care folosește segmentele ca cele mai mici unități ce se depun pe harddisk, memoria expandată sau în memoria extinsă pentru a face loc în memoria operativă altui fragment de program, care nu a încăput. Cu foarte puține excepții (rutinele de tratare întrerupere de ex.) administrarea dependențelor este asigurată de "managerul de overlay-uri". Programatorul e degrevat.

Deasemenea o noutate este "Profiler"-ul, cuprins pentru prima dată într-o furnitură Borland. Profiler-ul este un soi de cronometror-numărător soft, care poate urmări gestiunea resurselor critice ale calculatorului. Se pot cronometra anumite secvențe de program, se pot număra accesele la disc făcute în anumite secvențe de program sau pe parcursul anumitor rutine, se pot urmări și trage concluzii asupra activității tastaturii, imprimantei, întreruperilor. Astfel, punctele critice ale unui program pot fi mult mai ușor localizate. În același scop poate fi utilă statistica apelurilor, care pentru fiecare rutină sau linie de program contorizează de cîte ori a fost parcursă pe timpul execuției. Informații desigur utile oferă și lista de referințe încrucișate a funcțiilor (care funcție de unde este apelată și ce funcții apelează). Acest "profiling" nu trebuie confundat cu optimizarea internă a compilatorului. Compilatorul poate doar să înlocuiască unele secvențe de cod cu altele, mai eficiente, dar care produc exact același efect. Optimizarea algoritmilor folosiți rămîne în continuare sarcina programatorului, care are acum - prin Profiler - un instrument eficace pentru detectarea părților mai slabe ale unui program.

Multe îmbunătățiri apar și în Turbo-Debugger, prezent acum în versiunea 2.0. În afară de faptul că poate fi utilizat cu mouse, s-a îmbunătățit considerabil modul de administrare al memoriei - debugger-ul nu mai folosește în memoria principală decît 70 kbyte. Tot restul este "depozitat" în memoria extinsă sau cea expandată. Se remarcă în mod deosebit faptul că și debugger-ul este adaptat la OOP, permițînd lucrul cu clase și păstrînd în felul acesta "rotunjimea" acestui mediu C++.

Inovația care în mod cert va cîștiga pe toți cei care s-au luptat cu așa ceva este legată de TSR-uri și drivere. Depanarea acestora este - de regulă - o mare problemă; ce ține de trecut, dacă folosiți noul debugger Borland! Acesta se transformă el însuși într-un TSR, artificiu prin care depanarea de programe rezidente - altfel, "imposibilă", devine relativ confortabilă, ușor de făcut. După ce te-ai obișnuit ca din debugger-ul lansat să instalezi un program rezident (lucru care, după cum bine se știe, nu ar trebui făcut niciodată ! ...) sesiunea de depanare decurge relativ normal:

1. se încarcă Turbo-Debugger-ul
2. se instalează programul TSR de depanat
3. se fixează un breakpoint pe un punct de intrare în TSR
4. se face rezident și Turbo-Debugger-ul
5. se activează TSR-ul de depanat la nivel DOS, folosind tasta sau combinația de taste corespunzătoare

Odată activat TSR-ul, acesta se derulează normal, pînă cînd "dă de breakpoint". În acest moment debugger-ul intră pe fir și depanarea se face ca la un program obișnuit.

Această metodă neconvențională, care funcționează

Comparație: Turbo-C++ - MS-C 6.0

Caracteristica	MS-C 6.0	Turbo-C++ 1.0
orientare obiect	nu	da
optimizări relative la UC cu		
8086/8088	da	da
80286	da	da
80386	da	da
80486	nu	nu
optimizări relative la sistemul de operare		
DOS	da	nu
OS/2	da	nu
Windows	da	nu
optimizări relative la codul produs		
ptr. viteză la execuție	da	da
dimensiunea codului	da	da
optimizarea buclelor	da	da
eliminare de cod "mort"	da	da
eliminare expresii parțiale	da	da
răspîndirea constantelor	da	da
biblioteci		
toate modelele de memorie	da	da
80x87	da	da
OS/2	da	nu
Windows	da	nu
debugger		
integrat	nu	da
stand-alone	da	da
profiler		
integrat	nu	nu
stand-alone	nu	da
source-browser		
integrat	nu	nu
stand-alone	da	nu
grafica		
bibliotecă	da	da
drivere	nu	da
CGA	da	da
Hercules	da	da
EGA/VGA	da	da
Super VGA	nu	da*

*Driverele pentru SuperVGA trebuiesc achiziționate separat

asemănător şi în cazul driver-elor, face din Turbo-Debugger cel mai bun depanator soft disponibil la ora actuală. Nici chiar CodeView-ul Microsoft-ului, în versiunea 3.0, livrat împreună cu Microsoft 6.0, nu ajunge la astfel de performanţe.

Şi încă nu e totul - debugger-ul Borland-ului este în stare să execute pas cu pas un program, **înapoi** !! Te prinde mirarea că astfel de metode nu au fost realizate mai de mult.

Cu toate acestea, nu trebuie uitat că un compilator trebuie, înainte de toate, să producă cod. Şi şi aici, Turbo-C++ nu are motive să se ascundă: chiar dacă C++ produce un overhead destul de substanţial, codul produs de Turbo-C++ încă este semnificativ mai rapid decît cel produs de Quick-C 2.0, fiind destul de apropiat de MS-C 5.1. Calitatea codului produs de MS 6.0 oricum însă nu este atinsă. În ce priveşte optimizările de cod la momentul execuţiei, compilatorul Microsoft este, probabil, imbatabil la ora actuală.

Unul din elementele de strălucire ale oricărui compilator Borland a fost şi rămîne biblioteca grafică. BGI - "Borland Graphic Interface" - lucrează cu drivere ce pot conlucra cu aplicaţii realizate de alte firme, conferind Turbo-C++ - ului o portabilitate fără egal pe sistemele grafice DOS. Cine vrea, poate lucra sub Turbo-C++, fără probleme, cu Super VGA de 1024 x 768 puncte a 16 sau chiar 256 de culori.

Nu trebuie uitat help-ul online oferit de Turbo C++, un ajutor realmente de nepreţuit. Pe o structură tip hipertext, help-ul permite o găsire foarte rapidă a informaţiilor necesitate. Şi nu numai atît: exemplele de utilizare a funcţiilor pot fi copiate din help în programele utilizator, prin intermediul clipboard-ului, apăsînd doar cîteva taste - sau făcind două - trei mişcări şi clic-uri cu mouse-ul.

Mai trebuie amintite şi documentaţiile de utilizare - foarte bune, utilizabile atît de către începători cît şi de către profesionişti.

Ce-i lipseşte Turbo-C-ului pentru a fi totuşi "number one"? Bibliotecile pentru OS/2 şi Windows. Pentru programatorul profesionist care lucrează la aplicaţii ce vor trebui să meargă şi sub OS/2, şi sub Windows, Turbo-C-ul nu este o soluţie. Încă: pentru că deja circulă o versiune de test C++ pentru Windows 3.0. Vremuri grele pentru Microsoft? MS-SDK - pachetul necesar dezvoltării aplicaţiilor sub Windows - devine superfluu; va fi folosit în schimb Whitewater Group Toolkit, care merge numai sub Windows.

La noi, Turbo C++ poate fi achiziţionat de exemplu de la firma Logic, (Bucureşti, 90/75.49.00 sau 75.71.35; Sibiu, 924/4.66.52 sau 4.54.75). Preţurile la zi trebuiesc aflate de acolo; în vest, ele sînt (erau) cam la 450 DM pentru versiunea "normală" (numai compilatorul) şi 690 mărci pentru versiunea "profesională" (compilator + asamblor, debugger, profiler).

Ultima oră: şi Borland C++ 2.0, ("the only complete C and C++ programming environment for DOS and Windows") este disponibil deja; printre alţii, şi la LOGIC, la preţul de 143.055 lei vom mai reveni.

(ing. Iosif Fettich)

Teste de viteză pentru "large model"

	MS-C 6.0	MS-C 5.1	Quick-C 2.0	Turbo-C 2.0	Turbo-C++ 1.0
Array1	30.6	36.0	43.9	51.0	51.0
Array 2	3.0	4.1	38.0	37.4	32.1
Fibonacci	36.5	43.0	53.2	45.7	45.6
Float	36.0	43.0	54.4	58.0	58.0
Integer	0.0	1.9	48.1	48.6	48.6
Loop	0.0	0.0	27.6	27.5	22.1
Sieve 1	20.5	22.5	25.5	25.0	24.2
Sieve 2	16.8	23.0	23.9	23.6	23.5
Whetstone	7.0	6.5	9.0	7.5	7.5

Programe de test de la Microsoft, măsurate pe un AT la 12 MHz

Concurenţii dintr-o privire

Produsul	Turbo-C++ 1.0	MS-C 6.0	Topspeed C
Producător	Borland	Microsoft	Jensen & Co
orientare obiect	da	nu	da
bibliotecă Windows	nu	da+SDK	da+SDK
bibliotecă OS/2	nu	da+SDK	da+SDK
editor	da	da	da
debugger	da	da	da
mouse	da	da	da
linkeditor ptr. overlay-uri	da	da	nu
profiler	în versiunea profesională	nu	în versiunea profesională
preţul (în mărci)	456 684 ptr. versiunea profesională	1585	489 - DOS, 1198, versiunea profesională, 1498 - OS/2

Paradigma Paradox

Înainte să programaţi în PAL (Paradox Application Language), trebuie să înţelegeţi modelul Paradox.

Există vreun utilizator de Paradox pe acolo, pe la Dvs. ? Sau vreun utilizator de Xbase care a auzit o mulţime de lucruri bune despre Paradox şi s-a gîndit să-l exploreze mai îndeaproape ?

Acest articol se adresează ambelor grupuri de utilizatori. Paradox are o creştere considerabilă în acest moment, în special în rîndul corporaţiilor. Neaşteptat Paradox este SGBD-ul (Sistemul de Gestiune a Bazelor de Date) relaţional, pentru PC-uri, cel mai vîndut în rîndul corporaţiilor importante. Un institut de cercetări afirmă că Paradox a acaparat mai mult de 44% din această piaţă, în timp ce dBase III împreună cu dBase IV ating doar 35%, iar Fox şi R:Base au sub 10%. (Datele se referă la S.U.A.)

Borland însăşi pare să aibă un impuls datorat opţiunii client-server SQL (Structured Query Language), unei noi versiuni Paradox Engine, şi posibilităţilor de cuplare cu Quattro Pro 2.0, programul propriu de calcul tabelar.

Dar Paradox a rămas um mister pentru o mulţime de oameni. Mulţi programatori şi utilizatori ai altor SGBD-uri nu înţeleg cu claritate stilul şi modelul Paradox. Paradox are nemeritata reputaţie că ar fi o unealtă de calibru redus recomandabilă doar utilizatorilor finali. De fapt, este un mediu de dezvoltare complet, cu posibilitatea de a genera machete (forme) şi rapoarte, cu un QbE (Query by Example - interogare prin exemplu) pe care toţi doresc să-l copieze şi cu un limbaj puternic care permite proiectanţilor să acceseze, prin intermediul unor construcţii de programare tradiţionale, orice element al mediului interactiv Paradox.

Să înţelegem modelul Paradox

Începătorii în Paradox, adesea, se acomodează greu. Paradox este diferit de alte SGBD-uri tradiţionale; în special utilizatorii de Xbase au probleme de acomodare. Aceasta se întîmplă, în general, din cauză că programele Paradox (script) în loc să facă parte din mediul integrat Paradox, lucrează cu acesta, manipulează tabele în spaţiul de lucru şi navighează prin meniuri. Sub multe aspecte programarea în PAL se aseamănă mai mult cu programarea în limbajul de programare a macrourilor aparţinînd unui program de calcul tabelar, cum ar fi Lotus 1-2-3 sau Quattro Pro, decît cu programarea în dBase sau în clonele sale.

O comparaţie a modelului Paradox cu modelul Xbase ne va ajuta să clarificăm diferenţele.

Modelul Xbase

În nucleul dBase-ului şi al altor produse Xbase stă un mecanism orientat pe articole (Fig. 1). Acest mecanism este proiectat pentru a manipula articole sau linii stocate în fişiere. El le poate indexa, poate căuta anumite valori, executa calcule, face selecţii şi chiar realiza legături rudimentare.

Mecanismul Xbase este optimizat pentru sarcinile pe care le are de îndeplinit. Cînd programatorii în Xbase au nevoie să "înhaţe" un articol, atunci o pot face mai elegant şi mai rapid decît în oricare alt limbaj.

Mecanismul orientat articol este manipulat direct de limbajul de programare Xbase, care este, în toate privinţele, un limbaj tradiţional foarte puternic. El, de asemenea, este optimizat pentru a manipula articole, utilizînd sau nu indexuri, în funcţie de dorinţa utilizatorului sau programatorului.

În exteriorul limbajului de programare Xbase se află interfaţa utilizator. Cei care utilizează interactiv produsele Xbase lucrează direct în această interfaţă, dar de fapt ei nu fac altceva decît să creeze şi să execute mici programe (task-uri) aflate în spatele scenei. În cazul celei mai simple interfeţe, prompterul punct, utilizatorii introduc cîte o singură comandă a limbajului la un moment dat. Din centrul de control al lui dBase IV, atunci cînd utilizatorul selectează generatorul de machete pentru a crea o machetă pentru introducerea de date şi apasă tasta "save", dBase generează sute de linii de cod. Acest cod este interpretat de mecanism pentru a afişa şi manipula date utilizînd această machetă.

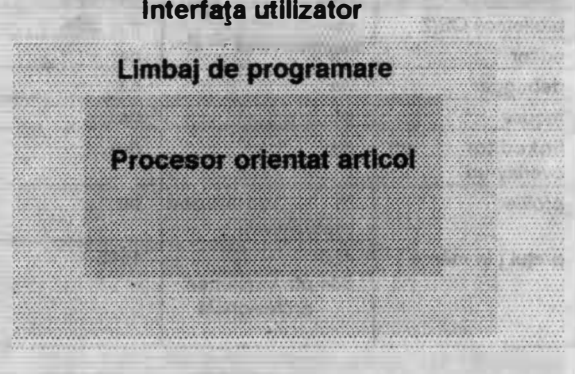


Fig. 1
Modelul Xbase. Limbajul Xbase este situat la mijloc între procesorul de articole şi interfaţa utilizator. Programul manipulează articole la nivelul de jos iar interfaţa utilizator trebuie codificată.

În loc să lucreze cu mediul interactiv, un programator avizat poate crea uşor această machetă în limbajul de programare Xbase, lucrînd cu comenzi familiare.

Din cauză că interfaţa utilizator şi limbajul de programare sînt legate foarte uşor în acest fel şi deoarece interfaţa utilizator se află în exteriorul stratului de cod, este simplu pentru produsele Xbase să aibă interfeţe utilizator diferite. În fapt Ashton-Tate a schimbat interfaţa utilizator a dBase-ului de cel puţin trei ori, iar diferitele clone, cum ar fi FoxPro şi dBXL, îşi au propriile interfeţe utilizator.

Funcţia interfeţei utilizator, în modelul Xbase, este de a crea şi de a manipula cod. Cum se ajunge la nucleul limbajului, fie printr-o interfaţă utilizator (front end), prin intermediul unui limbaj, sau scriind direct într-un editor, practic este lipsit de importanţă.

Modelul Paradox

Modelul Paradox (Fig. 2) diferă de modelul Xbase în trei puncte esenţiale.

În nucleul Paradox-ului, ca şi în cel al dBase-ului, se află tot un mecanism orientat articol. Indiferent de structura sau de modelul SGBD-ului, interfeţei utilizator, sau limbajului de programare, fie că este orientat pe articole sau pe seturi, la un moment dat trebuie manipulate articole individuale stocate în fişiere.

Procesorul orientat pe articole, momentan, este disponibil ca un produs Borland distinct: Paradox Engine. Acest produs este destinat programatorilor în C, C++, Pascal sau Windows care doresc să realizeze aplicaţii cu baze de date utilizînd formatele fişier Paradox, cu un suport complet pentru indexuri, protecţie prin parolă şi protocoale de blocaj multiutilizator. Paradox Engine nu are capabilităţi de interfaţă utilizator, programatorul trebuind să-şi scrie propria interfaţă sau să utilizeze una dintre bibliotecile puse la dispoziţie.

Orientarea tabelă. Prima zonă majoră în care Paradox diferă de Xbase este mijlocul diagramei. Imediat în exteriorul stratului procesorului de articole se află un procesor orientat tabelă, care lipseşte cu desăvîrşire în dBase.

Paradox este proiectat pentru a manipula tabele, în timp ce dBase a fost proiectat pentru a manipula articole. Din acest punct de vedere Paradox este mult mai închis spre un SGBD relaţional, care este de asemenea orientat tabelă. De ex., spre deosebire de Xbase, Paradox nu stochează un număr de articol ca pe un pointer la o linie dintr-o tabelă, ci fiecare linie este identificată prin valoarea primei ei chei primare.

Procesorul orientat tabelă este optimizat pentru a manipula tabele ca seturi relaţionale. "Inteligenţa" înglobată permite Paradox-ului să selecteze o abordare eficientă la nivelul procesării articolelor care stă la baza tuturor operaţiilor pe seturi.

Această optimizare analizează strategia corectă pentru o operaţie pe seturi de fiecare dată cînd este executată. Optimizarea ţine cont de memoria disponibilă şi de celelalte resurse, mărimea tabelelor, prezenţa indexurilor, şi de o mulţime de alţi factori. O aceeaşi cerere poate fi executată utilizînd abordări diferite de fiecare dată.

În consecinţă Paradox execută comenzile relaţionale la nivel de seturi mai rapid decît majoritatea SGBD-urilor. Acest lucru se întîmplă deoarece Borland a avut abilitatea să implementeze procesorul orientat tabele ca o soluţie curată şi elegantă client-server în noul produs Paradox SQL Link; spre deosebire de dBase, Paradox dispune de un procesor SQL puternic integrat în program.

Odată ce se înţelege orientarea pe tabele a Paradox-ului, există o serie de trucuri care pot fi folosite pentru a simplifica programarea.

Orientarea obiect. Tabelele Paradox sînt un tip specific de obiecte, exact cum sînt şi machetele şi rapoartele Paradox. Mult din programarea în Paradox se rezumă la a manipula obiectele construite în program, a naviga printre tabele, a completa imaginile pentru cereri, a selecta machete şi a lista rapoarte.

Aceasta este a doua categorie de diferenţe între Xbase şi Paradox: în Xbase se consumă mai mult timp manipulînd cod pentru a executa operaţii de bază, care în Paradox sînt implementate ca obiecte predefinite.

Deşi Paradox nu are nici una din facilităţile limbajelor orientate

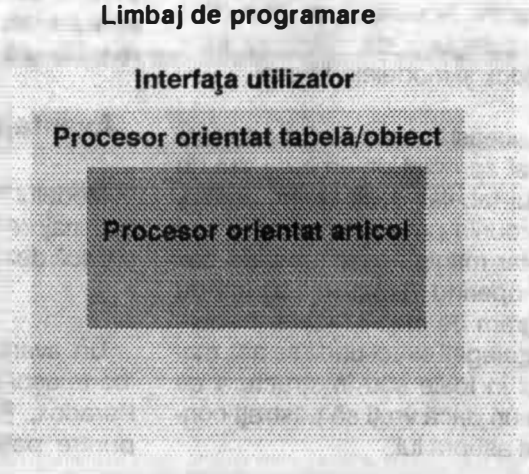


Fig. 2. Modelul Paradox. Modelul Paradox permite PAL să manipuleze construcţii de nivel înalt incluzînd tabele (cu operaţii la nivel de set) şi toate facilităţile interfeţei utilizator

obiect, cum ar fi SmallTalk sau Lisp, el are mai mult dintr-o abordare obiect decît Xbase-ul. Nu există cod PAL nici în machete nici în rapoarte. Acestea există ca şi obiecte binare discrete care pot fi manipulate direct în loc să trebuiască procesate sute de linii de cod.

Interfaţa utilizator. Al treilea nivel în modelul Paradox îl constituie interfaţa utilizator. Aceasta constă din sistemul de meniuri inelar extensiv, incluzînd şi capabilităţile query by example, editorul de machete şi de rapoarte şi diferite module minore, cum ar fi sistemele grafic, de sortare şi de protecţie prin parolă.

Interfaţa utilizator Paradox este locul în care se creează obiectele Paradox, incluzînd tabele, indexuri, machete, rapoarte, setări de imagini şi condiţii de validare a datelor. Acestea se realizează interactiv lucrînd cu ajutorul meniurilor sau editoarelor înglobate.

Unde este PAL ?

Cel mai la exterior nivel este limbajul de programare, care se află "deasupra" interfeţei utilizator. Programele Paradox (script) manipulează interfaţa utilizator ca şi cînd ar apăsa tastele corespunzătoare în locul Dvs. A plasa o tabelă în spaţiul de lucru presupune a executa aceleaşi operaţii cînd programaţi, care ar trebui executate şi interactiv. Aceasta este cea de-a treia categorie de diferenţe între Paradox şi modelele Xbase.

În această abordare Paradox este mai apropiat de un program de calcul tabelar decît de un SGBD. Macrourile programelor de calcul tabelar manipulează foaia de calcul, operînd direct în ea. Într-un program de calcul tabelar trebuie să înţelegeţi exact unde se află macroul în foaie sau în structura de meniuri dacă vreţi să păstraţi controlul asupra lui.

În cazul Paradox-ului se întîmplă acelaşi lucru. Pentru a executa cu succes programele Dvs., trebuie să înţelegeţi exact ce situaţie a creat programul Dvs. în interfaţa utilizator şi în spaţiul de lucru.

De ex., adăugînd datele dintr-o tabelă la cele din altă tabelă se schimbă tabela curentă din spaţiul de lucru. Trebuie să fiţi conştient de aceasta şi programul Dvs. va trebui fie să îndepărteze această tabelă din imagine, fie s-o utilizeze în continuare ca tabelă curentă. În mod similar dacă tastaţi `Do-It!` `[F2]` într-o machetă multitabelară, cursorul sare înapoi în tabela proprietar (master). Interactiv, dacă doriţi să rămîneţi în tabela de detaliu, va trebui pur şi simplu să tastaşi `[F3]` sau `[F4]` pînă cînd ajungeţi din nou în tabela în chestiune. Programul (script-ul) Dvs. va trebui să facă acelaşi lucru.

Această legătură închisă cu mediul interactiv este una dintre raţiunile pentru care oamenii gîndesc uneori că Paradox-ului îi lipseşte un limbaj de programare robust şi puternic şi că deci este imposibil de a se realiza în el aplicaţii dificile.

De fapt PAL este un limbaj de programare complet, permiţînd variabile dinamice, funcţii şi comenzi definite de utilizator, gestiunea memoriei virtuale şi un set complet de comenzi, funcţii şi construcţii aparţinînd programării structurate. El este un limbaj foarte complet care are deci acces deplin la oricare trăsătură a mediului interactiv.

Avantaje şi dezavantaje

Modelul Paradox are două avantaje majore faţă de modelul Xbase şi două dezavantaje relativ minore.

Un avantaj este acela că este foarte uşor să creezi prototipuri în Paradox. Puteţi naviga interactiv printre paşii necesari, să salvaţi cheile tastate şi codul rezultat este *acelaşi* program pe care l-aţi fi putut scrie cu mîna cu ajutorul unui editor. Paradox permite un număr de moduri de a salva ceea ce aţi făcut într-un program, care ulterior poate fi executat sau combinat în programe mai mari şi mai complete.

Al doilea avantaj major al modului în care este proiectat Paradox este acela că Paradox permite crearea unor utilitare de productivitate ridicată care să opereze direct în mediul Paradox. De ex., nu este dificil să se creeze un program mic care să plaseze în spaţiul de lucru valori de validare pentru cîmpul curent. Programul navighează prin toate aceste meniuri pînă cînd găseşte valorile pe care le caută, le salvează în variabile şi le afişează pe ecran.

Un dezavantaj al modelului Paradox îl reprezintă performanţele mai slabe pe care le obţin programele PAL trecînd prin interfaţa utilizator.
Reprezentarea suplimentară a interpretărilor scade viteza de execuţie a unui program. În mod uzual aceasta nu este o problemă deoarece comenzile PAL realizează mult mai mult într-o linie decît programele Xbase.

De ex., comenzile pentru a afişa o tabelă şi a utiliza o machetă predefinită sînt:

```
view     "CLIENTI"
pickform     "1"
```

Toate indexurile sînt deschise şi aplicate în mod automat. Nu este necesar cod (sursă program) pentru a desena macheta, din moment ce Paradox cunoaşte la cel mai de jos nivel cum să interpreteze şi să folosească obiectul binar.

(continuare în pag. 43)

(R.M.)

Static sau dinamic ?

OOP - Partea a treia

Tema principală a acestei a treia părţi a cursului o constituie metodele virtuale. Cu ajutorul lor, în ultima parte a acestui curs, vor fi tratate structurile dinamice.

Metodele prezentate în primele două părţi ale acestui curs au fost în totalitate statice. Aceasta înseamnă că încă din momentul compilării, se rezervă spaţiul de memorie necesar şi se rezolvă cu ajutorul acestor metode toate referinţele.

Facilitatea de a redefini pentru obiecte derivate, metode noi avînd acelaşi nume ca şi cele vechi, permite adaptarea metodelor în funcţie de necesităţi.

Definiţia obiectului Gr_Pozitie din Unit-ul BPunct din listingul LIST13.PAS este o încercare de definire generală a unui obiect grafic. Obiectul derivat din acesta, Gr_Punct, conţine toate metodele necesare pentru iniţializarea, desenarea, ştergerea şi mutarea unui punct.

Această miniierarhie poate fi extinsă după dorinţă. Toate imaginile imaginabile, cum ar fi cercuri, pătrate sau elipse pot fi derivate din Gr_Pozitie. Pentru fiecare din aceste obiecte grafice există metoda Deseneaza. Dar sub acelaşi nume în funcţie de caracteristicile imaginii grafice se vor ascunde desigur implementări diferite.

Se poate spune şi că fiecare obiect grafic posedă caracteristica de a se putea desena singur pe ecran. Cuvîntul Deseneaza este utilizat pentru a desemna toţi algoritmii posibili de desenare. Acest lucru este uşor de înţeles. Dacă din obiectul Gr_Punct este derivat, de ex., obiectul Gr_Cerc, atunci metodele **Init** şi **Deseneaza** vor trebui reprogramate, fără îndoială. Un cerc necesită ca parametru pentru iniţializare şi raza. Rutina de desenare a unui punct este alta decît cea pentru desenarea unui cerc.

Daca se programează static atunci şi metodele Muta_Abs sau Muta_Rel trebuie redefinite, cu toate că secvenţa de cod nu se modifică. Acest lucru nu s-ar întîmpla dacă în momentul compilării, adresele rutinelor **Sterge** şi **Deseneaza** ar rămîne deschise. Atunci ar exista o posibilitate ca în momentul execuţiei să se stabilească metodele **Sterge** şi **Deseneaza** care să fie folosite. Tocmai această posibilitate o oferă programarea orientatăobiect. Turbo-Pascal, începînd din versiunea 5.5, sprijină mecanisme pentru tratarea metodelor virtuale.
{Listing LIST13.PAS}

Pentru o mai bună înţelegere vom mai numi odată diferenţele dintre metodele statice şi cele virtuale. În cazul metodelor statice, în momentul compilării se stabileşte exact ceea ce trebuie să facă. În cazul metodelor virtuale decizia asupra a ceea ce trebuie făcut rămîne deschisă.

Cum funcţionează aceasta ? Unit-ul Gr_Obj conţine noile definiţii. Cuvîntul rezervat **Procedure** înaintea declaraţiilor Punct.Init şi Cerc.Init este înlocuit cu cuvîntul **Constructor**. Un constructor este un antet special de procedură, care rezolvă cîteva iniţializări interne pentru mecanisme virtuale. El este necesar deci pentru a putea lucra cu metode virtuale. Înafară de aceasta constructorul trebuie apelat neapărat înaintea primului apel al unei metode virtuale. El stabileşte referinţele necesare cu ajutorul aşa numitei **VMT** (Virtual Methods Table - tabela metodelor virtuale). În VMT pe lîngă mărimea obiectelor sînt conţinuţi şi pointeri la codul metodelor obiectului.

Dacă nu este apelat Constructor, în locul de ramnificaţie spre o metodă virtuală se află o altă adresă. Urmarea poate fi o blocare a sistemului.

Compilatorul nu are nici o posibilitate de a verifica succesiunea apelurilor metodelor. Este deci sarcina programatorului de a rezolva acest lucru. Singurul ajutor aflat la dispoziţie în Turbo_Pascal este comutatorul {$R+}.

Acesta, gîndit de fapt pentru o verificare a domeniilor, verifică la toate metodele virtuale starea de iniţializare a instanţelor. în cazul în care anumite iniţializări lipsesc, în timpul execuţiei, va fi dat un mesaj de eroare domeniu şi execuţia programului va fi întreruptă. Acest comutator este foarte util în faza de dezvoltare a unei aplicaţii. După testarea programului, comutatorul {$R+} poate fi îndepărtat din nou, programul devenind astfel mai rapid.

Dacă într-o ierarhie de obiecte o metodă este declarată odată virtuală, atunci toate metodele avînd acelaşi nume, ale acestei ierarhii, trebuie să fie de asemenea virtuale. O metodă statică nu poate înlocui niciodată o metodă virtuală. Auxiliarul **Virtual** trebuie scris deci în declaraţiile metodelor pentru toate metodele cu acelaşi nume ale unei ierarhii de obiecte.

Listingul LIST13.PAS clarifică avantajele pe care le aduce utilizarea metodelor virtuale. Obiectul Punct defineşte toate metodele care sînt necesare pentru manipularea unui obiect grafic. Pe lîngă Constructor pentru iniţializare sînt implementate şi rutinele pentru desenarea, ştergerea şi mutarea unui punct. Oricare altă imagine grafică, fie ea un cerc, un unghi drept sau o curbă oarecare, necesită o metodă **Deseneaza** specială pentru reprezentarea ei, dar şi o metoda specială **Sterge**. Metodele pentru mutarea absolută şi relativă însă nu mai trebuie definite şi pentru urmaşi.

Obiectul **Punct** conţine, să zicem aşa, o metodă mamă pentru mutarea tuturor obiectelor grafice care sînt urmaşii obiectului Punct. Pentru acest scop în rutina **Punct.Muta_Abs** şi **Punct.Muta_Rel** numele metodelor **Deseneaza** şi **Sterge** nu conţin nici o referinţă la un obiect anume. Astfel de metode, cum sînt cele două pentru mutarea obiectelor grafice, sînt desemnate în programarea orientată obiect ca metode abstracte. Ele utilizează metode virtuale. În acest caz se va hotărî abia în momentul execuţiei programului care obiect anume trebuie mutat. {Listing GR_OBJ.PAS}.

În partea a doua a acestui curs vom arăta că obiectele pot fi tratate dinamic în acelaşi mod în care pot fi tratate şi variabilele. Unit-ul Gr_Obj demonstrează cum pot arăta astfel de declaraţii. Tipurile de pointeri Gr_PozitiePtr, Gr_PunctPtr şi Gr_CercPtr permit crearea şi îndepărtarea unor obiecte din heap (memoria disponibilă). Pentru acest scop procedurile Turbo-Pascal **New** şi **Dispose** au fost astfel extinse încît să se poată referi şi la obiecte. Pentru utilizarea lui New există două posibilităţi:

```
New (UnCerc);
New (UnCerc,Init(100,100,50));
```

În ambele cazuri se rezervă suficientă memorie în heap pentru a permite o instanţiere a acestui obiect. Apoi **New** transmite înapoi adresa acestei zone de memorie în pointer. În al doilea caz rezervarea de memorie este făcută de constructorul **Init**. Avantajul celei de-a doua metode este la îndemînă. În acest caz odată cu rezervarea spaţiului de memorie necesar, este apelat simultan şi constructorul Init, şi ne asigurăm în acest fel că iniţializările necesare vor fi efectuate. Înafară de aceasta programatorul economiseşte o instrucţiune, deoarece în cazul primei versiuni mai trebuie declarat şi constructorul.

Ca şi toate celelalte tipuri de date tradiţionale, care sînt depuse în heap, şi obiectele pot fi îndepărtate din memorie cu ajutorul procedurii **Dispose**. Dar nu întotdeauna apelul lui **Dispose** este cea mai bună soluţie. Un obiect individual poate conţine pointeri la alte structuri de date sau obiecte, care ar trebui eliberaţi într-o anumită ordine. În acest caz este de bun simţ ca această eliberare a memoriei să fie cuprinsă într-o singură metodă.

Turbo-Pascal oferă această metodă prin intermediul lui **Destructor**. Cuvîntul rezervat Destructor înlocuieşte cuvîntul **Procedure**. Apelul lui **Destructor** uneşte eliberarea zonei de memorie rezervate cu toate celelalte operaţiuni de "curăţenie" necesare.

Cu aceasta devine evident că destructorii acţionează asupra obiectelor dinamice. Ei îşi găsesc întrebuinţarea atunci cînd din heap trebuie îndepărtate obiecte polimorfe. Un pointer la Cerc poate referi un anumit număr de pointeri de tipul Punct. În general dimensiunile obiectelor individuale diferă, deci în momentul compilării nu se cunoaşte mărimea zonei de memorie ce va trebui rezervată. Aceasta explică de ce utilizarea individuală a lui Dispose este interzisă.

Un destructor consultă VMT-ul instanţei apelante. Acolo este memorată dimensiunea în bytes a obiectului. O eliberare reală a memoriei nu se obţine însă decît atunci cînd Destructor este apelat ca o extensie a procedurii Dispose.

```
Dispose (UnCerc,Done)
```

În acest caz destructorul Done este prelucrat ca o metodă normală. După ce instrucţiunea End este atinsă, dimensiunea instanţei este transmisă procedurii Dispose care preia munca de eliberare propriu-zisă. Metoda Destructor însăşi poate să fie goală, eliberarea memoriei se produce totuşi. Ea se realizează practic în fundal.

```
Destructor Done;Virtual;
Begin
End;
```

Exemplul de program "OOP_Incercare Virtuala" scoate în evidenţă mecanismele virtuale. Toate declaraţiile de obiecte sînt plasate în Unit-ul Gr_Obj. Variabilele pointer "UnPunct", "UnCerc" şi "OLinie" pot indica instanţe ale acestui tip de obiect. Ele trebuie deci create cu New. În liniile:

```
New (UnPunct,Init(200,200));
New (UnCerc,Init(100,100,50)); şi
New(OLinie,Init(300,100,150,150));
```

este repetată sintaxa extinsă a procedurii **New**. Cu ajutorul acestor instrucţiuni se rezervă spaţiul necesar de memorie pentru instanţe şi în acelaşi timp este apelat constructorul corespunzător. Apelul virtual propriu-zis este realizat prin sarcina Muta_Rel a instanţei "UnCerc" din linia:

```
UnCerc^.Muta_Rel(50,50);
```

Pentru a putea urmări exact procesul trebuie utilizat modul "Trace Into" al mediului integrat de dezvoltare. Deoarece tipul de obiect Gr_Cerc nu are definită o metodă proprie pentru mutare relativă este utilizată metoda moştenită de la obiectul Gr_Punct. În rutina **Gr_Punct.Muta_Rel** sînt prezente metodele **Sterge** şi **Deseneaza**. În acest caz se stabileşte în momentul execuţiei dacă vor fi utilizate metodele **Sterge** şi **Deseneaza** ale lui Gr_Punct, ale lui Gr_Cerc sau ale lui Gr_Linie. În exemplu sînt utilizate cele ale lui Gr_Cerc.

Primele trei părţi ale acestui curs pot fi rezumate în cele ce urmează: Obiectele sînt tipuri de date definite de utilizator, care înglobează structuri de date şi metodele corespunzătoarte lor. Accesul la date pe o altă cale decît prin intermediul metodelor nu este posibil. Datele sînt deci încapsulate. Tipurile de obiecte pot lăsa moştenire altor obiecte toate caracteristicile lor. În acest mod pot fi construite structuri ierarhice arborescente.

Metodele moştenite pot fi declarate din nou cu acelaşi nume în obiectele fiu. Un obiect fiu poate utiliza atît noile metode definite cît şi pe cele vechi. Utilizarea aceloraşi nume pentru desemnarea unor metode diferite într-o ierarhie este denumită polimorfie.

Obiectele, ca şi variabilele, pot fi tratate şi dinamic. În acest scop procedurile New şi Dispose au fost extinse pentru a putea opera asu-

pra obiectelor. Turbo-Pascal susţine mecanismele pentru metode virtuale. Utilizînd aceste metode se decide abia în momentul execuţiei programului ceea ce este de făcut. Înainte de a utiliza mecanisme virtuale acestea trebuie iniţializate cu ajutorul constructorilor. La eliminarea din memorie a obiectelor dinamice se recomandă utilizarea destructorilor.

Cu aceasta fundamentele programării orientate obiect au fost enunţate. În cursul următor vom clarifica modul de aplicare al acestor tehnici prin intermediul cîtorva exemple.

Metode virtuale

Ele sînt declarate tot la fel ca şi metodele statice, doar că se adaugă cuvîntul Virtual. Toate metodele care vor purta acest nume vor trebui declarate de asemenea virtuale. Ele prezintă faţă de metodele statice avantajul că abia în momentul execuţiei se decide care metodă anume trebuie utilizată. Acest lucru este dependent de tipul instanţei apelante.

Constructor

este un cuvînt rezervat, care înlocuieşte cuvîntul rezervat Procedure. O rutină constructor iniţializează mecanismul metodelor virtuale şi trebuie apelată înainte de apelul lor. El stabileşte o legătură între instanţa care apelează constructorul şi VMT-ul obiectelor.

Virtual Methods Table (VMT)

memorează dimensiunea obiectelor şi conţine pentru fiecare metodă virtuală un pointer la codul ei.

Destructor

Un destructor este o metodă specială, pentru eliminarea de obiecte din memorie şi eliberarea memoriei. Conlucrînd cu procedura extinsă Dispose, zona de memorie ocupată de un obiect poate fi eliberată pînă la ultimul byte. Eliberarea propriu-zisă se petrece în fundal, astfel încît un destructor îşi îndeplineşte funcţia chiar şi atunci cînd nu conţine nici o instrucţiune.

(R. M.)

Paradigma Paradox

(continuare din pag. 40)

În spatele scenei, aceste două comenzi execută mai multe acţiuni:

`[F10]` afişează meniul
`"view"` selectează această opţiune a meniului
`CLIENT` tastează numele tabelei în dreptul prompterului
`[Enter]` acceptă intrarea
** tabela apare pe ecran
`[F10]` afişează meniul
`"Image"` selectează această opţiune a meniului
`"Pickform"` selectează această opţiune a meniului
`"1"` selectează această machetă din lista machetelor disponibile
** imaginea de pe ecran se modifică de la tabelă la machetă.

Performanţele obţinute lucrînd în mediul interactiv sînt îmbunătăţite de un alt punct forte al Paradox-ului, managerul memoriei virtuale, care manipulează eficient aplicaţiile care sînt mai mari decît memoria aflată la dispoziţie. Paradox ajustează dinamic memoria asignată tabelelor, buffer-elor, stivelor şi modulelor programului, în funcţie de resursele hardware disponibile şi de operaţiile executate de către utilizator.

Plasarea în Paradox a limbajului PAL deasupra interfeţei utilizator este restrictivă în alt mod: este dificil pentru Borland să modifice interfaţa utilizator. Din cauză că programele sînt atît de strîns legate de interfaţa utilizator, a o schimba ar însemna să faci aceste programe neutilizabile. Aceasta nu înseamnă că schimbarea este imposibilă ci doar că implică un efort complex şi delicat.

Deci care este concluzia ?

Pentru utilizatorii finali mediile Xbase şi Paradox sînt similare. În fiecare din ele utilizatorul lucrează în contextul interfeţei utilizator fără să ţină cont de faptul că reprezentarea interpretărilor este sau nu prezentă.

Proiectanţii, oricum, *trebuie* să ţină cont de relaţia cu modul de reprezentare. În Paradox, de ex., un programator poate uza de reprezentarea tabelelor pentru a obţine maximum de productivitate. În Xbase trebuie manipulată reprezentarea articolelor pentru a obţine aceeaşi funcţionalitate.

Probabil că cea mai mare diferenţă este aceea că în Paradox programatorul *trebuie* să se plaseze în afara mediului interactiv. Paradox este un program "modal". Atunci cînd navigaţi prin meniuri, selectînd, acţionînd şi manipulînd obiecte, Paradox navighează prin diferite moduri de operare. Fiecare mod este definit de meniul de opţiuni care este disponibil atunci cînd tastaţi `[F10]`.

Anumite acţiuni sînt permise numai în anumite moduri. De ex., unealta de copiere a unei tabele sub un alt nume este apelabilă numai din meniul principal (main). Dacă tastaţi `[Alt-F9]`, (sau determinaţi programul să tasteze `[Alt-F9]`), atunci intraţi în modul `"CoEdit"`, în care acţiunile posibile sînt cele disponibile în acest meniu; copierea unei tabele nu este una dintre aceste opţiuni.

Acest lucru nu este neapărat "mai bun". Avantajele Paradox-ului sînt legate de alte aspecte, cum ar fi creearea de prototipuri şi abilitatea de a manipula date la nivel de set.

Programatorilor Xbase le ia timp pînă să folosească avantajele date de modul în care este proiectat Paradox, de aceea este important să se înţeleagă mediul interactiv şi modelul Paradox înainte de a plonja în PAL.

(R. M.)

```
Unit BPunct;

Interface

Uses Graph;

Type

Gr_Pozitie = Object
  Gr_X, Gr_Y : Integer;
  Procedure Init (X,Y : Integer);
  Function GetPos_X : Integer;
  Function GetPos_Y : Integer;
End;

Gr_Punct = Object (Gr_Pozitie)
  P_Culoare : Integer;
  Procedure Init (X,Y :Integer);
  Procedure Deseneaza;
  Procedure Sterge;
  Procedure Muta_Rel(Xd,Yd:Integer);
  Procedure Muta_Abs
            (X_Nou,Y_Nou:Integer);
End;

Implementation

Procedure Gr_Pozitie.Init
          (X,Y : Integer);
Begin
  Gr_X := X;
  Gr_Y := Y;
End;

Function Gr_pozitie.GetPos_X : Integer;
Begin
  GetPos_X := Gr_X;
End;

Function Gr_Pozitie.GetPos_Y : Integer;
Begin
  GetPos_Y := Gr_Y;
End;

Procedure Gr_Punct.Init
          (X,Y : Integer);
Begin
  Gr_Pozitie.Init (X,Y);
End;

Procedure Gr_Punct.Deseneaza;
Begin
  PutPixel (Gr_X,Gr_Y,GetColor);
End;

Procedure Gr_Punct.Sterge;
Begin
  PutPixel (Gr_X,Gr_Y,GetBkColor);
End;

Procedure Gr_Punct.Muta_Rel
          (Xd,Yd : Integer);
Begin
  Sterge;
  Gr_X := Gr_X + Xd;
  Gr_Y := Gr_Y + Yd;
  Deseneaza;
End;

Procedure Gr_Punct.Muta_Abs
          (X_Nou,Y_Nou : Integer);
Begin
  Sterge;
  Gr_X := X_Nou;
  Gr_Y := Y_Nou;
  Deseneaza;
End;

End.

{Listing LIST13.PAS}
```

```
Program OOP_Incercare_Virtuala;

Uses Crt,Graph,Gr_Obj;

Var
  UnPunct : Gr_PunctPtr;
  UnCerc : Gr_CercPtr;
  OLinie : Gr_LiniePtr;
  Gr_Driver,Gr_Mode : Integer;

Begin
  DetectGraph (Gr_Driver,Gr_Mode);
  InitGraph (Gr_Driver,Gr_Mo-
de,'C:\TP\BGI');
  New (UnPunct,Init(200,200));
  UnPunct^.Deseneaza;
  ReadLn;
  UnPunct^.Muta_Rel(100,0);
  New (UnCerc,Init(100,100,50));
  UnCerc^.Deseneaza;
  ReadLn;
  UnCerc^.Muta_Rel (50,50);
  ReadLn;
  Dispose (UnCerc,Done);
  New (OLinie,Init(300,100,150,150));
  OLinie^.Deseneaza;
  ReadLn;
  OLinie^.Sterge;
  Dispose (Olinie,Done);
  Dispose (UnPunct,Done);
  CloseGraph;
End.

{Listing LIST15.PAS}
```

```
{$R+}
Unit Gr_Obj;

Interface

Uses Crt,Graph;

Type
  Gr_PozitiePtr = ^Gr_Pozitie;
  Gr_PunctPtr = ^Gr_Punct;
  Gr_CercPtr = ^Gr_Cerc;
  Gr_LiniePtr = ^Gr_Linie;

{ aici urmeaza LIST13.PAS incepind
                  de la linia 9 }

Gr_Pozitie = Object
  Gr_X,Gr_Y : Integer;
  Constructor Init (X,Y : Integer);
  Function GetPos_X : Integer;
  Function GetPos_Y : Integer;
End;

Gr_Punct = Object (Gr_Pozitie)
  P_Culoare : Integer;
  Constructor Init (X,Y : Integer);
  Destructor Done;Virtual;
  Procedure Deseneaza;Virtual;
  Procedure Sterge;Virtual;
  Procedure Muta_Rel (Xd,Yd : Inte-
ger);Virtual;
  Procedure Muta_Abs
    (X_Nou,Y_Nou : Integer);Virtual;
End;

Gr_Cerc = Object (Gr_Punct)
  Raza : Integer;
  Constructor Init (X,Y,R : Integer);
  Procedure Deseneaza;Virtual;
  Procedure Sterge;Virtual;
End;

Gr_Linie = Object (Gr_Punct)
  XE,YE : Integer;
  Constructor Init
    (X,Y,XEnd,YEnd : Integer);
  Procedure Deseneaza;Virtual;
  Procedure Sterge;Virtual;
End;

Implementation

Constructor Gr_Pozitie.Init
    (X,Y : Integer);
Begin
  Gr_X := X;
  Gr_Y := Y;
End;

Function Gr_Pozitie.GetPos_X : Integer;
Begin
  GetPos_X := Gr_X;
End;

Function Gr_Pozitie.GetPos_Y : Integer;
Begin
  GetPos_Y := Gr_Y;
End;

Constructor Gr_Punct.Init
    (X,Y : Integer);
Begin
  Gr_Pozitie.Init (X,Y);
End;

Procedure Gr_Punct.Deseneaza;
Begin
  PutPixel (Gr_X,Gr_Y,GetColor);
End;

Procedure Gr_Punct.Sterge;
Begin
  PutPixel (Gr_X,Gr_Y,GetBkColor);
End;

Procedure Gr_Punct.Muta_Rel
    (Xd,Yd : Integer);
Begin
  Sterge;
  Gr_X := Gr_X + Xd;
  Gr_Y := Gr_Y + Yd;
  Deseneaza;
End;

Procedure Gr_Punct.Muta_Abs
    (X_Nou,Y_Nou : Integer);
Begin
  Sterge;
  Gr_X := X_Nou;
  Gr_Y := Y_Nou;
  Deseneaza;
End;

Destructor Gr_Punct.Done;
Begin
End;

Constructor Gr_Cerc.Init
    (X,Y,R : Integer);
Begin
  Gr_Pozitie.Init (X,Y);
  Raza := R;
End;

Procedure Gr_Cerc.Deseneaza;
Begin
  Circle (Gr_X,Gr_Y,Raza);
End;

Procedure Gr_Cerc.Sterge;
Var Co : Word;
Begin
  Co := GetColor;
  SetColor (GetBkColor);
  Deseneaza;
  SetColor (Co);
End;

Constructor Gr_Linie.Init
(X,Y,XEnd,YEnd : Integer);
Begin
  Gr_Pozitie.Init (X,Y);
  XE := XEnd;
  YE := YEnd;
End;

Procedure Gr_Linie.Deseneaza;
Begin
  Line (Gr_X,Gr_Y,XE,YE);
End;

Procedure Gr_Linie.Sterge;
Var Co : Word;
Begin
  Co := GetColor;
  SetColor (GetBkColor);
  Deseneaza;
  SetColor(Co);
End;

End.
```

{Listing GR_OBJ.PAS}

Sisteme expert şi asociativitate

Reţele neuronale

Partea a III-a

Sistemele expert acumulează fapte şi reguli. în bazele lor de cunoştinţe; pornind de la acestea şi folosind strategii, ele raţionează.

Cuprinsul cursului:

- Partea 1: Privire de ansamblu asupra temei
- Partea 2: Teoria algoritmului propagării înapoi
- **Partea 3: Sisteme expert şi asociativitate**
- Partea 4: Recunoaşterea scrisului

Sistemele expert simulează cunoaşterea umană în anumite domenii. De aceea se mai numesc şi sisteme bazate pe cunoştinţe. În general sînt compuse din componente diferite. Pe de o parte, baza de cunoştinţe, în care sînt stocate cunoştinţele ce urmează a fi prelucrate (fapte şi reguli). Apoi, maşina inferenţială, folosind strategiile de control preprogramate, "raţionează", emiţînd judecăţi proprii bazate pe datele existente şi cunoştinţele stocate. Pentru o utilizare prietenoasă, acestora li se mai adaugă de regulă o componentă de dialog, care pune întrebări şi preia răspunsurile, o componentă de achiziţie, care "prepară" ştiinţa experţilor pentru a fi preluată în baza de cunoştinţe şi o componentă explicativă, care are menirea de a lăsa să se vadă cum anume ajunge programul să îşi formeze concluziile.

Să considerăm următorul exemplu simplu, care se compune dintr-un fapt şi o regulă. Regula este: "orice om merge în poziţie verticală", faptul este "Bogdan este un om". Se poate concluziona de aici că Bogdan merge în poziţie verticală, deşi acest fapt nu este specificat în mod explicit. El este conţinut implicit în baza de cunoştinţe.

Spre înţelegere, încă două noţiuni. Orice regulă se compune dintr-o parte condiţională şi o parte de acţiune:

DACĂ condiţie ATUNCI acţiune

În exemplul dat, condiţia este "cineva este om" iar acţiunea corespunde "mersului în poziţie verticală". De aceea, regula se mai poate formula şi

DACĂ cineva este om,
ATUNCI el merge în poziţie verticală.

Nu vom intra în amănunte relative la cum se implementează sistemele expert folosind limbaje de programare logică. Ne interesează mai mult care este rolul reţelelor neuronale în acest domeniu.

Este clar că situaţii care pot fi descrise exact cu reguli şi fapte se abordează cel mai bine folosind tehnicile verificate ale programării logice. Dar deja în exemplul dat se poate constata că acest lucru nu este neapărat lipsit de dificultăţi: regula dată nu este general valabilă. Pentru bebeluşi de exemplu, ea nu este adevărată. Şi mai dificilă devine situaţia în care ar trebui prelucrate cunoştinţe nesigure. Nu orice caracteristică poate fi descrisă clar prin două stări (lat sau îngust, lung sau scurt). De cele mai multe ori, ea se manifestă numai în anumite proporţii (merele de exemplu nu sînt 100% roşii sau verzi, ci, de obicei, "şi-şi") sau prin anumite mărimi (1,94m în loc de "înalt").

Cercetarea unor forme adecvate pentru reprezentarea de cunoştinţe nesigure în calculator constituie fundamentul dezvoltării de sisteme IA (inteligenţă artificială).

Acest proces e denumit în general "reprezentarea cunoştinţelor". Se disting în cadrul ei pe de o parte reprezentări simbolice, care pentru prelucrare folosesc tehnici bazate pe reguli logice, şi reprezentări subsimbolice, care apar din modul de lucru al reţelelor neuronale. Care procedeu este mai bun într-un caz particular, trebuie încercat pe aplicaţia concretă.

Cînd şi cum se folosesc însă reţelele neuronale pentru sisteme expert ? Folosirea lor se recomandă atunci cînd partea condiţională a regulilor se compune din mai multe proprietăţi sau cerinţe (de exemplu: "cineva este om" şi "vîrsta lui e mai mare de 2 ani") şi cînd pentru partea de acţiune sînt disponibile mai multe variante (de exemplu "merge aplecat", "merge tîrîş", "aleargă", "sare" etc.). Diversele condiţii ale regulilor se interpretează ca neuroni de intrare ai unei reţele neuronale, unde fiecare proprietate se reprezintă printr-o valoare numerică (de exemplu numărul de ani pentru vîrstă). Variantele posibile ale părţii de acţiune sînt văzute ca şi clase diferite şi li se atribuie neuroni de ieşire diferiţi.

Pentru clarificarea modului de lucru vom prezenta aplicarea metodelor neuronale la un sistem expert exemplu. (*)

Scopul: trebuiesc clasificate nişte obiecte pe baza unor proprietăţi. În afară de aceasta, se doreşte recunoaşterea unor reprezentanţi ai claselor care nu îndeplinesc 100% condiţiile de apartenenţă. Ca obiecte, fie date următoarele tipuri de fructe: cireşe, portocale, mere, pere, prune, banane, struguri, lămîi. Ca proprietăţi

	cireaşă	portocală	măr	pară	prună	banană	struguri	lămîie
are sîmburi	da	da	da	da	da	nu	da	da
coaja e comestibilă	da	nu	da	da	da	nu	da	nu
formă rotundă	da	da	da	nu	nu	nu	da	nu
autohton	da	nu	da	da	da	nu	da şi nu	nu
dulce	da	da	da	da	da	da	da	nu
acru	nu	da	da	nu	nu	nu	da	da
culoare roşie	da	nu	da şi nu	nu	nu	nu	nu	nu
culoare portocaliu	nu	da	nu	nu	nu	nu	nu	nu
culoare albastru	nu	nu	nu	nu	da	nu	da	nu
culoare galben	nu	nu	da şi nu	da şi nu	nu	da	nu	da
culoare verde	nu	nu	da şi nu	da şi nu	nu	nu	da	nu

sînt disponibile: cu sau fără sîmbure, coajă comestibilă, formă rotundă, sînt autohtone, dulce, acru, culoare roşie, culoare portocalie, culoare albastră, culoare verde, culoare galben.

Mai întîi, obiectele cu proprietăţile lor sînt adunate într-un tabel. Se vede imediat că nu este foarte simplu de luat decizii exacte. Merele de exemplu sînt mai degrabă roşii, galbene sau verzi? Deoarece există atît roşii cît şi galbene sau verzi, şi nici una din culori nu este obligatorie, probabil că un "da şi nu" neutru descrie cel mai bine situaţia.

La culori contează numai cele de la fructul copt, astfel că "verde" de exemplu nu intră în discuţie pentru lămîi şi banane. Bazată pe dimensiunea tabelei se construieşte o reţea neuronală care să reprezinte corect proprietăţile şi clasele. Cele 11 proprietăţi sînt interpretate ca 11 celule de intrare. Celor 8 obiecte de clasificat li se asociază cîte o celulă de ieşire, şi anume celula de ieşire nr. 1 îi corespunde obiectului "cireaşă", celula de ieşire nr. 2 obiectului "portocală" ş.a.m.d. Pentru reţeaua neuronală sînt stabilite astfel straturile de intrare şi de ieşire. Reţelele cu propagare înapoi se caracterizează printr-un strat intermediar suplimentar (hidden). Se alege un strat intermediar de 10 celule. Cu ajutorul editorului integrat, se pot produce datele de antrenament pentru reţea. Ele constau din vectori cu 11 componente (cîte una pentru fiecare caracteristică), în cadrul cărora "nu" e reprezentat prin valoarea 0 iar "da" prin valoarea 255. Aici apare o problemă tipică de reprezentare a cunoştinţelor. Cum să fie reprezentat "şi da, şi nu" ca valoare numerică cu care să lucreze reţeaua neuronală ?

Două alternative pot fi luate în considerare: "şi da, şi nu" se poate considera valoare intermediară, ca 128 deci; sau, altfel, se admite atît "da" cît şi "nu". Atunci se stochează două modele diferite, unul cu o componentă nulă pentru "nu" şi unul cu o componentă de valoare 255 pentru "da". În felul acesta se realizează a doua posibilitate, căci "şi da, şi nu" admite ambele cazuri, la urma urmei.

Reţeaua se "antrenează" cu 19 modele de antrenament, care se compun din 8 modele diferite pentru măr, 4 pentru pară, două pentru struguri şi cîte un vector pentru cireaşă, portocală, prună, banană şi lămîie. Obiectivul fazei de antrenament constă în modificarea tuturor ponderilor în aşa fel încît fiecare din cele 19 modele de învăţare să fie clasificat corect. Căci fiecărui vector de intrare îi corespunde un vector de ieşire: astfel de exemplu vectorului de intrare "cireaşă" (255, 255, 255, 255, 255, 0, 255, 0, 0, 0, 0) îi corespunde vectorul de ieşire (1, 0, 0, 0, 0, 0, 0, 0, 0, 0) care reprezintă prima clasă de ieşire.

În acest moment ar putea apare ideea că acest lucru ar putea fi realizat banal, prin simpla stocare a perechilor de antrenament prin intermediul unei tabele indexate. În acest caz, valorile de ieşire nu ar fi calculate, ci s-ar regăsi ca într-un dicţionar. Foarte mici abateri ale "cuvîntului" căutat faţă de cuvîntul memorat ar duce însă la eşecul căutării.

Deoarece valorile de intrare sînt ponderate la reţelele neuronale, numerele individuale contribuie mai mult sau mai puţin la calculul rezultatului, al vectorului de ieşire. De aceea, valori asemănătoare la intrare produc valori asemănătoare la ieşire. Se vorbeşte astfel de capacitatea asociativă a reţelelor neuronale. Modele de intrare incomplete se pot astfel clasifica corect sau completa.

Proprietăţile asociative trebuiesc dezvoltate în faza de învăţare. Este limpede că acest lucru devine cu atît mai dificil cu cît trebuie reprezentaţi mau mulţi parametri de intrare (neuroni de intrare) şi mostre de antrenament. Dacă în plus modele foarte diferite aparţin unei aceleiaşi clase, procesul de convergenţă este îngreunat suplimentar. De aceea, mai ales începătorilor li se întîmplă deseori

ca sistemul să fie divergent, deci învăţarea să nu aibă succes. Ca rezultat al unui proces de convergenţă nereuşit, reţeaua reprezintă modelul pe o celulă de ieşire greşită, neîndeplinindu-şi astfel scopul.

Ca proiectant începător de reţele neuronale, ce-i de făcut? Este iarăşi evident că selecţia regulii de învăţare corecte, mărimea parametrilor potriviţi, structura reţelei (numărul de neuroni) şi reprezentarea cunoştinţelor sînt totdeauna dependente de problemă. Dacă ar exista penru aceasta o regulă general valabilă, atunci aceasta ar fi fost deja de mult implementată în soft. Astfel, toate problemele reprezentabile prin această tehnică ar fi deja rezolvate. Deci este necesară o anumită experienţă în utilizarea acestor tehnici pentru a atinge obiectivele dorite. Ceea ce nu e o noutate, desigur: şi pentru a bate un cui în perete, trebuie să ştii să umbli cu ciocanul, deoarece altfel apar probleme cu degetele.

Pentru a rezolva problema de mai sus a clasificării fructelor, a dus la reuşită regula de învăţare 1, predefinită, cu parametri predefiniţi beta1 = 1.0 şi beta3 = 1.0, toleranţa predefinită T = 850 şi condiţia de abandon 3, care abandonează în mod automat procesul de învăţare cînd pentru fiecare din modelele de învăţare sînt satisfăcute condiţiile de toleranţă impuse.

Deoarece regula de învăţare nr. 1 lucrează pendinte de un generator aleator, mai multe teste diferite au dus la rezultate diferite pentru durata procesului de învăţare.

Pentru a "învăţa" cele 19 modele, sistemul a avut nevoie de 130 pînă la 270 de iteraţii. Ceea ce pe un AT (lent) de 10 MHz corespunde unei durate de 1-2 minute. Nu trebuie uitat însă că un începător are nevoie de un timp semnificativ mai mare pentru a găsi valorile optimale menţionate mai sus.

După terminarea cu succes a fazei de învăţare se poate şi trebuie verificată "capacitatea" reţelei. Ceea ce se face modificînd, cu editorul incorporat, valorile unui model existent sau producînd pur şi simplu modele noi care sînt predate apoi spre evaluare (identificare) reţelei, prin simplă apăsare a unei combinaţii de taste. În felul acesta se pot face rapid şi comod studii particularizate. Pentru verificarea profesională a capacităţilor învăţate se pot testa şi fişiere întregi, în mod automat, şi ieşi cu rezultatele fie la ecran, fie la imprimantă. Se vede că reţeaua "învăţată" e "testată" cu modele necunoscute (ei). Rezultatele nedorite se corectează, preluînd mostrele greşit identificate în fişierul de antrenament şi refăcînd faza de învăţare, pentru a îmbunătăţi astfel performanţele reţelei. Prin aplicarea repetată a combinaţiei învăţare/verificare, prin intermediul setului (fişierului) de date de antrenament, se obţine o apropiere din ce în ce mai mare de caracteristicile optime dorite ale reţelei.

Proprietatea de asociativitate mai sus pomenită a fost testată pe un caz "dur". Se cere antrenarea unei reţele care să recunoască cuvinte tastate greşit. Pentru acest lucru au fost selectate 12 comenzi DOS: CHDIR, CHKDSK, COPY, DEL, DIR, FORMAT, MKDIR, MODE, PATH, PRINT, RMDIR, TYPE. Ca intrare în reţea se foloseşte valoarea corespunzătoare ASCII a literei, deci de exemplu pentru CHDIR vectorul (67, 72, 68, 73, 82). Pentru a îngreuna sarcina reţelei, se permit cuvinte avînd pînă la 7 caractere, astfel că CHDIR se poate scrie >_CHDIR_<, >__CHDIR< sau >CHDIR__< cu vectorii de intrare total distincţi (20, 67, 72, 68, 73, 82, 20), (20, 20, 67, 72, 68, 73, 82), (67, 72, 68, 73, 82, 20, 20). Fiecăruia din cele 12 cuvinte i se asociază o celulă de ieşire a reţelei. Scopul este de a antrena reţeaua în aşa fel încît celula de ieşire corespunzătoare lui CHDIR să se activeze de exemplu şi cînd la intrare este >CHDIT<. În felul acesta pot fi eliminate erorile de tastare. Pentru a putea aprecia modul de punere al problemei, ar trebui ţinut seama şi de asemănarea între trei din comenzile selectate: CHDIR, RMDIR şi MKDIR. Ca fişier de antrenament s-a pornit pentru început cu cîte două variante deplasate pentru >FORMAT< şi >CHKDSK<, respectiv cîte 3 variante pentru celelalte cuvinte, astfel că în total s-au obţinut 34 de modele pentru antrenament. Fişierul de test în schimb conţine toate variaţiile posibile, dar cu o singură literă greşită - în total, 3.856 de cuvinte.

După o învăţare reuşită, cu 16 neuroni ascunşi şi regula de învăţare nr. 1, din cele 3856 de mostre necunoscute, 2170 au fost clasificate corect. Deci din 34 mostre de antrenament se asociază 2170. Prin adăugarea altor mostre în fişierul de antrenament, rezultatul a putut fi îmbunătăţit: cu 123 mostre de antrenament (şi 18 neuroni ascunşi) au fost clasificate corect 2818 cuvinte; cu 198 mostre de antrenament (şi 20 de neuroni ascunşi), 3191.

Pentru memorarea ponderilor necesare la 20 de neuroni ascunşi şi a informaţiilor relative la topologia reţelei sînt necesari numai 4 kbyte de memorie.

Cine îşi face gînduri relativ la numărul mare de mostre de antrenament, va realiza că interesantele proprietăţi ale reţelelor neuronale ar putea fi îmbunătăţite prin combinare cu alte tehnici. Cum anume se face acest lucru, veţi afla în episodul următor.

(I.F.)

(*) Acest curs ar trebui desigur secondat de simulatorul la care se fac referiri. Sperăm că experienţele descrise pot arunca totuşi o geană de lumină asupra domeniului. chiar şi aşa, "pe uscat".(N.red.)

Parolă variabilă

N-aţi fost niciodată în situaţia de a porni sistemul cu un musafir nepoftit în spatele Dvs ? Nu vi s-a întîmplat, mai ales dacî nu tastaţi foarte repede, ca acesta să "prindă" parola de acces şi să trebuiască s-o modificaţi ?

Ei bine, dacă da, vă puteţi rezolva problema în 20 de minute, asamblînd programul ce urmează.

Vă propun o parolă care să fie data curentă în forma "llzz" pentru varianta prezentată în listing sau "zzll" dacă inversaţi liniile marcate prin comentarii ca linia 1 şi linia 2.

Programul PASSW este foarte strict: trebuie să tastaţi exact 4 cifre corecte. Mai multe caractere tastate (se admit maximum 101) sau un singur caracter greşit duc la blocarea sistemului într-o buclă infinită din care doar reset-ul îl poate scoate - şi atunci o luăm de la capăt.

PASSW nu afişează caracterele tastate, ci doar cîte un punct pentru a facilita controlul introducerii lor. Aţi greşit un caracter ? Nici o problemă: puteţi utiliza liniştit backspace pentru corectarea erorii.

Deci: creaţi fişierul PASSW.ASM cu ajutorul unui editor, apoi, în ordine, executaţi comenzile DOS

> tasm passw
> tlink passw

(sau echivalent, dacă nu lucraţi cu Turbo - Assembler).

Pentru ca PASSW.EXE pe care l-aţi obţinut să se execute automat după pornirea sistemului, trebuie ca fişierul AUTOEXEC.BAT să conţină liniile exemplificate în PASSW.BAT

Dacă totul a mers corect, nu rămîne decît să resetaţi PC-ul şi să introduceţi în ordine data, ora (le puteţi lăsa nemodificate) şi parola. Nu uitaţi: data introdusă va fi cea cerută de parolă - chiar dacă este una falsă. Atunci cînd vă secondează un musafir nepoftit, nu aveţi decît să introduceţi o dată falsă, pentru a fi sigur că nici în acea zi el nu va putea avea acces la PC-ul Dvs. Chiar dacă reuşeşte să urmărească ceea ce tastaţi, parola pe care o va desoperi e valabilă doar o singură zi pe an.

(Daniel Buleu, Iaşi)

```
1           data    segment
2   ED      db  08h, ' ', 08h, '$'
3   EPAR    db  ' Parola :$'
4   LOC     db  0,0,0,0
5   LC      db  101 dup (0)
6   EAD     db  0dh, 0ah, 07h, 07h, 20 dup (' ')
            db  , 'Accesul interzis !!'
7           db  07h, 0ah,'Opreste calculatorul !!$'
8   EOK     db  0dh, 0ah, 0ah, 0ah, 'Bine ati venit !!!!$'
9   data    ends
10
11  code    segment
12  assume  cs:code, ds:data,ss:code
13
14  THA     proc
15          mov     ah,0
16          mov     bh,10
17          div     bh
18          add     ax, 3030h
19          mov     LOC [si], al
20          inc     si
21          mov     LOC [si], ah
22          ret
23  THA     endp
24
25  start:
26          mov     ax, data
27          mov     ds, ax
28          mov     ah, 9
29          mov     dx, offset EPAR
30          int     21h
31          mov     ah, 2ah
32          int     21h
33          mov     si,0
34          mov     al, dh
35          call    THA
36          inc     si
37          mov     al, dl
38          call    THA
39          mov     si, 0
40  beg:    mov     ah,07
41          int     21h
42          cmp     al,0dh
43          je      e1
44          cmp     al,8
45          jne     e0
46          mov     ah,9
47          mov     dx, offset ED
48          int     21h
49          dec     si
50          jmp     beg
51  e0:     mov     LC [si], al
52          inc     si
53          cmp     si,100
54          je      ad
55          mov     ah,2
56          mov     dl,'.'
57          int     21h
58          jmp     beg
59  e1:     dec     si
60          cmp     si, 3
61          jne     ad
62  e2:     mov     dh, LOC [si]
63          cmp     dh, LC [si]
64          jne     ad
65          dec     si
66          cmp     si, 0
67          jne     e2
68          mov     ah, 9
69          mov     dx, offset EOK
70          int     21h
71          jmp     quit
72  ad:     mov     ah,9
73          mov     dx, offset EAD
74          int     21h
75          jmp     ad
76  quit:   mov     ah,4ch
77          int     21h
78  code    ends
79          end  start
80
```

PASSW.ASM

```
date
time
echo off
break off
passw
break on
echo on
```

(în) AUTOEXEC.BAT

Protecţia programelor

Cu filtrul prezentat în cele ce urmează se poate proteja orice program EXE sau COM. Se inserează ca primă instrucţiune în fişier instrucţiunea "ret", ce are codul pe un singur octet (0c3h). Efectul este că la apelul programului se execută acest "ret", astfel că programul se termină şi se întoarce în DOS. Prin această inserare, sistemul de operare este determinat să trateze şi fişierele EXE ca pe cele COM. Prin urmare, fişierele EXE mai mari de 64kbyte care au fost protejate prin MODIF vor anunţa printr-un mesaj de eroare că nu încap în memorie, ceea ce altfel oricum însă nu deranjează. Repetarea comenzii MODIF pentru un fişier îndepărteză protecţia pusă. Fişierul de comenzi prezentat anexat permite protejarea comodă a programelor.

(Computer Persönlich, 7/91)

```
1   uses    Dos;
2   const   Max = $fd00;
3   var     R : registers;
4           b : array [1..Max] of byte;
5   begin
6       R.DX := Ofs(b); R.DS := Seg (b);
7       R.CX := 1;      R.BX := 0;      R.AH := $3f;
8       MsDos (R);
9       if b[1] $c3 then begin
10          b[2] := b[1];
11          b[1] := $c3;
12          R.CX := 2; R.BX := 1;       R.AH := $40;
13          MsDos (R);
14      end;
15      repeat
16          R.CX := Max;
17          R.BX := 0;
18          R.AH := $3F;
19          MsDos (R);
20          R.CX := R.AX;
21          R.BX := 1;
22          R.AH := $40;
23          MsDos (R);
24      until R.AX = 0;
25  end.
```

MODIFY.PAS

```
echo off
modify <%1> modify.tmp
copy modify.tmp %1 > NUL
del modify.tmp > NUL
echo on
```

MODIF.BAT

Reset la imprimantă

PRNRESET poate fi folosit într-o aplicaţie BASIC pentru a produce resetarea imprimantei. Pentru aceasta, programul conţine o scurtă rutină în asamblor, care este introdusă în memorie folosind instrucţiuni POKE. La lansare trebuie specificată imprimanta (portul paralel) ce trebuie resetat.

(Computer Persönlich, 10/91)

```
1   1 DRESET = &HFF00
2   2 DATA &h55, &h89, &he5, &h8b, &h76, &h6, &h8b, &h14
3   3 DATA &hb4, &h1, &hcd, &h17, &h5d, &hca, &h2, &h0
4   4 FOR I=0 TO 15: READ B%
5   5 POKE (DRESET+I), B% : NEXT I
6   10 PRINT  "Reset la imprimanta !"
7   20 INPUT  "Imprimanta (0,1,2): ", A%
8   30 CALL DRESET (A%)
```

PRNRESET.BAS

Ieşire în DOS

De multe ori în unele aplicaţii am vrea să putem executa comenzi DOS, fără a ieşi din aplicaţia noastră. O alternativă o oferă programul TPDOS.PAS. Ideea programului a pornit de la programul demonstrativ al firmei Borland, EXECDEMO.PAS, livrat odată cu mediul de programare Turbo Pascal 5.5.

Îmbunătăţirea faţă de EXECDEMO.PAS este afişarea prompterului (ca pentru pg). În acest sens am folosit procedura GETDIR(.,.), unde primul parametru este un întreg ce reprezintă unitatea (0 = curentă, 1 = A, 2 = B, ...), iar al doilea parametru este un string ce reprezintă calea. Propun îmbunătăţirea programului DOS.PAS prin interpretarea unei comenzi PROMPT cu sintaxa: PROMPT [$litera], unde prin [ ] am marcat o secvenţă ce se poate repeta.

(Ioan Tiberiu Socaciu, Cluj)

```
1   {$M 8192, 0, 0}
2
3   uses DOS;
4   type comanda = string[127];
5   var ok:       boolean;
6       path:     string[100];
7       drive:    integer;
8       c:        comanda;
9
10  procedure command (c: comanda);
11  begin
12      SwapVectors;
13      Exec (GetEnv ('COMSPEC'), '/C ' + C);
14      SwapVectors;
15      if DosError < > 0 then
16          WriteLn ('Could not execute COMMAND.COM');
17      WriteLn;
18  end;
19
20  begin
21      writeln;
22      writeln ('DOS Simulator');
23      writeln ('(C) 1991 by SOKYSOFT');
24      writeln ('type soky to exit');
25      writeln;
26      ok: = true;
27      drive: = 0;
28      repeat
29          getdir (drive, path);
30          write (path, ' > ');
31          readln (c);
32          if c < > 'soky' then command (c)
33                          else ok: = false;
34      until not ok
35  end.
```

TPDOS.PAS

Există o dischetă în unitate ?

Ideea din următoarea funcţie este că, folosind perifericul NUL al DOS se verifică dacă unitatea de disc este pregătită (cu dischetă) sau nu. Programul este scris în Pascal.

(Computer Persönlich, 10/91)

```
1  program     TestDisk;
2  uses        crt;
3  var         ch: char;
4  const       ESC = #27;
5
6  FUNCTION    E_disc_in_unit ( drive : char) : boolean;
7
8  var
9       f: file; dstr: string;
10 begin
11      E_disc_in_unit : = false;
12      dstr : = drive + ':\NUL';
13      Assign (f, dstr);
14      FileMode : = 0;
15      {I-}
16      Reset (f);
17      {I+}
18      if IOResult = 0
19           then begin
20                E_disc_in_unit : = true;
21                Close (f);
22           end;
23      FileMode : = 2;
24 end;
25
26 begin
27      WriteLn ('Se iese din test apasind <ESC>');
28      repeat
29           ch : = ReadKey;
30           if  E_disc_in_unit ('A')
31           then      WriteLn ('Este discheta in unitatea A:')
32           else      WriteLn ('Nu este discheta in unitatea A:');
33      until ch = ESC;
34 end.
```

TESTDISC.PAS

```
1  #include <conio.h>
2  #include <dos.h>
3  #include <stdio.h>
4
5  void main (void)
6  {
7       char     far * cFP;
8       char     far * cHP;
9       short i;
10
11      FP_SEG (cFP) = FP_SEG (cHP) = 0x4000;
12      FP_OFF (cFP) = FP_OFF (cHP) = 0xff67;
13
14      for (i = 0; i < 20; i + + )
15      {
16           printf ("Far-Pointer: %04x:%04x, "
17                "Huge-Pointer: %04x:%04x\n",
18                FP_SEG (cFP), FP_OFF (cFP),
19                FP_SEG (cHP), FP_OFF (cHP) );
20           cFP + = 0x10;
21           cHP + = 0x10;
22      }
23 }
```

FARHUGE.C

Autoverificare contra viruşilor

Cu ajutorul unităţii SELFCHK ce v-o prezentăm mai jos, un program oarecare Turbo Pascal poate să-şi dea singur seama dacă este sau nu atacat de un virus. Unit-ul, care nu exportă nimic, constă dintr-o parte de iniţializare care depune o sumă de control în program şi o parte de verificare, care la fiecare lansare a programului - exceptînd-o pe prima - verifică faptul că suma calculată coincide cu cea memorată. În caz de diferenţe, rămîne la latitudinea utilizatorului să oprească lucrul sau să ignore pericolul. În caz de coincidenţă, programul se desfăşoară normal. Pentru instalarea unit-ului, este suficientă numirea lui în directiva Uses.

(Computer Persönlich, 7/97)

```
1  unit SelfChk;
2  Interface
3  Uses Crt;
4  Implementation
5  var f          : File;
6      i, citit   : Word;
7      pw, dd     : Array [1..16] of Char;
8      sum        : Array [1..4] of Char;
9      xsum, chksum : LongInt;
10 begin
11     pw : = 'if 5/91 PRACTICA';
12     sum : = #0#0#0#0;
13     xsum : = 0;
14     Assign (f, ParamStr(0)); Reset (f,1);
15     repeat
16          BlockRead (f, dd, 16, citit);
17          for i : = 1 to 16 do xsum : = xsum + Ord(dd[i]);
18     until (citit < 16) or (dd = pw);
19     Blockread (f, chksum, 4);
20     if chksum = 0 then begin
21          WriteLn ('Checksum marcat !');
22          Seek (f, FilePos(f)-4);
23          BlockWrite (f, xsum, 4);
24          Close (f);
25     end else begin
26          Close (f);
27          if (chksum = xsum) and (citit = 16) then begin
28               WriteLn ('Autotest terminat cu succes!');
29          end else begin
30               WriteLn ('Pericol de infectie cu virus.');
31               WriteLn ('Continuare sau Stop (C,S): ');
32               if UpCase (ReadKey) < > 'C' then Halt (1);
33          end;
34     end;
35 end.
```

SELFCHK.PAS

Pointeri FAR şi HUGE în C

Modul de lucru cu segmente de memorie a procesoarelor Intel a dus la împărţirea pointerilor din C în trei categorii: "near", "far" şi "huge". Dacă un pointer "near" este doar un offset de 16 biţi la segmentul curent, celelalte două tipuri sînt pointeri de 32 biţi, constînd din segment şi offset.

Atît pointerii "far" cît şi cei "huge" permit adresarea întregului spaţiu de memorie. Diferenţa între ele este determinată de mărimea obiectelor de administrat. În timp ce pointerii "far adresează obiecte nu mai mari de 64 kbyte, cu pointeri "huge" se poate lucra cu obiecte de orice mărime. De aceea, la incrementarea respectiv decrementarea unui pointer "huge" se foloseşte o rutină mai complexă, care să controleze depăşirile de segment. Aceste diferenţe sînt demonstrate de programul care urmează. În timp ce pointerul "far" nu "se prinde" la ieşirea din limitele segmentului, pointerul "huge" modifică valoarea de segment.

(Computer Persönlich, 8/91)

Outsourcing

Păcatele scumpe ale conducătorilor oficiilor de calcul

Preşedintele avea o bănuială: oficiul de calcul lucrează ineficient. De-abia comanda către o întreprindere externă pentru preluarea prelucrării datelor ("outsourcing") a relevat potenţialul de scădera a costurilor. Outsourcing este în multe cazuri calea corectă pentru a evidenţia scumpele păcate ale conducătorilor oficiilor de calcul şi de a organiza prelucrarea datelor mai eficient.

Conducătorul oficiului de calcul era foarte calificat. 20 de ani de vechime în meserie. Consilierii de la Programator au ştiut imediat ce este de făcut: i-au dat un alt job, în care nu mai avea nici o legătură cu patronul lui de pînă acum. Iar oficiul de calcul (secţia de prelucrare a datelor) al ramurii din Germania a concernului scandinav Nokia a fost preluat de Programator, de personal care nu a avut pînă atunci nimic de-a face cu Nokia.

Programator procedează întotdeauna ca la Nokia. Întreprinderea scandinavă este specializată în a prelua în regie proprie oficiile de calcul ale întreprinderilor mari (cu peste 20 de angajaţi). Programator promite o scădere drastică a costurilor prelucrării datelor după preluare. Conform experienţei scandinavilor, cu personal extern pot fi evitate dezvoltări greşite în oficiile de calcul ale întreprinderilor.

Strategia de afaceri proprie Programator se hrăneşte din erori pe care oficiile de calcul "ale casei" le fac aproape întotdeauna în întreprinderile mari. Catastiful păcatelor şefilor oficiilor de calcul:

- hard se cumpără după buget. Sumele aprobate de preşedinte se cheltuiesc fără discernămînt, chiar dacă nu este necesar. Se cumpără cele mai performante calculatoare care "intră" în buget.

- decizii odată luate sînt apărate cu argumentul "bazei deja instalate". Domină justificarea investiţiilor anterioare. Gînduri asupra unor soluţii mai bune apărute între timp rămîn neformulate.

- Importanţa şefului oficiului de calcul în întreprindere este măsurată atît de către acesta cît şi de conducerea întreprinderii după numărul de persoane din subordine. Impulsurile pentru a organiza preluarea datelor mai eficient sau de a renunţa la lucruri inutile lipsesc.

Programator crede că are reţeta potrivită pentru a combate toate aceste neajunsuri: prelucrarea datelor ca o prestare de serviciu orientată spre piaţă. La client se face o prelevare exactă a datelor de care are nevoie şi cu ce mijloace pot fi ele culese cel mai avantajos. Prestarea realizată de Programator se facturează şi face posibilă, pentru conducerea întreprinderii, evidenţierea a ce servicii pentru cîţi bani s-a făcut.

Cu acest concept, Programator vrea să devină şi în Germania o casă de soft de frunte. În Suedia, întrepinderea s-a afirmat deja ca prestator de servicii informatice. Cu 2500 de angajaţi, Programator este cel mai mare consilier de întreprindere şi de prelucrarea datelor din Scandinavia. În concernul înfiinţat în 1964, de care aparţine şi firma de consultanţă pentru construcţii Jacobsen & Widmark, lucrează circa 4600 de persoane.

De faptul că prelucrarea datelor este mai ieftină ca soluţie conformă cu piaţa decît printr-o secţie proprie s-au lăsat convinse pînă acum printre altele societatea de stat pentrutelefoane, Svenska Televerkert, societatea de asigurări Skandia, concernul electrotehnic Ericsson precum şi filiala-mamă a concernului Nokia din Finlanda. Ideea s-a născut în 1979. Atunci Programator a preluat prelucrarea datelor pentru un producător de oţeluri superioare din Karlstad/Suedia. Între timp peste 20 de concerne se numără printre clienţi, de la uzina de ţesut celular pînă la specialistul în tehnica frigului Frigoscandia.

Drumul spre succes a trecut şi prin experienţe dureroase. "De la clienţi noi vrem să preluăm prelucrarea datelor complet, atît ca personal cît şi ca şi capital", declară Hans Peter Clieves, directorul Programator/ Germania. Rigoarea aceasta vine din experienţe nefaste făcute cu joint-ventures (societăţi mixte).

În faza de constituire, în unele cazuri Programator a acceptat participări de 50%. Jumătate din acţiuni aparţineau întreprinderii din care se separa prelucrarea datelor, cealaltă jumătate aparţinea concernului Programator. Modelul nu s-a dovedit viabil. Clieves: "O gîndire consecventă pentru prestarea de servicii nu are nici o şansă, dacă cel care a făcut comanda stă cu 50% în consiliul de conducere."

(I.F.)

8514/A	Advanced Display Adapter
A/D	Analog / Digital
AES	Application Environment System (Atari)
ALU	Arithmetic Logic Unit
ANSI	American National Standards Institute
API	Application Programm Interface
ARLL	Advanced Run Length Limited
ASCII	American Standard Code for Information Interchange
AT	Advanced Technology
ATF	Automatic Track Finding
BIOS	BASIC I/O System
CAD	Computer Aided Design Computer Aided Drafting
CADD	Computer Aided Drafting Design
CAM	Computer Aided Manufacturing
CASE	Computer Aided Systems Engineering
CBT	Computer Based Training
CCD	Charge Coupled Device
CD	Compact Disc
CD-ROM	Compact Disc - Read Only Memory
CDI	Compact Disc Interactive
CGA	Color Graphics Adapter
CISC	Complete Instruction Set Computer
CLI	Command Line Interface
CMOS	Complementary Metal Oxide Semiconductor
CPU	Central Processor Unit
CSMA/CD	Carrier Sense Multiple Access / Collision Detection
DAM	Direct Access Method
DAT	Digital Audio Tape
DBA	Data Base Administrator
DBMS	Data Base Management System
DD	Double Density
DDE	Dinamic Data Exchange
DDL	Data Definition Language
DDS	Digital Data Storage
DEC	Digital Equipment Corporation
DIP	Digital Image Processing
DMA	Direct Memory Access
DML	Data Manipulation Language
DOS	Disk Operating System
DR DOS	Digital Research DOS
DTP	DeskTop Publishing
DVI	Digital Video Interactive
EGA	Enhanced Graphics Adapter
EMM	Expanded Memory Manager
EMS	Expanded Memory Standard
EPROM	Erasable Programmable Read-Only Memory
FAT	File Allocation Table
FBAS	Farb-/Bild-/Austast-/Synchronsignal
GEM	Graphics Environment Manager
HC	Home Computer
HD	High Density
HGC	Hercules Graphics Adapter
HP	Hewlett Packard
HPGL	Hewlett-Packard Graphics Language
IBM	International Busines Machine
IEEE	Institute of Electrical and Electronics Engineers
IFF	Interchange File Format
ISO	International Standards Organization
I/O	Input/Output
JESSI	Joint European Submicron Sillicon
LAN	Local Area Network
LCD	Liquid Crystal Display
LED	Light Emitting Diode
LIM	Lotus Intel Microsoft
MCA	Micro Channel Architecture
MCGA	Multi Color Graphics Array Memory Controller Gate Array
MDA	Monochrome Display Adapter
MFM	Modified Frequency Modulation
MGA	(Hercules) Monochrome Graphics Adapter
MIDI	Musical Instrument Digital Interface
MIPS	Millions Instructions Per Seconds
MFLOPS	Millions FLOating Points OPeration per Seconds
MMU	Memory Management Unit
MS-DOS	Microsoft DOS
NEAT	New Enhanced Advanced Technology
NVR	Non Volatile RAM
OCR	Optical Character Recognition
OOP	Object Oriented Programming
OS/2	Operating System /2
PAL	Paradox Application Language
PbE	Prompt by Example
PC	Personal Computer
PCI	Personal Computer Instrumentation
PGA	Professional Graphics Adapter
PS	Personal System
QbE	Query by Example
RAM	Random Access Memory
RAW	Read After Write
RGB	Red Green Blue
RISC	Reduced Instruction Set Computer
RLL	Run Length Limited
ROM	Read Only Memory
SAA	Systems Applications Architectur
SCSI	Small Computers System Interface
SDK	System Developement Kit
SIMM	Single In line Memory Module
SQL	Structured Query Language
TFT	Thin Film Transistor
TIFF	Tag Image File Format
TOS	Tramiels Operating System (Atari)
TSR	Termine and Stay Resident
TTL	Transistor-Transistor Logic
VDI	Virtuell Device Interface
VGA	Video Graphics Array
VMS	Virtual Memory System
WYSIWYG	What You See Is What You Get
XGA	eXtended Graphics Array
XT	eXtended Technology

Din scrisorile primite la redacţie

"...la CAS' 85 s-a discutat şi despre faptul că n-avem timp să aşteptăm 20-30 de ani, pînă cînd se recunoaşte că neologismele respective s-au încetăţenit în limba curentă, astfel ca termenii să figureze în DEX şi să nu mai fim acuzaţi de barbarism tehnic.

Dicţionar de specialitate ? Încă din '87 am discutat cu cei de la Editura tehnică (în '88 a fost şi tov. Ilici la CAS) dar spuneau că n-au voie să scoată dicţionare, acesta fiind apanajul EA sau al Editurii ştiinţifice şi enciclopedice. Numai că termenii la care ne referim sînt mai degrabă tehnici, nu ştiinţifici. Poate acum s-au mai schimbat lucrurile.

Cristian Malide, Bucureşti

"Aş dori informaţii asupra posibilităţii de achiziţionare de soft şi documentaţie de firmă la preţuri rezonabile în lei, dacă există furnizori oficiali de soft în România şi care ar fi aceştia.

De asemenea ar fi bine ca revista dvs. să aibă posibilitatea contactării unor eventuali beneficiari ai produselor şi aplicaţiilor noastre (un fel de piaţă de soft). În această întreprindere ar fi interesaţi şi furnizorii de hardware, care fără aplicaţii nu pot exista. Anunţurile noastre nu sînt prea eficiente deoarece revista nu este citită de prea mulţi doritori de aplicaţii ci mai degrabă de specialişti (furnizori).

Mica publicitate

Ofer programe şi jocuri pentru calculatoare ATARI. Garantez calitatea înregistrărilor. Tel. 985/10.609.

Execut orice lucrări de proiectare de sistem şi aplicaţii, service, cursuri pentru firme posesoare de PC-uri ca şi colaborator extern sau angajat temporar. Tel. 90/42.25.05.

Ori este foarte greu să ne vindem unul altuia programe care întîmplător fac lucruri asemănătoare.

Constatînd cu tristeţe că nici anunţurile din ziarele de mare tiraj nu prea au efect pentru că marea masă a cititorilor nu este interesată în specialitatea noastră, iar directorii de întreprinderi sînt prea ocupaţi ca să citească şi mica publicitate ne vedem în situaţia de a nu putea oferi (chiar licita) programe utile uneori multora.

Aş sugera, dacă s-ar putea, să existe o "bursă" a programelor, prin care conducătorii unităţilor economice să fie atenţionaţi şi interesaţi în achiziţionarea de software. Altfel în curînd vom muri de foame."

Năstase Ileana, Bucureşti

Nu sînt atît de sceptic ca şi corespondenta noastră, dar evident nici mulţumit de ce (nu) am reuşit. Sînt multe cauzele ştiute sau bănuite; o mare parte din ele sînt inevitabile, sînt "datul problemei", dar o altă mare parte sînt vina noastră. Dacă este adevărat că anunţurile de vînzare/cumpărare hard şi cumpărare soft (aplicaţii) sînt de regulă mult prea "strînse" în timp ca să nu devină total neinteresante prin publicarea la un termen incert şi oricum la distanţă mare în timp - lucru pe care încă sperăm că vom reuşi să-l contracarăm, astfel încît revista noastră să ajungă în chioşcuri şi la abonaţi la dată fixă în fiecare lună - oricum lipsa anunţurilor de vînzare soft mă nedumereşte. Dacă am scris o aplicaţie şi am vîndut-o deja odată sau de cîteva ori, ea a ajuns deci la o anumită maturitate şi rotunjime, nu poate fi decît neglijenţă sau dezinteres în a nu repeta cel puţin de cîteva ori un anunţ care să o facă cunoscută în ţară, permiţîndu-mi să o vînd în zeci sau sute de locuri! Investiţia - financiară şi de timp - este derizorie în comparaţie cu cîştigurile probabile. Ideea că revista nu este citită de utilizatorii finali este, după părerea mea, doar parţial adevărată: chiar dacă e clar că nu am să fac reclamă unui program care-l concurează pe al meu, eu voi fi interesat să ştiu cît mai multe despre programe care să-l completeze sau să aducă alte foloase beneficiarilor pe care îi deservesc, şi care vor veni cu atît mai încrezători la mine cu cît li s-a întîmplat mai des să fie mulţumiţi de ceea ce au putut cumpăra de la mine - sau la recomandarea mea. Ca să nu vorbim de faptul că în mod normal pot conveni cu ceilalţi realizatori de aplicaţii pentru a încasa un procent pentru orice vînzare realizată prin intermedierea mea.

Vrînd-nevrînd, va trebui să ne intereseze mult mai mult desfacerea - lucru care pînă acum s-a realizat de regulă atît de defectuos încît nici nu merită pomenit. Pe de altă parte, cred că noi - profesioniştii în informatică - va trebui să facem un pas sau chiar mai mulţi înspre beneficiari, şi nu să aşteptăm să înveţe ei lucrurile pe care noi le ştim, fără ca nouă să ne pese prea mult de ce îi doare pe ei. Dar, mai ales, va trebui să mărim scara la care sîntem obişnuiţi să lucrăm - să nu ne mulţumim să vindem o aplicaţie scump, de cîteva ori, aşteptînd apoi ca beneficiarii să se înghesuie bazat doar pe reclama implicită pe care şi-o face produsul. Cu "bursa", vom mai vedea, vrem, dar n-avem putere să promitem.

Şi, în încheiere, scuze tuturor acelor care ne scriu şi cărora - deşi ar trebui să o facem - nu ajungem să le răspundem în mod individual.

Să auzim (numai) de bine!

Iosif Fettich

În numărul viitor:

- **La ce e bun codul de bare**
- **EMS - referinţă**
- **Şi totuşi, Cobol?...**

Important !

Revista "**if**" îşi propune editarea unui catalog al întreprinderilor (de stat sau particulare) a căror sferă de preocupări are tangenţă cu informatica (producători hard, producători soft, depanatori, distribuitori, comercianţi, etc.).
Catalogul va conţine adresele întreprinderilor, o scurtă descriere a obiectului de activitate al acestora şi o prezentare succintă a produselor şi serviciilor oferite.
Catalogul va fi distribuit prin reţeaua de difuzare a revistei "**if**", putînd ajunge astfel direct la persoanele interesate. Întreprinderile interesate de publicarea acestui catalog sînt rugate să ia legătura cu redacţia.

Important !

Căutăm difuzori/distribuitori autorizaţi pentru difuzarea revistei "**if**" !.
Condiţii avantajoase !
Rugăm persoanele interesate să ia legătura cu redacţia.

Talon de comandă

Doresc să-mi trimiteţi |____| exemplare ale revistei **If, pe o perioadă de** |_____| apariţii, pe adresa:

Nume&prenume ....

Strada: .... Nr.: ...... Bloc: ..... Ap.: .... Tel.: ....

Cod: .... Localitatea: .... Judeţ (sector) ....

Preţul unui exemplar este 57 lei. Preţul unui abonament se va calcula cu formula: 57 lei * nr. exemplare * nr. apariţii. Banii se vor expedia prin mandat poştal pe adresa: Micro ATCI, C.P. 64, O.P. 1, 4300 Tîrgu Mureş, Cont nr.: 40729-96010402 Banca Română pt. Dezvoltare Mureş.

If 5 / 91 Data: .... Semnătura. ....

Talon pentru mica publicitate

Vă rog să inseraţi în numărul viitor al revistei **If** următorul anunţ:

_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_
_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_	_

Preţul unui anunţ este de 15 lei pentru un rînd. Costul unui anunţ se calculează respectînd grila de mai sus - 15 lei per rînd început, indiferent cîte caractere conţine, şi se achită prin mandat poştal pe adresa firmei Micro ATCI, cont 40.729-96.01.04.02 deschis la Banca Română pt. Dezvoltare Mureş, cel tîrziu la o lună după apariţia anunţului în revistă. Pe mandat (la rubrica "Loc pentru corespondenţă") se va adăuga o explicaţie relativă la suma achitată.
Redacţia nu garantează în nici un fel apariţia anunţurilor achitate anticipat.
Garantez că deţin toate drepturile asupra produselor şi/sau serviciilor ofertate.

Data. .... Semnătura .... **If 5 / 91**

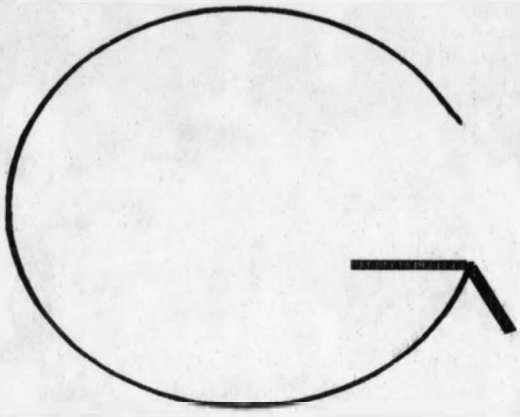
Important !

Revista **"If"** vă oferă spaţiu pentru publicitate şi reclamă la următoarele tarife :

o pagină, alb-negru,	
la prima apariţie	24.000 lei
la a doua apariţie succesivă	18.000 lei
de la a treia apariţie succesivă încolo	12.000 lei
o pagină, alb-negru şi o culoare suplimentară	
la prima apariţie	28.000 lei
la a doua apariţie succesivă	21.000 lei
de la a treia apariţie succesivă încolo	14.000 lei
o pagină color,	
la prima apariţie	36.000 lei
la a doua apariţie succesivă	27.000 lei
de la a treia apariţie succesivă încolo	18.000 lei

Pentru anunţuri publicitare care ocupă numai o parte din suprafaţa paginii, preţurile sînt desigur proporţional mai mici. Aceste tarife sînt vaiabile începînd de la 1 aprilie 1991; deoarece ele mai pot varia, vă rugăm să luaţi legătura cu redacţia înainte de contractarea unei reclame, pentru a verifica valabilitatea acestor tarife. Reclamele Dumneavoastră ne pot parveni îm mod obişnuit prin poştă, însoţite de o comandă; facturarea se va face în momentul apariţiei. Eventuale modificări de preţuri la tarifele pentru publicitate vor fi desigur făcute cunoscute beneficiarilor din timp.
În cazul contractelor de reclamă pe termen lung, beneficiarul poate anula comanda în orice moment, cît timp nu s-a predat încă materialul "bun de tipar" pentru numărul de revistă în care urma să apară reclama.
În cazul în care ne lansaţi o comandă pentru reclamă, vă rugăm să specificaţi şi luna (numărul) în care să apară prima dată. Pentru informaţii suplimentare vă stăm, desigur, la dispoziţie.

Micro ATCI - "If"
C.P. 64, O.P. 1, 4300 Tîrgu Mureş
Tel. 954 / 31660 (direct),
33612, 24158, 20057, 33511, interior 189 sau 134
Fax 954 / 35208
Telex 65354



GENESYS

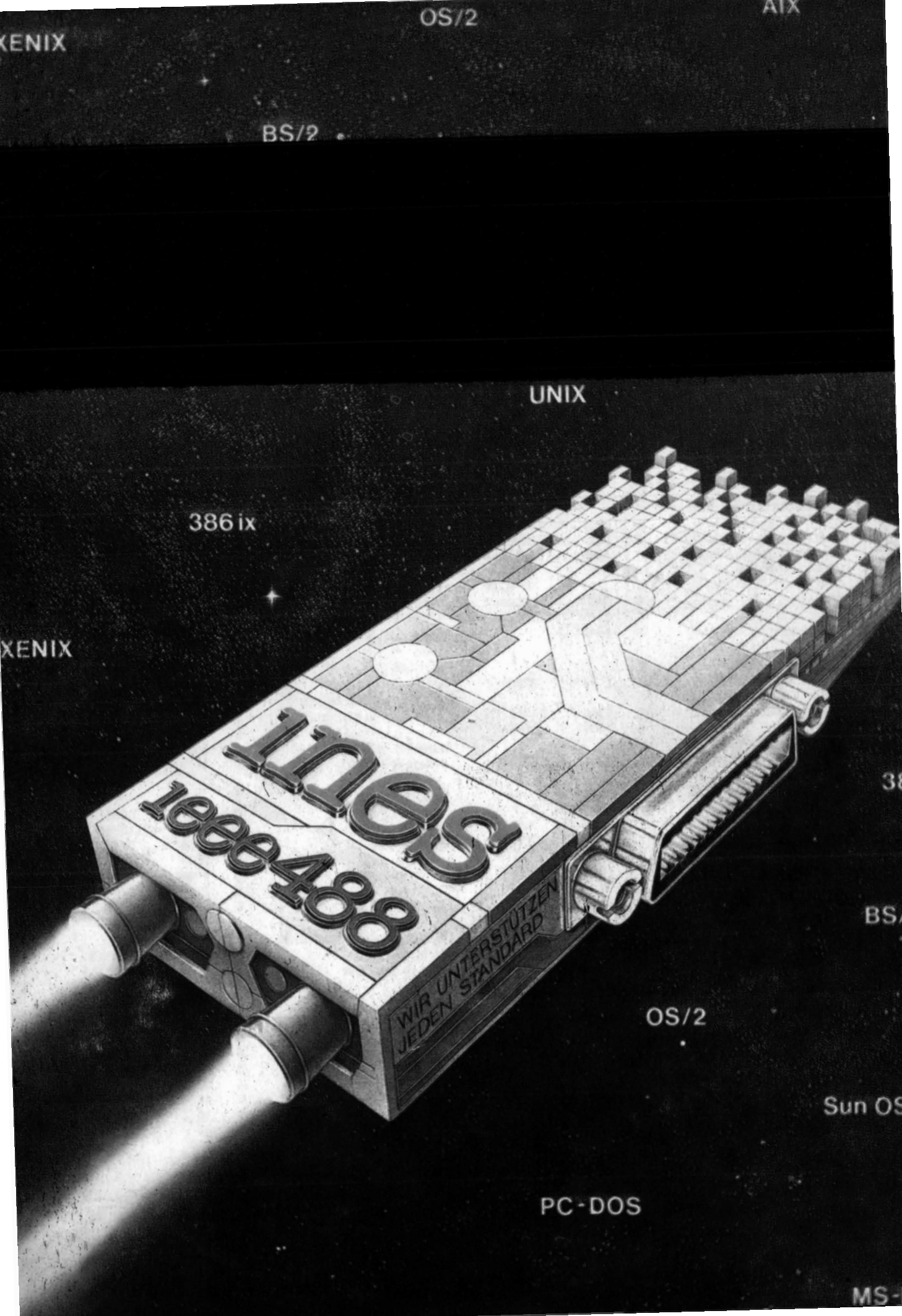
SZINTEZIS COMPUTER GYÖR UNGARIA

GENESYS ORADEA ROMANIA

str. Republicii nr. 35
3700 Oradea
tel: 991/31516

Denumire			Preţ (Valută)	Preţ (Lei)
Calculatoare XT				
GF XTT,,640k,,Mo	(N)	$	311	78486.3
GF XT,360k,640k,20M,Mo		$	693	174890.7
4D XTTbill,,640k,,,Arcn.		$	600	151420.5
Calculatoare AT 286				
GF AT-12,,1M,,Mo		$	366	92366.5
GF AT-12,1.2M,1M,40M,Mo		$	938	236720.7
GF AT-12,1.2M,1M,80M,Mo		$	1366	344734.0
GF AT-16,,1M,,Mo		$	393	99180.4
GF AT-16,1.2M,1M,80M,Mo		$	1394	351800.3
OFF AT-16,,1M,,Mo,Arcn.		$	948	239244.4
AT-16N,,1M,,Mo		$	518	130726.4
AT-16N,1.2M,1M,40M,Mo		$	1090	275080.6
terminale DTK				
DTK AT-16,1M,1.2,40M,Mo		A	10440	223063.6
DTK 386SX,1M,1.2Mf,noMO				
- 16Mhz,VGA,D-Top,noKey		A	9132	195116.6
- 20 Mhz,VGA,D-Top,noKey		A	10361	221375.7
DTK486-33M w 64K C.TOW				
- 4M,noW,FD,VGA,noMo,noKey		A	45077	963126.4
Calculatoare AT-386				
AT386,mainb,33,0M,nC		$	842	212493.4
386,12/25,1.2M,1M,40M,Mo		$	1526	385112.8
386,20/25,1.2M,2M,80M,Mo		$	2139	539814.1
CH.LT3600 16/20,40W,VLCD		$	3036	766187.7
LT.LA30,AT,1M,20W,MLCD		$	1960	494640.3
Monitoare				
12" mono FUJIT,VALTR		';U8.1		
14" mono SUPERTRON		$	110	27760.4
14" EGA SUPERTRON		$	407	102713.6
14" VGA (1024x768) SUPERT		$	414	104480.1
20" VGA monitor+cartela		A	86400	1846043.9
15" MAG allo A4		$	869	219307.4
Adaptoare grafice				
cartela mono		$	25	6309.2
cartela EGA		$	81	20441.8
VGA,8b,256k(800x600)		$	81	20441.8
VGA,16b,512k(1024x768)		$	138	34826.7
Subansamble				
FD 1.2Mb		$	81	20441.8
FD 1.44Mb		$	81	20441.8
FD contr. 2 driv. AT		$	25	6309.2
WD 1003 contr.		$	62	15646.8
controller SCSI		$	138	34826.7
Keyboard 101		$	46	11608.9

Denumire			Preţ (Valută)	Preţ (Lei)
GM6 mouse	(N)	$	21	5299.7
MICROSCAN		$	186	46940.4
COPROCESOR 287-8		$	168	42397.7
COPROCESOR 387-33		$	904	228140.2
MIDI tower+sursa		A	1747	37326.8
Discuri Winchester				
20 MB st225 (65ms)		$	233	58801.6
40 MB st251-1 (28ms)		$	331	83533.6
80 MB st4096 (28ms)		$	669	168833.9
1.2 GB CDC Winc.		$	5505	1389283.1
controller SCSI		$	138	34826.7
Memorii şi accesorii				
1 MB		$	66	16656.3
IC 41256-08		$	7	1766.6
Elemente de reţea				
ARCNET 8bit STAR-BUS		$	61	15394.4
Ethernet NE1000 8bit		$	138	34826.7
Etherlink 3COM 8bit		$	304	76719.7
Novell SFT 2.15		$	5654	1426885.8
Imprimante				
EPSON LX 800		D	235	35301.4
EPSON FX 1050		D	852	127986.4
EPSON LQ 2550		D	2760	414603.9
HP LaserJet III		$	2553	644294.2
HP LaserJet II/P		$	1604	404797.5
STAR NX1001		$	226	57035.1
Cartridge FX1050		$	5	1261.8
Surse neintreruptibile				
UPS 550 W		$	311	78486.3
APC UPS 1200 W		$	1228	309907.3
Altele				
Streamer Wangtech 60 MB IN		$	704	177666.7
Modem INT 1200/SM12H		$	94	23722.5
3.5" DS/DD floppy 3M		$	32	8075.8
3.5" DS/HD floppy 3M		$	40	10094.7
5.25" DS/DD floppy 3M		$	9	2271.3
5.25" DS/HD floppy 3M		$	16	4037.9
Leporello 240x1x12x1700		$	16	4037.9
Leporello 382x2x12x900		$	53	13375.5
Telefoane				
PHIL. D9026 (10 mem)		$	54	13640
CANON FAX 80		D	973	146162.9
SHARP 7850		$	2835	715461.9


OS/2
AIX
XENIX
BS/2
UNIX
386 ix
XENIX
ines
ieee488
WIR UNTERSTÜTZEN
JEDEN STANDARD
OS/2
Sun OS
PC-DOS
MS-