

**CALCULATOARE**

**PERSONALE**

**5 / 91**

**( 8 )**

Revistă lunară editată de Micro ATCI Tîrgu Mureş

**57 Lei**



Memento:

**Funcții Turbo-C**

**MS-DOS 5.0 vs. DR-DOS 5.0**

**Standarde grafice:**

**Dosar: Turbo C++ 1.0**

**Tlga, 8514/A, XGA**

**Paradigma Paradox.**

vă oferă tehnologie americană  
inteligentă europeană  
prețuri orientale la

produse din domeniile:

- tehnică de calcul (hardware și software)
- telematică și birotică

servicii în aceleași domenii:

- lucrări de cercetare / proiectare
- studii de dezvoltare și modernizare
- studii de marketing pe piața internă/externă
- consulting și asistență tehnică
- instalare, școlarizare, service
- sisteme "la cheie"

condiții de livrare: 2 - 4 săptămâni

Cîteva exemple din lista noastră de prețuri:

XT

<< Preț sistem complet 89.000 lei >>

AT-286 & AT-386

<<Preț sistem complet [lei] >>

AT	Winchester capacitate formatată	monitor + interfață		
		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	Lățimea	Nr. ace	Preț [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 Interesați pot obține gratuit lista de prețuri adresându-se la

AbMod Magazin Hardware-Software:

b-dul 22 Decembrie nr. 135

AbMod Sediul central:

str. Moșilor 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

**Redacția:**

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

**Colaboratori externi:**

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

**Tiparul:** tipografia Tîrgu Mureș

**Tipografi:**

Halațiu Mihai,  
Cormoș Mircea

Revista apare lunar.

Preț: 57 lei

**Adresa și telefonul redacției:**

Micro ATCI,  
RO-4300 Tîrgu Mureș,  
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 redacție, cu condiția să nu fi fost publicate și în altă parte. Prin expedierea unui manuscris pe adresa redacției, autorul consimte implicit la publicarea materialului său în cadrul revistei. Onorariul se negociază cu redacția. Materialele nepublicate nu se înapoiază și nu se rețin.

Revista noastră vă oferă spațiu pentru reclamă și publicitate. Pentru amănunte vă rugăm să luați legătura cu redacția.

Cei care doresc să anexeze revistei pliante publicitare tipărite în regie proprie, sînt rugați și ei să se adreseze redacției.

## Fantázia în primejdie

O revistă "ți conferă puteri însemnate"(\*), "dar n-ai voie să le folosești", "niciodată n-ai voie să intervii, indiferent de ce vei vedea, căci începînd din clipa aceasta propria ta părere nu mai are nici o importanță". "Trebuie să lași să se întîmple tot ce se întîmplă. Totul trebuie să aibă aceeași importanță pentru tine, și binele și răul, și ceea ce-i frumos, și ceea ce-i urît, și prostia, și înțelepciunea". "Nu ai voie decît să cauți și să întrebi, nu însă să și judeci după propria ta judecată. Să nu uiți asta niciodată!"

Bine, dar sîntem muritori și sîntem slabi și nu sîntem noi aleșii. Și uităm și greșim!

Nu puține publicații și nu doar cele aflate "de o anumită parte a presei" au avut în ultimele zile titluri de genul: "Este amenințată existența presei independente din România". Noi ce-am mai putea adăuga. Sîntem și noi în derivă în acest triumphi al Bermudelor: hîrtie - spațiu tipografic - difuzare. Cînd toate aceste elemente se scumpesc, în NLL (noua limbă de lemn) "se liberalizează", nu putem alege decît între a scumpi și noi revista, sau a renunța la editarea ei. Ne este rușine pentru fiecare scumpire și nu ne consolăm făcînd comparații de genul: de prețul unei reviste cîte felii de salam (cu sau fără soia) s-ar putea cumpăra. Oricum, "primum vivere".

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

Nu dorim să destabilizăm, așa că dacă nu ați încheiat încă negocierea salariilor vă rugăm să nu citiți articolul "60.000 DM deja pentru începători".

Cît de mult am reușit 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 lățit, și dacă din greșeală intra cineva cu piciorul acolo, dispărea piciorul - sau mîna - sau orice intra acolo. De altfel nu doare, numai că respectivului îi lipsește brusc o bucată din el."

Ing. Romulus Maler

(\*) Michael Ende - "Povestea fără sfîrșit"



# Cuprins

## Știri

- Adio ASCII . . . . . pag.4
- Netware unificat . . . . . pag.4
- Noul Netware dintr-o privire . . . . . pag.4
- ș ț ă î . . . . . pag.5
- ROMORGADATA . . . . . pag.6
- Perestroika . . . . . pag.6
- În curînd: Notepad + Pen . . . . . pag.6

## Magazin

- Salarii: 60.000 DM pentru începători . . . . . pag.7
- Mouse-ul: scurt istoric . . . . . pag.8
- Păcatele scumpe ale conducătorilor  
oficiilor de calcul . . . . . pag.52

## Hardware

- De la 80386 la 80486 . . . . . pag.10
- Vă prezentăm:  
familia Apple Macintosh... . . . . . pag.11
- Joystick . . . . . pag.16

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

- Evaluarea performanțelor  
harddisk-urilor . . . . . pag.22

## Standarde grafice

- 8514/A versus Tiga . . . . . pag.18

Concurența în domeniul adaptoarelor (cartelelor) grafice este puternică. XGA se află în fața ușii. Favoritul branșei a fost pînă de curînd Tiga. Un standard de facto cum este 8514/A mai are vreo șansă ? În ce direcție se îndreaptă dezvoltările ?

- Merită să mizezi pe XGA ? . . . . . pag.20

## Software

- DR-DOS 5.0 vs. MS-DOS 5.0 . . . . . pag.29
- OOP - Modă sau ... ? . . . . . pag.31
- "C" - Cuceritorul . . . . . pag.33

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

- Dosar: Turbo C++ 1.0 . . . . . pag.35
- Paradigma Paradox . . . . . pag.38

Înainte să programați în PAL (Paradox Application Language), trebuie să înțelegeți modelul Paradox.

## Cursuri

- OOP - Partea a treia . . . . . pag.41

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

- Rețele neuronale - partea a treia . . . . . pag.46

Sistemele expert acumulează fapte și reguli. În bazele lor de cunoștințe; pornind de la acestea și folosind strategii, ele raționează.

## Practică

- Parolă variabilă . . . . . pag.49
- Protecția programelor . . . . . pag.50
- Reset la imprimantă . . . . . pag.50
- Leșire în DOS . . . . . pag.50
- Există o dischetă în unitate ? . . . . . pag.51
- Autoverificare contra virușilor . . . . . pag.51
- Pointeri FAR și HUGE în C . . . . . pag.51

## Rubrici

- Caseta redacției . . . . . pag. 1
- Editorial . . . . . pag. 1
- Jargon . . . . . pag. 53
- Din scrisorile primite la redacție . . . . . pag.54
- Mica publicitate . . . . . pag.54

## 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 țări, setul de caractere ASCII impune restricții partenerilor de comunicație. În prezent, cîțiva dintre cei mai importanți 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 același nume. Codul de 16 biți folosit de Unicode asociază fiecărui semn utilizat în prezent un număr și 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 și Coreea, semnele scrise din alfabetele chirilic, ebraic, arab, grec și sanscrit. În afară de acestea, noul set conține simboluri matematice și tehnice, precum și semne diacritice pentru deasupra sau dedesubtul literelor, precum și coduri de comandă (ca de exemplu "avans linie"). Dacă Unicode se impune, munca de ani și ani de zile depusă de ISO (International Standard Organization) vadeveni inutilă. Firme importante (Xerox, Apple, IBM, Microsoft, Sun, Novell, Aldus, Go și Next) și-au declarat sprijinul pentru Unicode.

## Netware unificat

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

În viitor, un potențial cumpărător nu va mai trebui să-și elaboreze cu minuțiozitate decizia de achiziționare a unei rețele 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 și foarte serioasă.

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

Trecerea la Netware v2.2, care va putea fi achiziționat în versiuni pentru 5, 10, 50 și 100 de utilizatori, va ușura nu numai viața utilizatorilor, ci și a producătorilor de rețele. Căci organizarea service-ului și întreținerii pentru un singur produs sînt mai ieftine și mai sigure pentru un singur produs decît pentru patru.

Astfel că despărțirea 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 și pentru vînzători și 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 bucăți, ELS I, ELS II, SFT și Advanced Netware reprezentau circa 90% din vînzări (celelalte 10%: Netware 386).

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

## Noul Netware dintr-o privire

Odată cu unificarea liniei sale de produse pentru 286, Novell a introdus cîteva îmbunătățiri. 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 cîteva parametri și face apoi singur configurarea. Procesul de instalare (numit de Novell "intuitiv") e sprijinit de help-uri detaliate, apelabile oricînd cu F1.
- manuale mult îmbunătățite. Analog instalării mult simplificate, și manualele s-au subțiat 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 documentație, ceea ce în marea majoritate a cazurilor ar trebui să facă superflue căutările în manuale.
- structură îmbunătățită 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 și Tolkentalk.
- opțiuni îmbunătățite pentru imprimantă. Serverul de imprimantă este conținut în Netware 2.2, ca VAP și Exe. S-au făcut îmbunătățiri, s-au introdus comenzi, etc.
- suport extins pentru stații de lucru. În această categorie intră posibilitatea de a face automat un "Workstation Update"; funcții noi la suportul de workstation pentru DOS, Macintosh și OS/2.
- Suplimentar în Netware v 2.2 sînt integrate drivere de 3COM, s-au făcut modificări în "trustee rights" și au fost operate îmbunătățiri în utilitare ca FConsole, Login, Map și Filer.

## ș ț ă î

sau

## Computerele și Limba Română

Întrebarea dacă și la noi calculatoarele vor ajunge vreodată să joace vreun rol în viața și activitatea noastră nu mai are sens. Este clar că avem nevoie de ele. Ce ne facem însă că nu prea știu să vorbească românește, iar când o fac, vorbesc atît de peltic, încît nici nu știm să rîdem sau să plîngem. Evident, pînă acum s-a recurs la multe soluții, care mai de care mai ingenioasă, mai simplă sau mai complicată, și mai ales mai

"incompatibilă". Fără a avea pretenția de a fi găsit cea mai bună soluție, în urma studiilor noastre s-au conturat cîteva sugestii-rezolvări.

Pentru ca un calculator să poată fi folosit și în limba română, este necesar înainte de toate să codificăm cumva și literele românești în setul de caractere cunoscut de calculator. Este esențială stabilirea de comun acord și acceptarea la nivel național a unui asemenea 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 românești nu au fost prevăzute în nici unul din seturile disponibile. Două sînt seturile de interes pentru noi: setul obișnuit (437) și setul internațional extins (850, care sacrifică o parte din caracterele semigrafice în favoarea unor litere speciale). Deoarece introducerea literelor românești în tabelă nu se poate face decît prin renunțarea la alte caractere, este foarte importantă alegerea judicioasă a acestora. Propunerea noastră este următoarea:

(0x83; 131)	ă
(0x86; 134)	â
(0x8C; 140)	î
(0x8F; 143)	Ă
(0x91; 145)	Î

(0x92; 146)	Ț
(0xA6; 166)	Ș
(0xA7; 167)	Ş
(0xA8; 168)	Â
(0xAD; 173)	Î

Această amplasare conduce doar la pierderi minime de caractere (cîteva litere din alfabetul țărilor nordice și două semne de punctuație din limba spaniolă). În plus această substituție este posibilă în ambele tabele, atît cea normală cît și cea internațională, conducînd la apariția a două seturi de caractere: românesc și românesc extins.

A doua mare problemă este claviatura. Fiecare țară europeană a reinventat, cu mai mult sau mai puțin succes, o altă așezare a tastelor (vezi claviatura germană foarte dificil de utilizat pentru orice altceva decît editare de texte). Dificultatea proiectării unei claviaturi românești provine și din faptul că două dintre cerințe sînt contradictorii: una impune ca literele românești să fie amplasate în dreapta, în continuarea celorlalte litere (ca la mașina de scris: <K L Ș Ț> <O P Ă Î>) iar cealaltă cere ca semnele de punctuație, parantezele drepte și acoladele să rămînă în aceeași poziție. Rezultă clar că o claviatură românească tebuie să aibă două moduri comutabile funcție de activitatea preponderentă: dezvoltare de programe sau editare de texte. În mod normal, claviatura va genera aceleași coduri ca o claviatură americană, cu mențiunea că folosite împreună cu Alt sau Alt+Shift, tastele din dreapta vor genera codurile literelor românești. În modul editare de texte, funcția acestor cinci taste se inversează, generînd direct <ș ț ă î>, semnele de punctuație fiind generate doar apăsînd simultan Alt sau Alt+Shift.

Afișarea în ecran, pentru modul alfanumeric obișnuit, nu este o problemă, plăcile EGA și VGA permițînd definirea completă a setului de caractere.

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

După cum am spus și la început, aceasta nu se vrea soluția definitivă a problemei literelor românești pe calculatoare. Mai rămîn destule pete albe în dosarul acestei probleme (ca de exemplu modificarea numărărilor 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 discuții, cu convingerea că în cele din urmă hotărîrea ce se va adopta se va respecta la nivel național și va satisface cele mai exigente preferințe.

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

Pentru orice lămuriri, sugestii și propuneri, ne puteți contacta la următoarea adresă:

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

## ROMORGADATA

Desfășurată între 7.-10.05.1991, în Complexul Expozițional din Piața Științei (pardon, Presei Libere), expoziția specializată organizată de ROM-EXPO S.A. și I.E.G. Salingen a fost, se pare, un succes. Cel puțin pe partea care ne interesează - PC-uri & Co.

Au fost prezentate multe exponate, majoritatea de vîrf, și s-au putut face contracte pe loc. De la imprimante laser HP Laser Jet IIID și 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, sound-blaster sau laptop-uri Toshiba 486 - cam toată panopia PC-urilor a fost prezentă. Nu în ultimul rînd Borland a stîrnit interes, atît prin "programul de amnistie" anunțat cît și prin prețurile în general accesibile pentru marea majoritate a produselor. Și au putut fi văzute - și cumpărate - ultimele noutăți; 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 acțiune pentru EGA/VGA - primul joc shareware din URSS! Un joc pasionant, cu multă acțiune, efecte sonore comice și o grafică foarte bună. Jocul are 25 de nivele de joc și necesită din partea dumneavoastră capacitate de reacție rapidă și nervi tari. Scopul jocului este de a depăși cu micuța broscuță verde a perestroicii (GORBI) multe obstacole mici, mobile și de a aduna puncte. Cu cît crește nivelul de joc, crește și gradul de dificultate și dintr-o dată apar broaște uriașe, urîte și roșii (STALINOS) care vă vor viața. Perestroika, împreună cu Commander Keen și 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.

## În curînd: Notepad + Pen

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

Revoluționară este (va fi) interfața 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 poziției actuale a creionului în momentul cînd se apasă cu acesta asupra tabletei. Drumul parcurs de creion pe tabletă este afișat - 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 și mouse.

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



Creionul electronic ca dispozitiv de intrare.

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

# 60.000 DM deja pentru începători

*Salariile din industria calculatoarelor cresc vertiginos. Absolvenții unei școli 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 consultanță 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 și 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 crește și în continuare în favoarea angajaților. Conducătorii afacerilor caselor de hardware și software PC au obținut, conform așteptărilor, salarii de vis. Salariile lor au crescut față de anul anterior

cu 8000 DM ajungînd la 262000 DM anual.

## Cîștiguri mai mari pentru superiori

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

Șansa unui începător de a cîștiga 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 proiectanți, se pot

aștepta la un salariu anual între 60000 DM și 65000 DM, dar acumînd experiență în domeniu, în cîțiva ani, pot să obțină cu 20000 - 30000 DM mai mult. Dacă în decursul anilor începătorul devine șef de secție, și va avea deci o funcție de răspundere, atunci saltul va putea fi mai mare.

Un inginer proiectant care comitent este și șeful unei secții cu cel puțin 6 ingineri, va putea cîștiga cel puțin dublul salariului său de începător.

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

## Asigurări sociale

Și în cazuri de îmbolnăviri angajații din industria electronică sînt asigurați mai bine decît prevede legea. 88% dintre întreprinderi garantează plățile 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 relațiile întreprinderii cu exteriorul sînt cuprinși în această reglementare; în plus aceștia sînt asigurați pentru caz de deces sau de invaliditate, cel mai adesea cu polițe de asigurări pe viață, care se ridică la 100000 DM și care, de regulă, sînt valabile 24 ore/zi și nu doar pe timpul serviciului.

Concediul anual variază între 29 și 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
Șef echipă proiectare cu 2 sau mai mulți tehnicieni	95-119
Șef proiectare cu mai mult de 6 ingineri	123-146
Inginer de aplicații, pînă la 3 ani vechime	77-93
Inginer de aplicații, mai mult de 3 ani vechime	98-120
Inginer producție-marketing, pînă la 3 ani vechime	87-103
Inginer producție-marketing, mai mult de 3 ani vechime	111-129
Director producție-marketing (director tehnic)	125-143
Servicii externe (muncă de teren)	
Inginer expert într-un domeniu, pînă la 3 ani experiență în producție	76-94
Inginer expert într-un domeniu, mai mult de 3 ani experiență în producție	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
Șef comercial zonal, șef al unui inginer	137-160
Șef comercial zonal, șef a cel puțin 3 ingineri	167-224
Director	238-304

# Oameni și șoareci



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

## Șoarecii 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 încât comparându-l cu alte instrumente de introducere, a câștigat 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 generații de PC-uri s-au lăsat influențate de ideile lui Engelbart, începând cu Star-ul firmei Xerox, prin Lisa lui Apple și pînă la Macintosh-urile firmei Apple. Dar Engelbart nici nu și-ar fi putut închipui ce avînt va lua invenția sa.

Mouse-ul inventat de Engelbart era un simplu instrument de introducere, analogic, care transmitea soft-ului cîte un semnal la fiecare mișcare a mouse-ului, influențînd mișcarea cursorului. În interiorul acestui corp de lemn se aflau două roțițe metalice, legate între ele prin două rezistențe 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 "șoricelului" 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 puțin de 100 de exemplare, drumul pentru dezvoltarea calculatoarelor personale și al mouse-urilor a fost deschis.

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

După ce mouse-ul firmei Xerox a trezit tot mai multă atenție, 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 același timp Bill Gates, Paul Allen și Raleigh Roark discutau extinderea paletei de produse Microsoft cu un produs hardware adecvat. Astfel

s-a pus baza primului mouse Microsoft.

## Dintr-un boț de lut

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

## Strămoșul din lut/lemn

Raleigh Roark își aduce aminte de prezentarea acestui mouse: "Unii dintre noi stăteau ceasuri în șir la masa de conferințe, ocupați cu mișcarea înainte și înapoi a acelei grămăjoare de lut, pentru a afla dacă ne place sentimentul pe care ni-l dă această mișcare. Dar nici unul dintre noi nu era pe deplin mulțumit, astfel încît i-am schimbat înfățișarea și dimensiunile după imaginația 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 însoțit de Kay Nishi, care pe atunci era vicepreședintele Microsoft și președinte al comisiei ASCII din Japonia. Nishi și Roark s-au întîlnit acolo cu producători de aparatură electronică pentru a discuta ideile lor. Discuțiile s-au sfîrșit

repede și 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 povestește: "s-a discutat mult, de ce acest lucru ar fi imposibil, important era doar că nu se putea. Până când la un moment dat s-a făcut o liniște deplină și inginerul șef ne-a comunicat că se va retrage timp de o oră, după care ne va da o soluție. Într-adevăr inginerii au revenit cu o propunere de carcasă care avea capacitatea de a funcționa, iar câteva luni mai târziu Microsoft poseda primul mouse".

### Prima generație

"Crearea mouse-ului serial era pentru mine o problemă captivantă, o deschidere a concepțiilor, nerealizată de nimeni până atunci", spune Raleigh Roark conducătorul secției 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 având două taste, sprijinindu-se pe o sferă de oțel și câteva role, care înregistrau mișcările mouse-ului pe orice suprafață. "Îndemânarea" lui era dictată de o cartelă de 8 biți, care era echipată cu un cip Intel 8255, ca interfață periferică programabilă, și cu câteva cipuri auxiliare. Acest tip de mouse avea avantajul că nu consuma multă energie electrică și deci nu avea nevoie de o sursă distinctă de alimentare.

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

Odată cu apariția, în anul 1983, a sistemului de operare MS-DOS

2.0, mouse-ul a avut un cuvânt important de spus. MS-DOS era echipat cu un așa-zis driver de periferic instalabil la dorință, ceea ce simplifica foarte mult configurația oricărui calculator echipat cu sistemul de operare DOS și cu un mouse.

În 1985, la apariția AT-urilor IBM, odată cu creșterea numărului de cartele grafice de înaltă rezoluție s-a ținut cont și de cele două noi versiuni de drivere.

### Generația a doua

Mouse-ul firmei Microsoft cu o rezoluție de 200 puncte per inch a schimbat modul de utilizare al primului mouse. Dublarea rezoluției a avut ca efect faptul că utilizatorul nu mai trebuia să plimbe mouse-ul pe întreaga masă de scris pentru a mișca 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 și bila îmbrăcată în cauciuc în locul celei îmbrăcate în oțel. Adevărata diferență însă s-a simțit, nu s-a văzut. Cu dublarea rezoluției la 200 dpi (dot per inch - puncte pe inch), mouse-ul putea fi manevrat mult mai ușor, necesita o rază de acțiune mult mai mică pentru deplasări (aprox. 10-12 cm); cele mai multe probleme puteau fi rezolvate rapid și cu economie de mișcare. În comparație cu acesta modelul anterior părea greoi și grosolan, mai ales datorită razei mai mari de acțiune (aprox. 20-25 cm).

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

### Generația a treia

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

Mouse-ul celei de-a treia generații 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. Câteva modificări esențiale au dus la o controlabilitate mai ușoară; de ex., poziționarea bilei (care înainte se afla în spate) în fața mouse-ului și mărirea tastei din stînga față de cea din dreapta.

### Generația a patra

Actualul mouse, prezentat în 1989, de culoare albă și cu o rezoluție de 400 dpi aparține generației a patra. Aspectul este similar cu cel din generația a treia, dar tehnologic se află pe un plan net superior. Importante sînt următoarele aspecte: rezoluția net îmbunătățită pe de o parte, iar pe de altă parte sensibilitatea și 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 apariția 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 cărții "The Apple Macintosh Book", ediția a treia, Microsoft Press.

Astăzi deja un soft cu suprafață utilizator grafică este mai degrabă o regulă decît o excepție, și în paralel cu această dezvoltare acceptarea mouse-ului se extinde continuu:

Bun venit în era șoarecilor !

(I. M.)

## De la 80386 la 80486

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

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

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

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

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

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

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

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

Spațiul 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 și Windows 3.0. Această suprafață utilizator rezervă pe harddisk o așa-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 și memoria virtuală care stă la dispoziția programului.

Pentru a putea utiliza un 80386 la viteza sa maximă de lucru este necesar ca și periferia să fie tot la fel de rapidă. Din această cauză se încearcă tot felul de trucuri, pentru a intercala cît mai puține cicluri de așteptare (waitstates). Unul dintre aceste trucuri este așa-numitul "Pipelining", în cazul căruia unitatea de management a memoriei

folosește semitacturile pentru a determina în timp util adresele care urmează. De obicei procesorul află adresa și definiția 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 și 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 și modul de administrare. În această memorie se scriu datele care sînt necesare mai la urmă. Deoarece cel mai adesea un program nu "țopăie 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 funcție și de modul de administrare, este între 89% și 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 și o lățime de 128 biți. 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 unității centrale (CPU - Central Processor Unit) se folosește testul Dhrystone, iar pentru măsurarea vitezei cu un coprocesor se folosește testul Whetstone. Ambele sînt teste ("Benchmarks") standardizate și conțin un spectru prestabil de comenzi de procesare.

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

(I. M.)

## Să-l muști ...

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

Faptul că începătorii în ale calculatoarelor își încep cariera lor de utilizatori nu cu puțină teamă, la 10 ani de la introducerea calculatoarelor IBM-PC (1982), nu este fără motiv. Căci ca și pînă acum cei mai mulți utilizatori sînt nevoiți să utilizeze sistemul de operare MS-DOS și 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 revoluționar, și anume Macintosh. Un posesor de Mac nu trebuie să facă risipă nici de acrobație mentală și nici de răbdare pentru a descoperi secretele utilizării calculatorului, ci se poate concentra imediat după pornirea lui asupra muncii propriu-zise și chiar și cel mai neștiutor dintre toți poate lucra după 1/2 oră cu un Macintosh.

Ceea ce face Macintosh-ul atât de atractiv, (Macintosh este și numele unui sort de mere), este sistemul său grafic de manipulare și anume "Finder". Indiferent că se dorește lansarea în execuție a unui program, ștergerea sau copierea unui fișier, totul se rezolvă printr-un simplu clic de mouse și prin simboluri ușor de înțeles. Finder permite o utilizare nu numai ușoară ci și unitară. Toate programele pot fi deservite după aceeași schemă. Dacă utilizatorul cunoaște un program, atunci le cunoaște pe toate.

Un alt atu: toate calculatoarele Macintosh stăpînesc așa-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 fi rulate simultan.

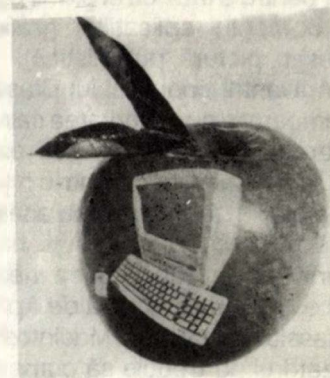
Avantajul: cînd utilizatorul scrie, de ex., o scrisoare unui client și are nevoie de informații despre partenerul de afaceri, poate comuta rapid într-un program bază de date, poate obține datele despre client și se poate întoarce în programul de prelucrare de texte - chiar și schimbul de date între aplicații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

### Pentru începători: Macintosh-Classic

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

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

Noul Apple înlocuiește Mac-urile compacte Macintosh SE și Macin-



tosh Plus. În exterior li se aseamănă amîndorura, tehnic însă este urmașul lui SE: Apple construiește ambele calculatoare constitutiv din aceleași componente. Totuși 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 acționat de un procesor Motorola 68000 cu un tact de 8 MHz. Puterea de calcul a lui Classic este suficientă pentru aplicații simple, pentru calcule complexe nu ajunge însă (cum ar fi de ex. pentru aplicații grafice profesionale sau pentru gestiunea bazelor de date mari).

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

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

### Numele de un milion de dolari

Atunci cînd șeful 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 însoarta

Florida, pentru a-și putea boteza noul produs "Classic". Ironie a sorții: Classic a fost (și este în SUA) și numele unui sort de Coca-Cola. Șeful firmei Apple John Sculley și-a început cariera managerială la Pepsi-Cola și și-a făcut un bun nume în ramură ca expert în marketing.

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

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

Alte posibilități de conectare: micuțul Apple are (din nou ca toate Mac-urile) o interfață SCSI (Small Computers System Interface - un standard de interfațare), la care pot fi conectate simultan pînă la șase periferice (de ex. un harddisk suplimentar și unități 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 și are dimensiuni reduse, avantajul: spre deosebire de majoritatea calculatoarelor MS-DOS el poate fi așezat pe masa de scris și mai rămîne loc și pentru acte și cărți. Un alt avantaj: deoarece este atît de mic și de ușor, și întrucît are inte-

grat în carcasă un mîner, poate fi foarte ușor transportat.

Dezavantaj: deoarece monitorul monocrom este înglobat în carcasă, trebuie să te descurci cu o diagonală a ecranului de doar 9" (23 cm) - pentru comparație: diagonală unui monitor normal măsoară cel puțin 12" (30 cm). Consolarea: cu toate că monitorul lui Classic este mic, textele și imaginile sînt afișate cu rezoluția 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 capacitățile (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 (preț: cca. 600 DM). Această cartelă mai conține și socluri pentru cipuri de memorie, așa-numite SIMM (Single In line Memory Module) cu ajutorul cărora memoria poate fi extinsă pînă la 4 MBytes. (preț pe MByte: cca. 150 DM).

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

## Calculatorul: Macintosh SE/30

El este tot atît de mic ca și modelul Classic, dar în interiorul lui mocnește puterea unui uriaș: Macintosh SE/30 (preț: începînd de la 7600 DM) lucrează cu rapidul procesor 68030 (momentan cel mai rapid procesor Motorola aflat pe piață) cu un tact de 16 MHz, căreia i se adaugă în cazul calculului deosebit de complicate și 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 și un slot (așa 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 și de o reprezentare color. Se recomandă pentru:

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

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

### Operare ușoară: Finder

Calculatoarele Macintosh sînt cele mai prietenoase calculatoare din lume în privința operării și acest lucru se datorează suprafeței grafice Finder (se pronunță Fainder). După un test al Institutului Fraunhofer din Stuttgart, din punct de vedere al operării calculatoarele compatibile IBM nu pot da clasă unui Mac nici chiar cu suprafața utilizator Windows.

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

Chiar dacă nu este la fel de ușor să folosești un IBM PC cum este de ușor să folosești un Mac, IBM PC

prezintă totuși un avantaj uriaș: 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 aplicație) 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 ieșire din impas există doar pentru posesorii de Mac-uri din puternica serie II (IIsi, IIfx și IIfx) și de SE/30. Aceștia pot utiliza programul de emulare Soft-PC (900 DM) care emulează un PC/XT și 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 și Japonia. Motivul: în ambele țări birourile uriașe sînt divizate în celule de lucru minuscule. Fiecare angajat dispune de un minibirou cu o suprafață de cca. doi pe doi metri și de o micuță masă de scris. Un PC normal nu prea se potrivește înăuntru - în asemenea locuri este necesar un SE/30 care ocupă cel mult suprafața unei mape de scris DIN A4 și care dispune totuși de o putere de calcul maximală.

### Curcubeu ieftin: Macintosh LC

Cu Macintosh LC Apple oferă în sfîrșit cumpărătorilor un model color plătit ("LC" semnifică "Low cost Color", în românește "color ieftin", dar ieftin în sensul că merită banii). Prețul 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" (rezoluție 384x512) LC poate reprezenta imagini în 256 de culori, cu un monitor color de 13" (rezoluție 480x640) poate reprezenta imagini în 16 culori.

Mai multe culori pot fi obținute doar prin adăugarea unei plăci de extensie suplimentare, o așa-nu-

mită cartelă "Video-RAM", care conține mai multă memorie pentru grafice; cu ajutorul ei pot fi obținute 32000 de culori cu un monitor de 12" și 256 de culori cu unul de 13".

În afară de aceasta un LC mai conține un procesor 68020 (tact 16 MHz), 2 MBytes RAM (maxim 10 MBytes) și un harddisk de 40 MBytes.

Cu această dotare el poate prelucra texte, fișiere și 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) și astfel pot fi digitalizate vorbire, muzică și zgomote, pot fi prelucrate și apoi redade. De asemenea textele și datele pot fi însoțite de comentarii vorbite, pentru a da de exemplu indicații colegilor în timpul lucrului cu un program.

Deoarece fișierele de sunete necesită foarte mult spațiu, 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.



Cu facilitățile sale color limitate și viteza sa medie de calcul Macintosh LC este un calculator indicat pentru școală, universitate și birou. Se recomandă pentru:

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

### A începe pentru a urca: Macintosh IIsi

Ele oferă un tempo de calcul ridicat și o grafică excelentă - așa-numita serie constructivă modulară (Macintosh IIsi, Ilic și IIfx). Aceste Mac-uri sînt modulare deoarece permit utilizatorului să și le adapteze după necesități 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 piață de Apple în octombrie.

IIsi lucrează ca și 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 și 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 rezoluție de 384x512 pixeli), 2 MBytes RAM și un harddisk de 40 MBytes. IIsi are un singur slot liber posibilitățile de extensie fiind astfel limitate. Totuși utilizatorul poate utiliza fie plăci de extensie specifice pentru SE/30 fie cartele "Nu-bus". Trucul în cazul plăcilor de extensie: ele conțin în plus un coprocesor 68882. IIsi este ca o combinație între un Mac LC și un SE/30:

#### Macintosh Portable

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

La fel de practic: cu puterea acumulatorului său, Mac Portable este economic ca un scoțian - el poate funcționa 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).

## Apple Story

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

Într-un garaj au construit primul calculator personal din istorie, Apple I, care a avut un succes deosebit. Cu câștigul obținut Jobs și Wozniak au finanțat la sfârșitul 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 urmașul lui Apple III, cît și precursora Macintosh-urilor Lisa, n-au avut priză la public și au

adus firma Apple în pragul falimentului. Abia cu Macintosh norocul s-a reîntors.

Șeful firmei Apple Steve Jobs (azi 35 de ani și șeful firmei Next care face concurență lui Apple) și-a propus în momentul realizării modelului Macintosh să realizeze un calculator cu putere mare de calcul, confortabil în operare, fără compromisuri și 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 și 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) și meniuri. Modelul său: o suprafață utilizator pe care Jobs și "pirații" (așa își spuneau proiectanții înșiși) au văzut-o în Xerox PARC (PARC: Palo Alto Research Center).

Datorită faptului că permitea o operare prietenoasă și că era fiabilă această suprafață utilizator a transformat Macintosh-ul într-o legendă vie. Totuși ea n-a putut afecta PC-urile IBM și ale altor concurenți printre altele datorită politicii de prețuri ridicate practicate 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 și-a dat

repede seama de faptul că Apple se afla în criză și datorită stilului incoerent de conducere al lui Jobs și l-a deposedat de putere. Cătrănit Jobs a părăsit firma Apple și a înființat o nouă firmă, Next Computer care produce stații de lucru (workstation).

John Sculley a renunțat la politica prețurilor ridicate abia în momentul în care în ultimul an procentul din piață și câștigurile firmei Apple au început din nou să scadă rapid ca urmare a scăderilor de prețuri la calculatoarele compatibile IBM și a apariției suprafeței utilizator Windows. În plus Apple a lansat pe piață trei noi Mac-uri ieftine (Classic, LC, IIsi) și a scăzut și prețurile la modelele rămase.

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

Apple s-a hotărît să riposteze concurenței cu un nou Macintosh II cu un procesor Motorola 68040, cu tactul de 25 MHz, de îndată ce acest procesor va fi la dispoziție în cantități suficiente de mari.

Se mai vorbește și 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 experții japonezi în materie de Laptop-uri Sony și Toshiba pentru realizarea unui Macintosh Laptop care ar trebui să apară pînă la sfârșitul acestui an. Acesta nu va fi tot la fel de puternic ca și Portable, dar va fi mult mai ușor.



Părinții  
Macintosh-ului:  
Steve Jobs (sus)  
și Stephen  
Wozniak (mijloc).  
Jos: șeful firmei  
Apple John  
Sculley

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

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

### Artistul: Macintosh IICI

Fratele mijlociu din seria modulară Mac se numește Macintosh IICI (preț: 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 folosește un coprocesor 68882 și are o memorie de 5 MBytes.

Și 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 toți cei care doresc să facă publicistică profesională. Programe de grafică puternice memorează documentele paginate pe dischete astfel încât acestea pot fi preluate și trimise direct în tipografie.

Condiția pentru aceasta este o putere de calcul foarte mare. Atunci când utilizatorul dorește imprimarea unei imagini color, calculatorul trebuie să preia așa-numita "separare a culorilor", pentru fiecare pixel al imaginii trebuind transmise culorile constitutive.

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

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

Data fiind puterea sa de calcul el poate fi utilizat, ca și SE/30, ca server într-o rețea.

### Extraterestrul: Macintosh IIFX

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



Cu un adaptor grafic de 24 de biți poate reprezenta simultan peste 16 milioane de culori. Macintosh IIFX este indicat pentru toate aplicațiile pentru care era recomandat și predecesorul său; în plus el este singurul Mac (și probabil singurul calculator personal, incluzând chiar și calculatoarele compatibile IBM PC 486) care este suficient de rapid pentru următoarele categorii de aplicații:

- desen tehnic pentru constructori, designeri și arhitecți (cuvânt cheie: CAD = Computer Aided Design, proiectare cu ajutorul calculatorului);
- animație
- aplicații 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
Preț (DM)	3000	7650	4900	10500	15150	22900	9000
<b>Procesor Motorola</b>							
Procesor, Tact (MHz)	68000, 8	68030, 16	68020, 16	68030, 20	68030, 25	68030, 40	68HC00, 16
Coprocesor	-	68882	-	-	68882	68882	-
<b>Memorie</b>							
Standard/Maxim (MBytes)	2/4	2/32	2/10	2/17	5/32	4/32	2/9
<b>Lector de dischete</b>							
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
<b>Harddisk</b>							
Capacitate (MBytes)	40	40	40	40	40	80	40
<b>Sloturi</b>							
Tip "Processor Direct Slot"	-	1	1	(1)	-	-	1
Tip "Nubus"	-	-	-	(1)	3	6 (5 libere)	-
<b>Interfețe</b>							
Imprimantă	*	*	*	*	*	*	*
SCSI	*	*	*	*	*	*	*
Modem	*	*	*	*	*	*	*
Lector de dischete extern	*	*	-	*	*	-	*
Intrare/ieșire audio	nu/mono	nu/stereo	da/mono	da/stereo	nu/stereo	nu/stereo	mono
<b>Monitor</b>							
Diagonală	9"	9"	12"	13"	13"	13"	9,8"
Tip	a/n, integrat	a/n integrat	color	color	color	color	a/n-LCD, integrat
Rezoluție	512x342	512x342	512x384	640x480	640x480	640x480	640x400
Nr. nuanțelor de gri, resp. al culorilor	2	2	256	256	256	>16,7 *10 <sup>6</sup>	2
Adaptor video integrat	*	*	*	*	*	*	*

# Joystick

*Cu calculatorul te poți și juca, dar pentru aceasta pentru anumite jocuri ai nevoie de un joystick. În cele ce urmează o să încercăm să vă explicăm ce este și cum funcționează un joystick.*

Jocurile pe calculator sînt un mod plăcut de petrecere a timpului liber. Pentru a putea conduce vehicule spațiale pe ecran, a simula pilotarea unui avion sau o cursă de mașini, nu este necesar doar un anumit program ci și o configurație corespunzătoare. Un accesoriu important este joystick-ul, care transformă mișcările mîinii în informații pentru calculator. Prin intermediul lui jucătorul ia parte la acțiunea de pe ecranul calculatorului.

Cea mai importantă componentă a joystick-ului este **maneta**. Ea este mișcată cu mîna pentru a da calculatorului comenzile de direcționare. Dacă se dorește mișcarea figurii proprii de pe ecran în sus, se împinge maneta în sus. Doar prin simple împingeri se pot obține în total opt sensuri de deplasare: sus, jos, stînga, dreapta, ca și direcțiile diagonale (diagonal dreapta sus etc.).

Atunci cînd se mișcă maneta într-o anumită direcție, mișcarea este înregistrată de cîteva componente electronice care se află în interiorul joystick-ului. În cel mai simplu caz cu maneta se apasă tasta unui așa-numit **microînterupător**. Ca și în cazul unui **înterupător** pentru lumină, se închide un circuit și 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 direcție nu au fost apăsată, dacă valoarea

este "unu", înseamnă că o anumită tastă a fost apăsată. Majoritatea joystick-urilor dispun pentru sesizarea direcției de deplasare de patru microînterupătoare de la care pleacă cîte un fir la calculator. Cu ajutorul lor calculatorul poate afla și dacă maneta a fost împinsă, de ex., diagonal stînga jos. În acest caz vor fi pe "unu" simultan valorile pentru "jos" și "stînga".

Joystick-urile cu microînterupătoare sînt foarte convenabile (performanțe/preț) și din acest motiv sînt utilizate de o mulțime de jocuri pentru HC-uri (Home Computer). Ele pot indica doar direcțiile de deplasare, dar nu și cît de departe a fost împinsă maneta într-o anumită direcție. Pentru aceasta sînt necesare așa-numitele joystick-uri analogice, care știu deosebi și situații intermediare cum ar fi "50% dreapta" sau "95% dreapta". În timp ce celelalte joystick-uri asemenea unui **înterupător** pentru lumină nu cunosc decît pozițiile "închis" și "deschis", modelele analogice acționează 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 reacție, se poate conduce mai exact. Joystick-urile cu microînterupătoare sînt denumite, pentru deosebire, "joystick-uri digitale".

Joystick-urile digitale utilizează în locul microînterupătoarelor două "potențiometre". Cu cît este deplasat cursorul unui potențiomtru, prin intermediul manetei, mai mult într-o anumită direcție, cu atît mai mare sau mică este doza de curent care este transmisă prin intermediul cablului de legătură la așa-numitul "convertor analog-digital", care transformă impulsurile de curent în valori digitale. În acest caz nu se face însă distincție doar

între "zero" și "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". Același lucru este valabil și pentru cel de-al doilea potențiomtru, care semnaleză pozițiile de sus pînă jos. Atunci cînd maneta este la mijloc ambele potențiometre trebuie să indice valoarea "31".

Joystick-urile analogice sînt utilizate mai ales pentru PC-uri. Acestea au nevoie în plus de un așa-numit "port pentru jocuri" (Preț: 50 DM plus 50 DM pentru joystick; joystick-urile digitale costă între 10 și 40 DM). Acesta este de fapt o placă de extensie care conține o interfață 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 declanșatoare** (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 opțiune dintr-un meniu.

Joystick-urile bune au și 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 și **picioarele de sprîjin**, 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 direcție jucătorul controlează acțiunea de pe ecran

### 2 Tastă declanșatoare

În funcție de joc cu ajutorul tastei declanșatoare se trage săgeți sau se activează un meniu.

### 3 Comutator

Mecanica funcționează ca și în cazul unui buton de sonerie - dacă se apasă pe comutator se transmite un semnal la calculator

### 4 Microîntrerupător

Cele patru întrerupătoare sunt acționate de manetă atunci când aceasta este împinsă într-o anumită direcție (sus, jos, dreapta sau stânga)

### 5 Suport

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

### 6 Microîntrerupător de "foc automat"

După acționarea manetei electronica simulează de mai multe ori pe secundă apăsarea tastei declanșatoare

## 8514/A versus Tiga

*Concurența în domeniul adaptoarelor (cartelelor) grafice este puternică. XGA bate la ușă. Favoritul branșei a fost pînă de curînd Tiga. Un standard de facto cum este 8514/A mai are vreo șansă ? În ce direcție se îndreaptă dezvoltările ?*

Standardul grafic 8514/A al lui IBM creat în urmă cu cîțiva ani, mult timp a dormit asemenea frumoasei din pădurea adormită. Aceasta s-a întîmplat mai puțin datorită standardului însuși, ci mai mult a ținut de ceea ce IBM a făcut din el. Deoarece puțin știi cu adevărat că 8514/A permite mai mult decît o rezoluție de 1024 x 768 în modul înjumătățire a imaginii (interlaced). Din contră, 8514/A este o interfață grafică completă și nu are nimic de-a face cu imaginea înjumătățită sau întreagă, respectiv interlaced sau non-interlaced. În specificațiile originale IBM este cuprinsă chiar și o rezoluție de 1280 x 1024 puncte. În mod regretabil IBM nu a făcut încă cunoscut cum vrea să utilizeze registrele corespunzătoare.

IBM a fost foarte reținut în ceea ce privește informațiile despre 8514/A, ceea ce a avut ca efect o acceptanță relativ restrînsă pe piață. Numai interfața pur soft, 8514/A-AI (AI = Application Interface), a fost publicată de IBM. Modul de adresare și funcțiunile registrelor hard ale lui 8514/A au fost ținute de IBM cu strictețe sub cheie. Direcționarea pe care IBM dorea s-o dea era clară: Big Blue dorea să determine, în sfîrșit, casele de soft să nu mai programeze la nivel hardware. O altă piedică importantă pentru răspîndirea lui 8514/A ținea de limitările impuse de IBM arhitecturii microcanal a seriei PS/2. S-a schimbat cumva situația ?

Cu Tiga (Texas Instruments Graphics Architecture) s-a realizat un nou standard pentru adaptoare

grafice de rezoluție ridicată, care să lucreze independent de adaptorul grafic propriu zis. Euforia de la început s-a disipat însă repede. Tiga a fost și este o interfață pur soft, care nu permite accese directe la hard. Avantajul vitezelor ridicate al procesoarelor grafice pe 32 de biți de la Texas Instruments (TI-34010 și TI-34020) se pierde din această cauză.

Acest lucru a fost observat de către cîțiva dintre producătorii de adaptoare grafice cu procesoare TI, și aceștia nu mai oferă chiar nici o interfață 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 funcțiile pan și zoom sau alte caracteristici, care de ex., atît timp cît există drivere speciale pentru soft-ul utilizat, oferă o viteză mai ridicată de execuție a programelor sau alte avantaje. Urmarea este că pentru fiecare adaptor și pentru fiecare pachet soft, la care se dorește să se umble la viteză, trebuie scris un driver special, ceea ce implică un efort serios. Și tempoul contează la majoritatea programelor grafice.

Ce efect are acest lucru în practică, pot povesti deja mulți utilizatori de adaptoare grafice de înaltă rezoluție: se instalează o nouă versiune a unui soft și se observă cu regret că programul nu funcționează pe PC-ul disponibil. Și pentru aceasta nici nu trebuie ca hard-ul să fie vechi. Cine deținea 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 lovește de aceleași probleme de care se lovise și cu un an înainte la actualizarea versiunii Windows 2.03 cu mai noua versiune 2.1. Trebuiau așteptate drivere noi, și 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 fabricație ? Atunci cel mai adesea și se spune: "Ne pare rău dar nu mai producem acest adaptor" și să scrii în acest caz drivere noi pentru soft este prea costisitor. Mulți producători dispun de capacități de producție limitate și au nevoie de ele pentru produse noi.

Tocmai aici se află șansa 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 piață. Acest lucru este valabil atît în domeniul sistemelor de operare DOS și OS/2 cît și pentru Unix. În totalitate există astăzi mai mult de 300 de pachete soft, care nu au probleme în funcționare cu 8514/A. Acesta este deja un motiv suficient pentru ca producătorii adaptoarelor grafice echipate cu cipuri TI, să ofere interfața soft 8514/A-AI și pentru adaptoarele lor. Această interfață program oferă modalități de programare sensibil mai apropiate de hardware decît interfața Tiga.

După cum a anunțat producătorul american de soft Grafpoint, emulări de soft, cum ar fi de ex. de terminale Tektronix și de servere X-Windows sub DOS, nu vor exista pentru Tiga, dar vor fi posibile pentru 8514/A. Cu toate acestea și 8514/A-AI este o interfață pur soft și 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 reușit să citească conținutul 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 preț. Și aceste clone sînt disponibile nu numai pentru microcanal ci și pentru magistralele normale ale AT-urilor. Copiile au o rezoluție de 1024 x 768 puncte la 16 și respectiv 256 de culori și lucrează în plus în modul imagine întregă (non-interlaced).

Această situație - după ce registrele hard ale lui 8514/A au devenit cunoscute - face acest adaptor încă și mai interesant pentru ca producătorii mari de soft să-l trateze în pachetele lor. Microsoft a făcut acest lucru cu Windows 3.0, același lucru făcîndu-l și Autodesk cu Autocad 11.0. Tocmai succesul uriaș 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 aplicații Windows, ci și cu programe CAD cum ar fi Autocad, PC-Draft sau cu soft-uri asemănătoare.

## Nici o grijă pentru soft

Și în domeniul Unix, 8514/A se scrie cu litere mari. X-Windows funcționează pe 8514/A, același lucru întîmplîndu-se și cu programele producătorilor Interactive și 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 piață. Criteriul important și hotărîtor al întregii situații este că utilizatorul nu trebuie să-și facă probleme pentru modul în care va fi sprijinit soft-ul. Pentru aceasta se angajează în competiție tot mai mulți producători de adaptoare grafice.

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

Testele comparative între Tiga și 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 același tact.

## Clonele sosesc

Alți producători se află deja în preajma lansării pe piață a unor seturi de cipuri 8514/A. Astfel un producător american de adaptoare grafice a anunțat 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 prețuri sensibil mai scăzute.

Prețul 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 rezoluție de 1024 x 768 la 256 de culori și în cazul unei echipări corespunzătoare chiar de 1280 x 1024 la 256 de culori.

Premisa de funcționare a unui adaptor grafic 8514/A este utilizarea unui monitor multi-frecvență, deoarece unele soft-uri accesează direct registrele hard. Aceasta poate însemna în cazuri concrete o rezoluție 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 și nu poate fi influențată din afară.

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

În dezvoltarea tehnică a adaptoarelor grafice sînt de așteptat cîteva noutăți interesante. Astfel vor exista adaptoare cu seturi de cipuri VGA "on-board". Prin două ieșiri separate pentru semnale VGA și 8514/A și o "Pass-Through" (trecere) internă a semnalelor VGA peste ieșirea de înaltă rezoluție, 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.

Și la capitolul ergonomie adaptoarele 8514/A au un avans: monitoarele cu o frecvență de împrăștiere a imaginii de 70 Hz aparțin deja standardului. Elaborarea unor adaptoare cu o frecvență 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 frecvență.

Pînă acum i-a reușit unui singur producător "să-i sufle" IBM-ului un standard grafic de pe piață, 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, toți ceilalți producători au urmat, mai devreme sau mai tîrziu, direcția de marș a uriașului 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 anunțarea de către IBM a lui XGA s-a deschis un nou capitol în problema standardelor grafice, o sentință rămîne însă: din punct de vedere tehnic sînt puține noutăți de semnalat. Clienții și distribuitorii sînt încurcați. Ce este XGA, ce avantaje aduc specificațiile ? Se vor impune ?*

IBM a anunțat un nou standard și întreaga branșă este tulburată. Extended Graphics Array (XGA) este conceptul care înfierbîntă multe sentimente în și așa destul de agitata lume a afacerilor cu grafică. "Interfața grafică XGA este o implementare busmaster, care oferă o nouă dimensiune în grafică", afirmă bineînțeles Big Blue. Totuși părerile celor din branșă sînt discordante.

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

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

Oferta de standarde în segmentul de piață al rezoluțiilor oferite de VGA Plus, Super-VGA și 8514/A era încilcită. Texas Instruments proclamase interfața soft Tiga, pentru adaptoare grafice de rezoluție ridicată echipate cu procesoare TI din familia TMS 340, ca "inteligentă". Vesa specificase un

standard propriu, iar producători ca Miro sau Spea pătrunseseră pe piață cu rezoluții de 2048 x 2048 pixeli și cu drivere proprii. IBM s-a ținut cu tărie de 8514/A și a așteptat să vadă dacă va exista necesitatea unor rezoluții mai mari sau a mai multor culori.

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

Ce avantaje oferă XGA ? Big Blue promite o creștere sensibilă a puterii sistemelor cu microcanal, deoarece adaptorul ca busmaster preia anumite funcțiuni ale procesorului principal degrevîndu-l pe acesta și lăsîndu-l să se ocupe între timp de alte probleme. IBM vorbește 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 instrucțiuni permite o creștere a vitezei. Suplimentar, utilizînd un adaptor XGA în modul standard VGA sînt posibile mai multe culori la o aceeași rezoluție.

### XGA ca busmaster

Extended Graphics Array a fost folosit mai întîi în sistemele microcanal PS/2 90 și 95. În modelul 90 XGA este integrat pe placa de bază ca interfață monitor. În 95 XP 486, XGA este prevăzut standard ca adaptor busmaster. Același adaptor grafic XGA poate fi instalat, la dorință, în toate modelele PS/2 cu procesoare 80386SX, 80386 și

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 și drivere pentru aplicații CAD (Computer Aided Design) speciale.

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

Branșa a privit pozitiv intrarea lui IBM în VESA (Video Electronics Standard Association). Acest pas indică cu claritate renunțarea lui IBM la rolul "călărețului singuratec" și lasă speranțe pentru consolidarea pieței de grafică. Cunoscătorii afirmă că deși specificațiile XGA au fost date publicității vortrece 9 pînă la 12 luni pînă cînd vor putea fi dezvoltate adaptoare noi. Greutățile vor fi produse mai ales de interfața de 32 de biți, ca și de cipurile care sînt mai complexe la XGA decît la VGA. După IBM este posibilă o creștere a rezoluției de pînă la 4096 x 4096 pixeli.

Dataquest a criticat totuși faptul că acest lucru nu este definit în mod clar, deoarece spațiul stabilit de IBM ar putea duce la incompatibilități. Un lucru este clar însă: IBM dorește să sprijine producătorii de adaptoare. Din această cauză XGA are șanse bune de a se impune. Doi "stropi de vermuț" există totuși: concurența din propria casă, 8514/A, care tocmai a început să se impună, și adaptorul Tiga care pînă acum a fost favorizat.

(R. M.)

# Standarde grafice

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

(Sursa: PC Magazin 4/91)

# Evaluarea performanțelor harddisk-urilor

*Cumpărătorul unui PC, chiar dacă de cele mai multe ori preferă soluția "de-a gata", din ce în ce mai des are posibilitatea opțiunii 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: prețul, capacitatea și "performanța" unității de disc, "performanța" fiind o sinteză a mai mulți parametri, ca de exemplu timpul de răspuns (acces), viteza de transfer, tratarea erorilor, fiabilitatea ș.a.m.d.

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

## Costul

Depinde desigur de firma producătoare și tipul unității de harddisk (vezi tabelul). Trebuie avută în vedere minimizarea costului total, nu pur și simplu cumpărarea celui mai ieftin disc disponibil la capacitatea dată. Trebuie avute în vedere mai multe considerente: interfața, controller-ul și funcțiile suplimentare ale acestuia, compatibilitatea cu restul echipamentului (nu doar ca interfață sau magistrală!), funcțiile integrate ( precum Cache Look-Ahead sau tratarea "transparentă", în timp real, a defectelor), posibilitățile de extindere ulterioară și fiabilitatea. Mai ales în condițiile noastre, unde depana-



rea ridică probleme, ar trebui să ezitam a cumpăra un harddisk garantat doar 6 luni sau un an și cu MTBF (Mean Time Between Failures - timpul mediu de defectare) nedepășind 25.000 ore, chiar dacă-i substanțial mai ieftin.

Totodată, nu trebuie exagerat - pentru aplicațiile uzuale, Hewlett Packard oferă discuri performante, cu MTBF de 150 și chiar 175.000 ore, iar Seagate dă la modelele de vîrf 150.000 ore în mediul unui birou obișnuit și 250.000 în sală calculator climatizată (clasa A). Ceea ce înseamnă 30 de ani la exploatarea non-stop în 3 schimburi și peste 100 de ani la utilizarea "casnică". Chiar în România, durata de viață morală a unui sistem nu depășește 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 octeți ce pot fi scriși pe o pistă. Capacitatea netă este cea utilă și este semnificativ mai mică - nr. octeți/sector x nr. sectoare/pistă x nr. piste/față disc x nr. fețe disc. Diferența se datorează în principal

spațiului dintre sectoare, informațiilor adiționale înscrise în antetul și "coada" fiecărui sector, spațiului (eventual) rezervat pentru compensarea erorilor. Factorul de corecție poate fi de 10-20%. Unele firme dau capacitatea brută, fiindcă livrează discul neformatat, deci utilizatorul poate să și-l organizeze cum dorește, 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 aplicațiilor 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 dactilografiate. De fapt, valoarea reală poate fi considerabil mai mică: informația în realitate nu-i compactă, ci cu mai multe "auxiliare", versiuni anterioare, fișiere provizorii etc. Sistemul de lucru ocupă de asemenea spațiu pe disc. Programele de procesare de texte ocupă adesea 2 Mo sau mai mult, iar aplicații 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.

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

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

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

1 - 3 ani.

Observație: relația capacitate - preț nu e liniară; modelele mai mari au costul specific mai redus. În plus, și timpul de acces este mai mic decât la modelele mai mici, ief-tine.

### Performanța

Un criteriu deosebit de important pentru performanță este timpul în care (în medie statistică) subsistemul de disc duce la îndeplinire o comandă de scriere sau citire a datelor. Acest timp se însu-mează cu timpii necesari pentru diversele operații pregătitoare pentru execuția comenzii. Pentru evaluarea performanței harddisk-ului nu este relevant timpul de overhead ("regie") al sistemului de operare și cel al controller-ului de disc, deoarece aceștia sînt relativ independenți de tipul unității de disc folosite. Pentru simplificare, putem considera - în primă aproximație - că unitatea de disc este legată direct la calculator, care-i sursa de date. Cînd unitatea de disc primește o comandă de citire sau scriere, trebuie ca mai întîi blocul capetelor să se poziționeze pe cilindrul unde există primul bloc de transferat. Acest timp se numește

timp de poziționare. Mărimea sa depinde evident de locul unde se face poziționarea, 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  $L$  poziț. medie =  $1/3 L$  poziț. max., relația fiind valabilă și pentru timpi. Valori extreme:  $\leq 10$  ms la discurile cele mai rapide, 20 - 30 ms în medie, 100 ms sau chiar mai mult la dischete sau discuri optice. Pentru siguranță, firmele serioase dau, în afara timpului mediu de poziționare, și pe cel maxim (pentru 100% din cursa capetelor). Unele discuri mai necesită un timp de poziționare mecanică, pentru stabilizarea la locația respectivă. Acest timp de stabilizare (settling) se specifică de obicei doar la unitățile "mici" de discuri, de 5 1/4" diametru.

După poziționare, trebuie așteptat pînă ce suprafața discului, care se rotește, aduce blocul de date dorit în dreptul capetelor de citire/scriere. Acesta este timpul de latență. Majoritatea unităților de disc rotesc discurile cu 3600 rpm (rotații pe minut), astfel că timpul necesar pentru o jumătate de rotație este de cca 8ms. Fiind egal la

toate discurile, nu-i un criteriu pentru departajare. Dar atenție! Unitățile de disc contemporane care vor să asigure performanțe superioare utilizează turații mai mari (lucru imposibil la discurile vechi, de diametru mare,  $\geq 8"$ ). Lucrînd la peste 4000 sau chiar 5600 r.p.m., timpul de latență scade la cca 5 ms - și atunci poate fi criteriu de selecție.

Urmează transferul datelor de pe disc la controller sau invers. Dacă nu se transferă cantități uriașe de date la o singură cerere de citire sau scriere, și acest timp poate fi dat factor comun și neglijat. Sistemele de operare precum VMS și OS, ca și majoritatea aplicațiilor (sisteme de baze de date, procesare de text, etc.) transferă în general puține 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 unităților de disc.

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

De altfel, o viteză foarte mare de transfer a datelor între disc și 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 și controller cu cea mai lentă, dintre controller și 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 decât posibilitățile reale, intrinseci.

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

#### Mic dicționar - harddisk-uri

### Tratarea automată a defectelor

Funcția 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 hărți ale defectelor (clasica tabelă cu "bad blocks") și să scrie soft utilitar dedicat evitării folosirii acestor zone.

La fabrică, suprafețele 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 conține locațiile tuturor sectoarelor cu defecte găsite pe disc. Pentru protecție 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 și 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 același echipament, în funcție 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, susținut (cînd bufferul SMB nu mai poate acționa) doar 1-2 Mo/s.

În concluzie, criteriile importante pentru stabilirea performanțelor unei unități de disc sînt:

- timpul de poziționare ;
- timpul de latență;
- timpul de transfer al datelor în întregul sistem (dat de interfața sau magistrala folosită).

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

Există o întreagă gamă de posibilități de a spori performanțele unui subsistem de harddisk.

### Optimizarea tranzacțiilor

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

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

Cazul optimal implică un șir destul de lung de așteptare a accesului la disc, șir deja format pentru unitatea de harddisk respectivă. Dacă nu-i cazul (de exemplu, dacă

nu-s multe cereri, deci șirul de așteptare este foarte scurt) îmbunătățirea este insesizabilă.

Pe lingă optimizarea prin sortare după distanța 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 rotație. Cum această metodă de optimizare a performanței este independentă de tipul unității de disc, nu este un criteriu de alegere a unui anumit tip de unitate de disc, ci doar o indicație (controllerul incorporat să "știe" acest mod de optimizare).

Pentru a spori performanța sistemului, se poate "instala" logic un disc mai mare, format din mai multe discuri fizice mai mici. Este inversul partiționării unui disc în mai multe discuri logice mai mici. Discul logic mare aduce două îmbunătățiri importante performanței:

- în primul rînd, unitățile de disc separate se pot poziționa suprapus;

-apoi, poziționarea se desfășoară 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 - unitățile de disc mici, de 20-40 Mo, au nu numai costul specific mai mare, dar și timpul de acces mai prost decît un harddisk mai mare, de 120-300 Mo. Se înlocuiește deci parțial selectarea mecanică a pistelor cu selectarea electronică a unităților de disc, iar fiabilitatea sistemului crește: 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 depășită, se poate face o împărțire (repartiție) 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 împărțire/separare aduce avantaje de viteză, dar mai mici decît se speră.

### Cache

O altă posibilitate, ulterioară, de creștere a performanței harddisk-ului este folosirea unui controller cu "cache", memorie-tampon de "prindere". ("Cache" se pronunță "caș" și vine din franceză: cacher = a ascunde, cache = ascunzătoare).

### Mic dicționar - harddiskuri

#### Multisegmentare

Unitățile clasice de disc de masă aveau un număr fix de sectoare pe pistă, indiferent de poziția pistei. Ori, pistele fiind dispuse concentric pe disc, este evident că pe cele exterioare este spațiu mai mult; cu cît pista este mai aproape de centrul de rotație, cu atît mai "strîmt" devine spațiul pentru sectoarele ce trebuie să încapă pe pistă. Restricția cu numărul sectoarelor se datora controllerelelor - realizate pe atunci cu circuite cu grad de integrare redus (TTL-uri MSI). Nevoia presantă de capacități de memorare mai mari - dublată de progresele uriașe în domeniul circuitelor VLSI, ULSI și ASIC - a dus la apariția unor controllere cu funcții deosebit de complexe, dispunînd de algoritmi sofisticăți de gestiune a spațiului fizic al discului. Controllerele cu firmware de multisegmentare împart pistele într-un număr variabil de sectoare, în funcție de poziția pistei - mai multe spre exterior, mai puține spre interior.

Fiecare cerere de scriere pe disc va fi procesată exact ca și la un controller fără cache: fiecare din blocurile de înregistrat este trimis la capetele de citire/scriere ale unității de disc cuplate. În plus însă, controllerul își "notează" adresele blocurilor și conținutul lor într-o memorie internă, denumită cache. Memoria aceasta are o capacitate variabilă în funcție de tipul unității de harddisk. Astfel, la controllerul QD-35 al firmei Emulex (pentru pînă la patru unități de harddisk cu interfață SMB, viteza de transfer de max. 3 Mo/s) memoria cache pentru stocarea intermediară a blocurilor de date și 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 poziționării, latenței, etc. Dacă nu se află în cache, va fi citit de pe disc și transmis, ca în cazul anterior. Pe deasupra, QD-35 reține și în această situație adresa și conținutul blocului.

Cînd memoria-tampon (bufferul-cache) s-a umplut, se va supra-

scrie, începînd cu blocul care n-a mai fost utilizat/solicitat de cel mai mult timp - o procedură gen FIFO. Se definește un coeficient de reușită, "hit-rate", a "prinderii" datelor în cache. Gu un cache-controller performant, coeficientul de reușită 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 funcție de împrejurări, de aplicația concretă.

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

-calculatorul are memoria principală relativ redusă;

-unitatea de harddisk primește un mare număr de cereri (ca și la "șirul de așteptare" - 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 aplicații care, în mod repetat, citesc și scriu în același fișier, cu schimbul.

### Controllere pentru discuri SMB

Harddisk-urile cu interfață SMB

sînt performante (capacitate mare, debit ridicat) și destul de scumpe, ceea ce explică de ce controllerele lor beneficiază de cele mai multe perfecționări. Emulează protocolul MSCP și au în consecință multe avantaje. Spre deosebire de alte emulări, MSCP lucrează independent de mărirea discului, de structura sa fizică. Softul calculatorului "întreabă" controllerul despre geometria/organizarea harddisk-ului conectat. La MSCP, sistemul vede doar blocuri numerotate în ordine crescătoare, nu piste, cilindrii, sectoare sau fețe 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 și ea transparentă.

Minimizarea timpului de poziționare dintre două cereri de citire succesive, menționată anterior, se obține cu ajutorul unui așa numit "algoritm de ascensor". Întodeuna cea mai apropiată cerere din șirul de așteptare va fi rezolvată - în ordinea distanței de poziționare pe disc. Pentru ca o anumită cerere să nu rămînă nerezolvată un timp foarte lung, există un indicator de "tratament cinstit", Fairness Count, și un șir de așteptare expres, care asigură soluționarea rapidă a cererii corespunzătoare.

Controllerele MSCP au și 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 necesități, pe timpul transferului de date, astfel încît să nu mai fie necesară o rotație 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 poziționarea corectă, ș. a. m. d.

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

### Noutăți

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

- multisegmentare
- cache adaptiv, ce evaluează modul în care datele sînt cerute de sistem și adoptează automat algoritmul cel mai eficient
- cache cu predicție, care minimizează timpul de acces la transferuri lungi
- consum redus de energie (2-3 W și chiar mai puțin în "standby")
- corecție automată a erorilor (ECC - cod polinomial de 48 biți, permite corecția unui sector cu "salvă" de pînă la 11 biți eronați; rata erorilor scade de 1000 de ori (!))
- firmware integrat de autodiagnostic, foarte complet, care identifică și semnalează prompt condițiile de eroare
- robustețe mecanică f. bună - rezistă la șocuri de 100 G cînd nu e în stare de funcționare - datorită dimensiunilor reduse și a sistemului automat de "parcare" a capetelor în zone special destinate
- sistem de servopозиționare hibrid, folosind suprafața discului pentru a reduce timpul de acces la valori de ordinul a 10 ms și servo înglobat (pistă specială cu semnale servo) pentru îmbunătățirea preciziei de poziționare a capetelor

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

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

Funcția 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, și nu pur și simplu în ordinea apariției lor.

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

Controllerul QD-35 are, în afara funcțiilor descrise mai sus și a memoriei cache, un procesor intern foarte rapid: un 68020 în capsulă ceramică Quadflatpack. Interesant este cum se face "prinderea" datelor - la eficiența unei memorii cache contribuind nu numai mărirea acesteia, ci algoritmi utilizați. Firma Emulex declară că la proiectarea lui QD-35 a avut în vedere în mod special găsirea unei modalități eficiente de "prindere", fără influențele negative ce se întîlnesc la alte controllere cu cache asupra performanțelor la lucrul cu mai multe unități de disc (MDP) sau la sortarea comenzilor (CS).

La QD-35, pentru prima dată, memoria cache este configurabilă după dorință. Utilizatorul are posibilitatea de a stabili ce domeniu al discului să fie "prins" și 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 și reglarea parametrilor, supraveghează tranzațiile cu harddisk-ul și dă informații statistice asupra valorilor timpilor de acces și a eficacității "prinderii", permițînd astfel stabili-

rea parametrilor optimi. Atenție! O memorie cache așa de mare poate prezenta și dezavantaje: dacă alimentarea sistemului de calcul nu este asigurată de o sursă neîntreruptibilă (UPS - Uninterruptible Power Supply), în caz de pană de curent toate informațiile din memoria controllerului se pierd ...

### Interfața

Evident, controllerul și unitatea de disc trebuie să aibă aceeași interfață. Actualmente, majoritatea harddisk-urilor uzuale (de 3,5" și 5,25", capacități pînă în 300 Mo) au controllerul incorporat ("imbedded" - se vorbește atunci de interfață AT sau IDE - Intelligent Drive Electronics). Sînt mai răspîndite 3 tipuri de interfețe:

#### ST506

Interfața 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) și RLL (Run Length Limited). Cu RLL, datele sînt procesate suplimentar: prin intermediul unei logici complicate, se obțin coduri în care pot exista între 1 și 7 zerouri succesive (RLL 1.7) sau între două și șapte zerouri succesive (RLL 2.7). Se obține astfel o îmbunătățire suplimentară a performanțelor, viteza de transfer poate crește cu 50%. (încă o dată trebuie subliniat că

este vorba de transferul între hard-disk și 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 rotații 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 și modul RLL al controllerului: toate unitățile de disc cu ST506 acceptă modul MFM, dar nu toate pot lucra și în RLL. Cablele sînt identice și uneori un controller RLL pare să se înțeleagă cu un disc MFM. Dar, în astfel de cazuri, datele Dvs. sînt în mare pericol!

#### ESDI

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

#### 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 interfața serială (Mbit/s)	5	7,5	10	12/40
-unitate (Mbit/s)			10/15	10/15
-sistem (ko/s)	450	650	850/900-1	850/1200
busul de date	serial	serial	serial	8 (16) biți
BIOS integrat	nu	da	da	da

nu este o simplă interfață de hard-disk, ci o veritabilă magistrală de intrare/ieșire. Pe magistrală SCSI se pot lega în paralel maximum 8 surse de date, 8 noduri. Cele 8 noduri pot fi un calculator și mai multe unități de harddisk, sau alte periferice: scanner, unitate de bandă magnetică, imprimantă, disc magneto optic, etc.

Magistrala SCSI "dă" mai multă inteligență perifericelor și cere mai puțin 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. Și cu asta se trece și la capitolul următor:

### Sistemul de operare

Inițial, am neglijat timpul de overhead al sistemului de operare, fiind același pentru toate tipurile de harddisk-uri. Totuși, 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 depășită chiar dacă hard-disk-ul are mai mult de o mie de cilindri (piste per față de disc): cineva, de regulă controllerul, va masca realitatea de pe disc, disponibilizând însă mai multe fețe 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 și Unix se poate lucra în multitasking, cu mai multe unități de disc mici - cum s-a arătat mai înainte, pînă ce unul își poziționează capetele, ne adresăm altuia pentru transferul de date.

### Transportabilitatea

Este un criteriu de selecție esențial atunci cînd se impune ca datele să poată fi transportate rapid și ușor 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 ușor detașată pentru transport. Unitatea de bază este un șasiu, care conține sursa de alimentare și conectorii, și poate accepta unul sau două containere cu discuri. Containerele, etanșe, cuprind un mediu controlat, clasă 100 (sub 100

redus, iar probabilitatea de apariție a unor erori la copiere este considerabil diminuată. Un alt suport de date interschimbabil ce le-ar putea face concurență este discul optic reînregistrabil (disc magneto-optic), dar acesta prezintă actualmente inconveniente ca:

- deși 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 și de 10 ori la scriere, în timp ce discurile amovibile au timpi de răspuns comparabili cu cei ai harddisk-urilor Winchester normale;

### Filecard

Este numele sub care sînt cunoscute harddiskurile ce se montează direct într-un slot liber al PC-ului, fiind solidare cu cartela care se introduce și 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.

de particule de praf / ft<sup>3</sup>), în care funcționează discurile.

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

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

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

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

Față de transportul sau asigurarea (back-up) datelor prin copierea pe bandă, harddisk-urile amovibile prezintă un timp de acces mult mai

- și rata de transfer este mai scăzută, 0,5-1 Mo/s față 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 înălțimi diferite ale cutiei:

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

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

Discurile de 8" (20 cm), cu interfețe tip SMB sau SCSI, au capacități de pînă la 2000-3200 Mo;

Cele mai mari unități de disc au diametrul de 9", dar sînt puțin răspîndite, nefiind competitive nici ca performanțe, nici ca preț (costisitoare, dimensiuni, consum și greutate foarte ridicate).

ing. Cristian Mulide, L.C.C.E.

Firma	Model	Preț (DM)	Formă	Capacitate (Mo)	Timp de acces (ms)	Rata de transfer (Mo/s)	Tip interfață	Mod de înregistrare	Cache integrat (ko)	Cilindri/capete	MTBF (măsoare)
Adcomp	ALD Turbo-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	3,5" Winchester	105	25	1,25	AT	RLL (2,7)			
	CP-3204F	1676	3,5" Winchester	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	145	25	1,5	SCSI	RLL (1,7)	24	1334/2	50
	M2614ESA	2.438	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)	1 x 60	1435/9+1	
	M2624SA	3.480	3,5" FH	520	12	3	SCSI-2	RLL (1,7)	1 x 60	1435/11+1	
Hewlett-Packard	HP C2233S		3,5"	234	13	2,5	SCSI-2				
Hitachi	DK516C-16		5,25"	1652	13,5		SCSI				
IBM	Modell 60/80	1.250	5,25"	41	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"	140	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"	142	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	145	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"	152	< 17	2	SCSI		64	1219/2	
	LPS 52 AT	798	3,5"	152	< 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"	142	19	2/4	SCSI	RLL (2,7)		834/3	
	Pro Drive 80S	927	3,5"	184	19	2/4	SCSI	RLL (2,7)		834/6	
	Pro Drive 120 AT		3,5"	120	< 15	4	AT	RLL (1,7)		1123/5	
	Pro Drive 210 AT		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"	140	25	5	ST-506				
Western Digital	Filecard 40	752	3,5"	140	29	0,9	ISA	RLL			



## Rutine care clasifică

Aceste rutine clasifică caracterele ASCII în litere, caractere de control, semne de punctuație, 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 șiruri alfanumerice în diferite reprezentări numerice și invers, din majuscule în minuscule și 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(double value, 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 și nume de căi

int	chdir(const char *path)	<dir.h>
int	findfirst(const char *pathname, struct fblk *fblk, int attrib)	<dir.h>
int	findnext(struct fblk *fblk)	<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 segs)	<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 interruptno)) ()	<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, unsigned ofs)	<dos.h>
void	outport(int portid, int value)	<dos.h>
void	outportb(int portid, unsigned char value)	<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	randhwr(struct fcb *fcb, int rcnt)	<dos.h>
void	segread(struct REGS *seggp)	<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	severify(int value)	<dos.h>

void	sleep(unsigned seconds)	<dos.h>
int	unlink(const char *filename)	<dos.h, io.h, stdio.h>

## Rutine de manipulare șiruri și zone de memorie

void	*memcpy(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	*strcpy(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	strncpy(const char *s1, const char *s2, size_t n)	<string.h>
char	*strdup(const char *s)	<string.h>
char	*strerror(int errnum)	<string.h>
size_t	strlen(const char *s)	<string.h>
char	*strrchr(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	*strset(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>

void	perror(const char *s)	<errno.h>	int far	getmaxy(void)	<graphics.h>
void far	bar(int left, int top, int right, int bottom)	<graphics.h>	char * far	getmodename(int mode_number)	<graphics.h>
void far	bar3d(int left, int top, int right, int bottom, int depth, int topflag)	<graphics.h>	void far	getmoderange(int graphdriver, int far *lomode, int far *himode)	<graphics.h>
void far	circle(int x, int y, int radius)	<graphics.h>	void far	getpalette(struct palette_t *palette)	<graphics.h>
void far	cleardevice(void)	<graphics.h>	int far	getpalettesize(void)	<graphics.h>
void far	clearviewport(void)	<graphics.h>	unsigned far	getpixel(int x, int y)	<graphics.h>
void far	closegraph(void)	<graphics.h>	void far	gettextsettings(struct textsettings_t *textypeinfo)	<graphics.h>
void far	detectgraph(int far *graphdriver, int far *graphmode)	<graphics.h>	void far	getviewsettings(struct viewport_t *viewport)	<graphics.h>
void far	drawpoly(int numpoints, int far *polypoints)	<graphics.h>	int far	getx(void)	<graphics.h>
void far	ellipse(int x, int y, int stangle, int endangle, int xradius, int yradius)	<graphics.h>	int far	gety(void)	<graphics.h>
void far	fillellipse(int x, int y, int xradius, int yradius)	<graphics.h>	void far	graphdefaults(void)	<graphics.h>
void far	fillpoly(int numpoints, int far *polypoints[])	<graphics.h>	char *far	grapherrormsg(int errorcode)	<graphics.h>
void far	floodfill(int x, int y, int border)	<graphics.h>	void far	_graphreemem(void far *ptr, unsigned size)	<graphics.h>
void far	getarccoords(struct arccoordstype, far *arccoords)	<graphics.h>	void far * far	_graphgetmem(unsigned size)	<graphics.h>
void far	getaspectratio(int far *xasp, int far *yasp)	<graphics.h>	int far	graphresult(void)	<graphics.h>
int far	getbkcolor(void)	<graphics.h>	unsigned far	imagesize(int left, int top, int right, int bottom)	<graphics.h>
int far	getcolor(void)	<graphics.h>	void far	initgraph(int far *graphdriver, int far *graphmode, char far *pathdriver)	<graphics.h>
struct palette_t *far	getdefaultpalette(void)	<graphics.h>	int far	installuserdriver(char far *name, int huge (*detect) (void))	<graphics.h>
char *far	getdrivername(void)	<graphics.h>	int far	installuserfont(char far *name)	<graphics.h>
void far	getfillpattern(char far *pattern)	<graphics.h>	void far	line(int x1, int y1, int x2, int y2)	<graphics.h>
void far	getfillsettings(struct fillsettings_t *fillinfo)	<graphics.h>	void far	linerel(int dx, int dy)	<graphics.h>
int far	getgraphmode(void)	<graphics.h>	void far	lineto(int x, int y)	<graphics.h>
void far	getimage(int left, int top, int right, int bottom, void far *bitmap)	<graphics.h>	void far	moverel(int dx, int dy)	<graphics.h>
void far	getlinesettings(struct linesettings_t *lineinfo)	<graphics.h>	void far	moveto(int x, int y)	<graphics.h>
int far	getmaxcolor(void)	<graphics.h>	void far	outtext(char far *textstring)	<graphics.h>
int far	getmaxmode(void)	<graphics.h>	void far	outtextxy(int x, int y, char far *textstring)	<graphics.h>
int far	getmaxx(void)	<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	registerbdriver(void (*driver) (void))	<graphics.h>
			int	registerbfont(void (*font)(void))	<graphics.h>
			void far	restorecrtmode(void)	<graphics.h>

char	*strupr(char *s)	<string.h>	long int	labs(long int x)	<stdlib.h>
size_t	strxfrm(char *s1, char *s2, size_t n)	<string.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>
[int   double]	abs([int   complex] x)	<math.h, complex.h, stdlib.h>	unsigned long	_lrot(unsigned long val, int count)	<stdlib.h>
[double   complex]	acos([double   complex] x)	<complex.h, math.h>	unsigned long	_lrotr(unsigned long val, int count)	<stdlib.h>
double	arg(complex z)	<complex.h>	int	_mexcep(enumerated type)	<math.h>
[double   complex]	asin([double   complex] x)	<complex.h, math.h>	int	matherr(struct exception *e)	<math.h>
[double   complex]	atan([double   complex] x)	<complex.h, math.h>	double	modf(double x, double *ipart)	<math.h>
double	atan2(double y, double x)	<complex.h, math.h>	double	norm(complex x)	<complex.h>
double	atof(const char *s)	<stdlib.h, math.h>	complex	polar(double mag, double angle)	<complex.h>
int	atoi(const char *s)	<stdlib.h>	double	poly(double x, int degree, double coeffs [])	<math.h>
long	atol(const char *s)	<stdlib.h>	[double   complex]	pow([double   complex] x, [double   complex] y)	<complex.h, math.h>
bcd	bcd(int x)	<bcd.h>	double	pow10(int p)	<math.h>
bcd	bcd(double x)	<bcd.h>	int	rand(void)	<stdlib.h>
bcd	bcd(double x, int decimals)	<bcd.h>	int	random(int num)	<stdlib.h>
double	cabs(struct complex z)	<math.h>	void	randomize(void)	<stdlib.h>
double	ceil(double x)	<math.h>	double	real(complex x)	<complex.h>
unsigned int	_clear87(void)	<float.h>	unsigned	_rot(unsigned val, int count)	<stdlib.h>
complex	complex(double real, double imag)	<complex.h>	unsigned	_rotr(unsigned val, int count)	<stdlib.h>
double	conj(complex z)	<complex.h>	[double   complex]	sin([double   complex] z)	<complex.h, math.h>
unsigned int	_control87(unsigned int new, unsigned int mask)	<float.h>	[double   complex]	sinh([double   complex] z)	<complex.h, math.h>
[double   complex]	cos([double   complex] z)	<complex.h, math.h>	[double   complex]	sqrt([double   complex] x)	<complex.h, math.h>
[double   complex]	cosh([double   complex] z)	<complex.h, math.h>	void	srand(unsigned seed)	<stdlib.h>
div_t	div(int numer, int denom)	<math.h>	unsigned int	_status87(void)	<float.h>
char	*ecvt(double value, int ndig, int *dec, int *sign)	<stdlib.h>	double	strtod(const char *s, char **endptr)	<stdlib.h>
[double   complex]	exp([double   complex] z)	<math.h>	long	strtoul(const char *s, char **endptr, int radix)	<stdlib.h>
double	fabs(double x)	<math.h>	unsigned long	strtoul(const char *s, char **endptr, int radix)	<stdlib.h>
char	*fcvt(double value, int ndig, int *dec, int *sign)	<stdlib.h>	[double   complex]	tan([double   complex] x)	<complex.h, math.h>
double	floor(double x)	<math.h>	[double   complex]	tanh([double   complex] x)	<complex.h, math.h>
double	fmod(double x, double y)	<math.h>	char	*ultoa(unsigned long value, char *string, int radix)	<stdlib.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, double y)	<math.h>			
double	imag(complex z)	<complex.h>			
char	*ltoa(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>



# Turbo C++ 1.0

## Memento - Funcții

void far	sector(int x, int y, int stangle, int endangle, int xradius, int yradius)	<graphics.h>	int	fprintf(const char *format [, argument,...])	<conio.h>
void far	setactivepage(int page)	<graphics.h>	int	puts(const char *str)	<conio.h>
void far	setallpalette(struct palettetype far *palette)	<graphics.h>	int	_read(const char *path, int attrib)	<io.h>
void far	setaspectratio(int xasp, int yasp)	<graphics.h>	int	_read(const char *path, int amode)	<io.h>
void far	setbkcolor(int color)	<graphics.h>	int	readnew(const char *path, int mode)	<io.h>
void far	setcolor(int color)	<graphics.h>	int	createmp(char *path, int attrib)	<io.h>
void far	setfillpattern(char far *upattern, int color)	<graphics.h>	int	scanf(char *format [, address, ... ])	<conio.h>
void far	setfillstyle(int pattern, int color)	<graphics.h>	int	dup(int handle)	<io.h>
void far	setlinestyle(int linestyle, unsigned upattern, int thickness)	<graphics.h>	int	dup2(int oldhandle, int newhandle)	<io.h>
void far	setpalette(int colormum, int color)	<graphics.h>	int	eof(int handle)	<io.h>
void far	setrgbpalette(int colormum, int red, int green, int blue)	<graphics.h>	int	fclose(FILE *stream)	<stdio.h>
void far	settextjustify(int horiz, int vert)	<graphics.h>	int	fcloseall(void)	<stdio.h>
void far	settextstyle(int font, int direction, int charsize)	<graphics.h>	FILE	*fdopen(int handle, char *type)	<stdio.h>
void far	setusercharsize(int multx, int divx, int multy, int divy)	<graphics.h>	int	feof(FILE *stream)	<stdio.h>
void far	setviewport(int left, int top, int right, int bottom, int clip)	<graphics.h>	int	ferror(FILE *stream)	<stdio.h>
void far	setvisualpage(int page)	<graphics.h>	int	flush(FILE *stream)	<stdio.h>
void far	setwritemode(int mode)	<graphics.h>	int	fgetc(FILE *stream)	<stdio.h>
int far	textheight(char far *textstring)	<graphics.h>	int	fgetchar(void)	<stdio.h>
int far	textwidth(char far *textstring)	<graphics.h>	int	getpos(FILE *stream, pos_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	puts(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 (pos_t *pos)	<stdio.h>
			int	fstat(int handle, struct stat *statbuf)	<sys/stat.h>
			long	tell(FILE *stream)	<stdio.h>
			size_t	fwrite(const void *ptr, size_t size, size_t n, FILE *stream)	<stdio.h>

### Rutine de intrare/ieșire

Aceste rutine asigură posibilități de intrare /ieșire la nivel stream și DOS

int	acces(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>

3



# Turbo C++ 1.0

## Memento - Funcții

void	*calloc(size_t nitems, size_t size)	<alloc.h>	void	abort(void)	<stdlib.h, process.h>
unsigned [long]	coreleft(void)	<alloc.h>	int	exec1(char *path, char *arg0, ... NULL)	<process.h>
void far	*farcalloc(unsigned long nunits, unsigned long unitsz)	<alloc.h>	int	exec1e(char *path, char *arg0, ... NULL, char **env)	<process.h>
unsigned long	farcoreleft(void)	<alloc.h>	int	exec1p(char *path, char *arg0, ...)	<process.h>
void	farfree(void far *block)	<alloc.h>	int	exec1pe(char *path, char *arg0, ... NULL, char **env)	<process.h>
int	farheapcheck(void)	<alloc.h>	int	execv(char *path, char *argv[])	<process.h>
int	farheapcheckfree(unsigned int filvalue)	<alloc.h>	int	execve(char *path, char *argv[], char **env)	<process.h>
int	farheapchecknode(void *node)	<alloc.h>	int	execvp(char *path, char *argv[])	<process.h>
int	farheapfillfree(unsigned int filvalue)	<alloc.h>	int	execvpe(char *path, char *argv[], char **env)	<process.h>
int	farheapwalk(struct farheapinfo *hi)	<alloc.h>	void	_exit(int status)	<process.h, stdlib.h>
void far	*farmalloc(unsigned long nbytes)	<alloc.h>	void	exit(int status)	<process.h, stdlib.h>
void far	*farmalloc(void far *oldblock, unsigned long nbytes)	<alloc.h>	int	getpid	<process.h>
void	free(void *block)	<alloc.h, stdlib.h>	int	raise(int sig)	<signal.h>
int	heapcheck(void)	<alloc.h>	void	(*signal(int sig, void (*func) int sig, int subcode)))(int)	<signal.h>
int	heapcheckfree(unsigned int filvalue)	<alloc.h>	int	spawnl(int mode, char *path, char *arg0, ... NULL)	<process.h>
int	heapchecknode(void *node)	<alloc.h>	int	spawnle(int mode, char *path, char *arg0, ... NULL, char *envp[])	<process.h>
int	heapwalk(struct heapinfo *hi)	<alloc.h>	int	spawnlp(int mode, char *path, char *arg0, ... NULL)	<process.h>
void	*malloc(size_t size)	<alloc.h, stdlib.h>	int	spawnlpe(int mode, char *path, char *arg0, ... NULL, char *envp[])	<process.h>
void	*realloc(void *block, size_t size)	<alloc.h, stdlib.h>	int	spawnrv(int mode, char *path, char *argv[])	<process.h>
void	*sbrk(int incr)	<alloc.h>	int	spawnrve(int mode, char *path, char *argv[], char *envp[])	<process.h>
int	setblock(unsigned segx, unsigned newsz)	<dos.h>	int	spawnrpe(int mode, char *path, char *argv[], char *envp[])	<process.h>

### Alte rutine

void	delay(unsigned milliseconds)	<dos.h>
struct locov	*locaicom(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 și opresc procese noi dinăuntrul altor procese

### 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>

7



```

int     getc(FILE *stream)                <stdio.h>
int      getch(void)                     <conio.h>
int      getchar(void)                   <stdio.h>
int      getchc(void)                    <conio.h>
int      gettimeofday(int handle, struct ttime *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      putchar(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 ttime *ftimep) <io.h>
int      setmode(int handle, int amode)    <io.h>
int      setvbuf(FILE *stream, char *buf, int type, size_t size) <stdio.h>
int      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 *s)                  <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      vscanf(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 interfață (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>
int      biostime(int cmd, long newtime)    <bios.h>
void     ctrlbrk(int (*handler)(void))     <dos.h>
void     disable(void)                      <dos.h>
int      doscerr(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>

```



```

char     *ecvt(double value, 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(long value, char *string, int radix) <stdlib.h>
void     *malloc(size_t size)               <alloc.h, stdlib.h>
int      putenv(const 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.h>
int      system(const char *command)        <process.h, stdlib.h>

```

### Rutine de afișare text în ferestre

```

void     clrcol(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     inline(void)                        <conio.h>
void     lowvideo(void)                      <conio.h>
int      movetext(int left, int top, int right, int bottom,
                void *destin) <conio.h>

```

```

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>
int      window(int left, int top, int right, int bottom) <conio.h>

```

### Rutine pentru dată și 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 preferințele 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, concurența de la Microsoft pare depășită: tot mai mulți vânzători de PC-uri oferă sistemul de operare DR-DOS ca dotare de bază.

Și este de înțeles 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 șansă primului. Pentru a recîștiga o parte din piață - sau măcar pentru a nu o pierde de tot, dat fiind că "bătrînul" DOS va determina încă și în continuare destinele multor PC-uri, Microsoft a "ieșit" 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 față de ceea ce vă prezentăm. Vă vom face cunoscută apariția versiunii finale de cum vom ști mai mult. (Gurile rele spun că nici nu are sens să pornești cu 5.0, fiind mai prudent să aștepti direct 5.01... Vorba rea are oarecare teme: la lansarea versiunii 4.0, au fost inacceptabil de multe erori și 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 neașteptat de lungă - cea mai completă și mai costisitoare de pînă acum, dacă ar fi să dăm crezare unor declarații făcute în acest sens de către reprezentanții ai Microsoft).

Avantajul major al DR-DOS era pînă acum modul de gestionare al memoriei. Astfel, în DR-DOS, este

posibilă "strămutarea" anumitor părți 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 aplicații pînă la 620 kbyte și chiar mai mult. Un progres imens, avînd în vedere faptul că aplicațiile moderne sînt dependente de orice bit de memorie operativă. Și nu numai sistemul de operare poate fi mutat, ci și 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 specificațiilor LIM.

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

Și totuși, 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 posibilități și dispun suplimentar de cîte o funcție de help proprie (care poate afișa, la cerere, lista parametrilor și semnificațiile lor). Ceea ce arată că pînă și un sistem de operare orientat pe comenzi poate fi "prietenos".

Din furnitură face parte și o suprafață utilizator grafică, care cuprinde aceleași funcții ca cea din MS-DOS 4.01. Deoarece Digital Research dispune oricum de soft comparabil, suprafața grafică DOS nu a mai trebuit special făcută: s-a recurs la deja consacratul "GEM", rebotezîndu-l "ViewMax". Acesta nu are toate funcțiile din "GEM", dar în ce privește forma de prezentare și modul de operare nu sînt mari

deosebiri. Altfel este la MS-DOS 4.01, unde Microsoft a respectat cu strictețe prescripțiile SAA ale IBM, creînd o suprafață complet nouă (dar uitînd cîteva funcții importante pentru ea).

Acest lucru ar urma să se schimbe acum, în MS-DOS 5.0: ca suprafață, se zvonea că va fi ceva asemănător cu Windows, (poate chiar o versiune mai veche de Windows), iar în plus, vor fi așteptate utilitare încît producătorii "clasici" de utilitare ca "PC Tools" sau Peter Norton vor fi nevoiți să-și schimbe meseria, la apariția unor programe ca "undelete", "unformat" sau "diskdoctor". Realitatea însă se prezintă mult mai modest: suprafața a rămas aceeași, cu cîteva inovații desigur, iar utilitare noi sînt puține: "unformat" și un program de compresie - decompresie. Și totuși: chiar dacă nu se vede din prima privire, verificarea capacității 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 concurenței: se folosește memoria suplimentară pentru a descărca memoria operativă de drivere și părți ale sistemului de operare.

În afara acestor performanțe îmbunătățite relative la memoria suplimentară, mare lucru nu s-a schimbat. Nici măcar n-au fost dotate cu funcții de help diversele utilitare. În continuare lipsește o protecție efectivă prin parolă a fișierelor 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, suprafața utilizator. La acest lucru se adaugă faptul că parola este înscrisă într-un fișier în care sînt trecuți și ceilalți parametri ai "shell" - ului. La prima vedere, combinația de litere este cifrată, dar o privire mai atentă relevă imediat faptul că este vorba de o simplă translație în cadrul codului ASCII. Deci nici un fel de piedică pentru profesioniști.

La acest lucru se adaugă faptul că în afara suprafeței utilizator, protecția datelor nu funcționează de loc, datele putând fi astfel deteriorate sau văzute chiar și de către începători. La DR-DOS, protecția este generalizată la nivelul sistemului.

Această listă poate fi cu ușurință continuată: cu comanda "type" de exemplu. MS-DOS 5.0 permite afișarea unui singur fișier la un moment dat, pe când DR-DOS permite folosirea de specificatori generici (wildcards) (\*,?). Pe lângă aceasta, o "pauză" după fiecare ecran se obține cu parametrul "/p", pe când MS-DOS cere lansarea unei a doua comenzi: "type nume | more". Cine vrea să schimbe configurația MS-DOS, trebuie să treacă prin dificultățile editării fișierelor "autoexec.bat" și "config.sys". DR-DOS în schimb oferă pentru aceasta un program special de setup, care permite definirea comodă a modului de folosire a memoriei, definește cartela grafică în uz sau configurează drivere pentru mouse-ul folosit.

În ce privește "depanările", Digital Research a renunțat la un echivalent al "debug"-ului de la Microsoft, scoțind în schimb la lumină o relicvă de pe vremea CP/M-ului: "sid", adică "symbolic instruction debugger", care îndeplinește în esență aceleași funcții ca și "debug".

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

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

Avantaj DR-DOS și la comenzile de ștergere: cu "delq" este posibilă ștergere selectivă a unor fișiere. Pentru fiecare fișier, programul pune o întrebare suplimentară la consolă pentru a se asigura că fișierul trebuie într-adevăr șters. Iar "xdel"

permite chiar șterge, ea de directoare și subdirectoare.

Comparația celor două suprafețe utilizator grafice nu relevă nici un fel de asemănări. Ambele răspund însă la două necesități importante: pe de o parte permit administrarea harddisk-ului, iar pe de altă parte permit lansarea fără probleme a unor aplicații DOS. Programele trebuie configurate înainte de lansare. Multe din ele, în special utilitare, au nevoie de preluarea unor parametri înainte de lansare. Ambele suprafețe permit astfel de aranjamente; se poate face ca parametri să fie predați în mod automat, sau se solicită o "fereastră" înainte de orice lansare. Pur vizual, Viewmax oferă mai mult decât Shell. În timp ce Shell atribuie oricărui program nou primit în listă un me-reu același 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 împărțită apoi în grupe și subgrupe. Utilizatorul își poate astfel alcătui "dosare" proprii pentru fiecare categorie de programe. Astfel, toate programele ce țin de contabilitate pot fi grupate în "dosarul contabilitate", în timp ce utilitarele pot fi grupate în "tools".

Nu prea sînt dificultăți cu nici un program, la lansarea lui de la nivelul suprafeței, deoarece ambii concurenți "se retrag" aproape total din memoria principală la lansarea unei aplicații DOS, revenind abia la terminarea programului.

Ambele sisteme de operare dispun de editoare orientate ecran. Editorul MS-DOS, care acum în sfîrșit ar putea înlocui de mult învechitul editor "edlin", orientat pe rînduri, lucrează chiar cu meniuri moderne tip "pull-down" și ferestre. Funcțiile disponibile sînt însă limitate la căutări, înlocuiri precum și unele funcții pe blocuri de text. Pe cînd ecranul poate fi configurat

după gust de fiecare utilizator. Ceea ce nu înseamnă sfîrșitul pentru "edlin" - cine îl dorește, îl găsește 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 câteva funcții pe care le oferă pot fi lansate numai prin combinații de taste. Pentru răsfațai obișnuiți cu ferestre și SAA, acest lucru cere acomodare și învățare.

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 alți bani) se găsește sistemul de programare "Quick Basic" în versiunea 1.0. Este practic un mediu de dezvoltare Basic, destul de performant. Păcat că lipsește opțiunea de compilare: e vorba deci în continuare de un interpretor "pur-sînge". În schimb, are un debugger aproape profesional și un help online la introducere. Subprograme pot fi găsite rapid cu ajutorul unor ferestre suplimentare, deasemenea funcții pot fi inserate via ferestre. Astfel încît cine are nevoie de un program rapid, vrea să se inițieze în programare sau pur și simplu are nevoie de un program pentru el, are tot ce-i trebuie în Quick Basic, chiar și în această versiune.

Relativ la compatibilitate, chiar și versiunea "de probă" MS-DOS s-a dovedit surprinzător de stabilă: n-au fost probleme cu nici una din aplicațiile testate, de la "Word" și "dBase" pînă la "Norton Utilities", deși aceste programe apelează deseori funcțiile 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 ... ?

Limbaajele de programare procedurale sînt înlocuite de cele orientate obiect. E acesta "leacul miraculos", soluția tuturor problemelor ? Sau trebuie ținut seama de anumite premise, obligatorii ? Și de ce OOP ?

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

În mod cert, niciuna din poziții nu este necondiționat corectă sau incorectă. Programele dezvoltate după metodele clasice au un dezavantaj major: întreținerea. Studii efectuate în această direcție au relevat faptul că pentru întreținerea unui program se cheltuiește 60 - 80 % din efortul total depus. Și nu este singurul dezavantaj: programele dezvoltate după metode clasice se modifică extrem de greu: deseori, pentru a rezolva sarcini

ușor modificate, programele trebuie rescrise. Apoi, și gradul de refolosire al modulelor este destul de redus - lucru evident prin lipsa de productivitate în soft față de ritmul de dezvoltare al hardului. O soluție pentru toate acestea se speră a fi limbaajele de programare orientate obiect.

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

Omul gîndește prin noțiuni, care desemnează obiecte - iar în OOP se folosesc "clase" care se concretizează în obiecte. Comunicarea umană e folosită ca model și în OOP - obiectele își trimit unul altuia "mesaje".

Obiectul - noțiunea de bază a filozofiei OOP - reprezintă o entitate

care conține atît un set de date cît și descrierea metodelor pentru manipularea lor.

Pentru că obiectele conțin funcțiile 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 "mărește-te" - etc. Mesajul nu este deci altceva decît o specificare a unei prelucrări asupra obiectului, care trebuie să precizeze:

- obiectul căruia i se adresează
- operația pe care obiectul trebuie să o execute
- parametri actuali necesari operației

Obiectul reacționează univoc la mesaj; despre polimorfism se vorbește cînd obiecte diferite reacționează în mod diferit la un același

## Realitate

Notiuni

se reprezinta prin

sint ordonate ierarhic

entitati lexicale

denumesc

Obiecte

aflate in relatii de proprietate, apartenenta, inrudire,functionale, etc.

entitati materiale

## Reprezentare

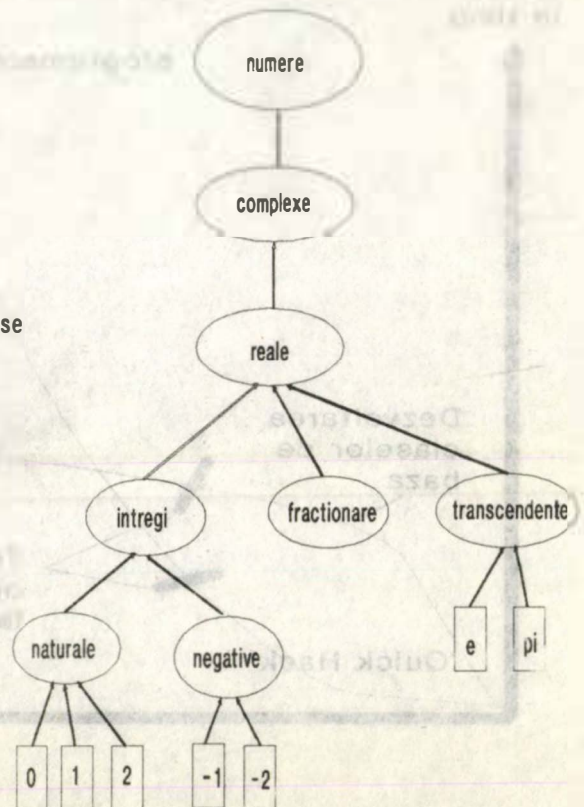
Clase

mostenesec de la supraclase resp. transmit la subclase

descriu

Instantieri

sint legate intre ele prin relatii



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

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

Avantajele OOP - independența reprezentării de implementare, încapsularea, ușurința 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 aplicația do ită din obiecte existente. Ceea ce înseamnă într-adevăr o revoluție pentru programarea clasică.

Procedeele pare foarte simplu - și așa și este. Totuși, cîteva dezavantaje majore ale OOP nu pot fi trecute pur și simplu cu vederea.

Programele OOP nu sînt capabile de multitasking; utilizarea de cod uzual este dificilă; pentru aplicații în timp real, există probleme serioase legate de eficiența codului; programele consumă multă memorie, iar pentru cei implicați în proiectare e necesar un efort semnificativ de școlarizare și/sau acomodare .

Dezavantajul major este însă în mod evident numai temporar: există încă puține "cutii de scule" (toolboxes) și 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 aplicația dorită. Citîndu-l pe Georg Heeg: "oricum ar fi, este o mare diferență între a învăța atît la elefant cît și la pisică sau girafă că gîtul este format din 7 vertebre și a învăța acest lucru o singură dată, la nivelul "mamifer".

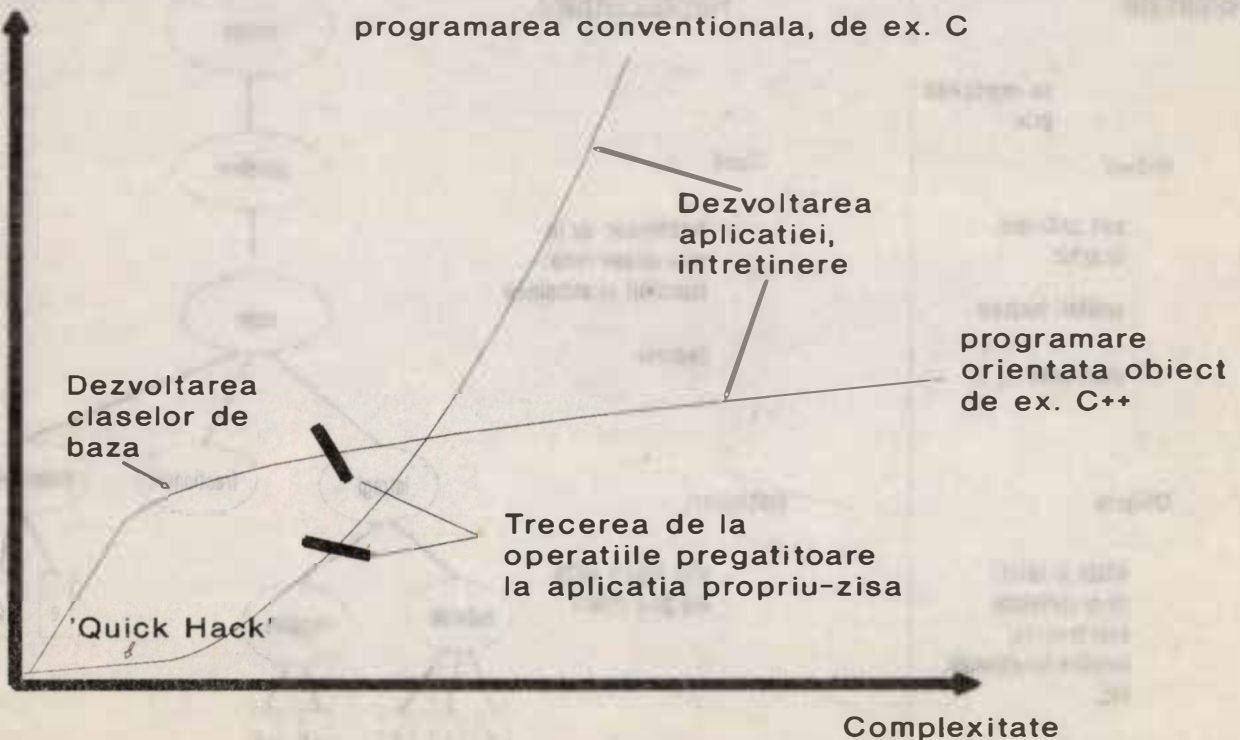
Pentru că, acum, principala muncă o constituie definirea claselor de bază. Se speră ca în viitor productivitatea față de programarea clasică să crească între 10 și 100 de ori. E de ținut seama de acest lucru; mai ales pentru echipe de proiectare mai mari, investiția în viitor merită.

Experții consideră că principalele domenii în care se vor utiliza OOP vor fi suprafețele utilizator grafice, aplicațiile tehnice și sistemele expert, mai puțin bazele de date, procesele numerice și aplicațiile în timp real.

(Iosif Fettich)

Efort  
in timp

programarea conventionala, de ex. C



## “C” - Cuceritorul

*Ce anume determină atracția crescândă 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 achiziționați o aplicație gata scrisă (programată) și o folosiți ca atare.

Dar dacă vă prinde microbul? Vreți să știți cum anume lucrează calculatorul, aveți idei noi pentru aplicația cu care lucrați, vreți să vă scrieți singur alta - pe scurt, vreți să programați. Primele semne de întrebare: în ce limbă să comunicați cu calculatorul? Primele ceretări vă abandonează într-o adevărată junglă de denumiri: Cobol, Fortran, Algol, Pascal, Assembler, Modula, Lisp, PL/I, C, Ada etc. Ce să alegeți - și de ce?

Vi se recomandă Basic, ca fiind extrem de ușor de învățat. Încercați - și într-adevăr, primele succese nu se lasă așteptate. Repede aveți o serie întreagă de programe care fac ceea ce v-ați propus.

Pretențiile vă cresc acum, scopurile pe care vi le-ați propus devin mai complexe, începeți să nu aveți răbdare cînd calculatorul "gîndește" prea mult. Entuziasmul inițial pentru Basic scade pe zi ce trece: constatați că programele mai complicate devin imposibil de descifrat la cîteva luni după ce le-ați terminat - și doar era totul așa de clar cînd ați 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 vroiați să-i faceți o demonstrație - ș.a.m.d. Dezamăgit, căutați un limbaj de programare mai bun - de data aceasta un compilator, pentru a obține viteze de execuție convenabile.

Cu 3 cerințe mai clar formulate acum (structurare, viteză de execuție, portabilitate) intrați din nou în "pădurea" limbajelor de programare. Sub marcajul "structurat" găsiți o cărare care vă duce pe lîngă Pascal, Ada, Oberon, Modula-2 și C. Insistînd asupra vitezei de execuție a codului produs, rămîn doar Modula-2, Assembler și C. Iar dacă țineți ca programele Dvs. să fie și 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 inițial pentru calculatoare mari, "C" s-a născut în 1972. Între timp, s-a răspîndit atît de mult încît practic nu are concurenți și aproape orice tip de calculator are (cel puțin) un compilator de C disponibil.

Mulți din producătorii de soft comercial - destinat vînzării - nu mai folosesc alte limbaje de programare, pentru că astfel nu numai că viteza de execuție a programelor crește, dar și 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. Trebuie să eventual adaptate numai caracteristicile speciale - grafică, intrări/ieșiri. Astfel încît nu este de mirare că un produs-program apare pe piață simultan pentru mai multe tipuri de calculatoare. În afară de aceasta, compilatoarele moderne de C generează un cod atît de bun și de eficient încît în foarte rare cazuri este necesară intervenția la nivelul limbajului 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 și eficiența lui. Standardul - terminat în 1988 - se referă atît la caracteristici ce țin de

tehnica programării cît și 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-și scrie programele structurate, clar, sau să scrie programe supereficiante, dar ilizibile. Deoarece există multe programe și 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 și altfel - și programatorul e liber să-și urmeze propriile opțiuni. Flexibilitatea deosebită a "C"-ului se datorează și simplității nucleului limbajului. Pentru cine are noțiuni de programare, învățarea "C"-ului nu ridică probleme - instrucțiunile și cuvintele cheie sînt foarte puține. Chiar atît de puține încît foarte repede poate să apară întrebarea: "bine, bine, dar cum să programez ceva într-un limbaj care nu are nici măcar niște instrucțiuni de intrare/ieșire?"

Secretul îl constituie filozofia "C"-ului - toate funcțiile sînt externe limbajului și sînt accesibile prin bibliotecă. Cele mai des folosite funcții (apeluri sistem) sînt adunate într-o bibliotecă standard care se atașează tuturor compilatoarelor C. Ea conține rutine pentru intrări/ieșiri generale, pentru gestionarea memoriei și tratarea șirurilor. Conținutul bibliotecii standard e prescris cu strictețe de standardul ANSI, astfel încît orice compilator "C" care corespunde standardului dispune de funcții identice (cel puțin ca dotare de bază).

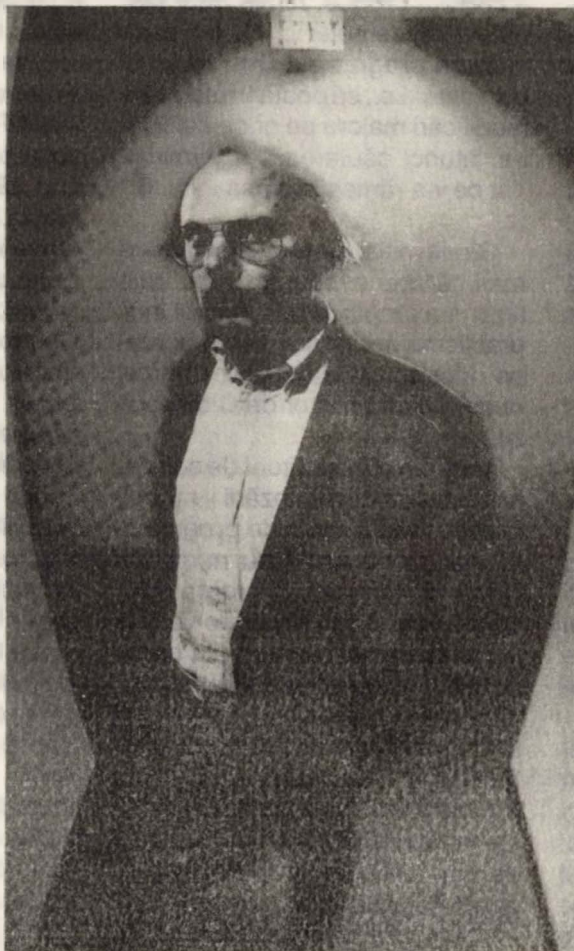
Principiul bibliotecilor de funcții este foarte avantajos pentru "traducerea" unui program în cod de

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

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

Colecțiile externe de funcții mai au un avantaj: deoarece o astfel de bibliotecă constă doar dintr-o înșiruire de funcții C compilate - bineînțeles că și funcțiile bibliotecii standard sînt formulate în C! - este foarte ușor a extinde acest set de funcții cu funcții noi sau a înlocui unele funcții cu altele mai bune. Astfel că tot "vocabulary" unui sistem C poate fi cu ușurință adaptat propriilor nevoi sau dorințe.

În afară de această, prin "scoateră" funcțiilor din limbaj, dependența de mașină a compilatorului dispare aproape complet, deoarece această dependență rămîne limitată la cîteva funcții de bază. În conformitate cu filozofia Unix, după care orice operațiune de intrare/ieșire (fie ea la ecran, imprimantă, tastatură sau altceva) este echivalată unei citiri/scrieri într-un



Bjarne Stroustrup, "tatăl" C++ - ului

fișier, aceste funcții de bază constau de regulă din cîteva mici rutine pentru a asigura intrare/ieșire la / de la diverse echipamente periferice. Astfel încît la portarea unui compilator C pe un alt calculator nu trebuie adaptate decât aceste rutinele respectivului sistem de calcul. Prin structura deschisă a "sistemului C", nu numai programele scrise în C sînt ușor portabile, ci și sistemul în ansamblu poate fi foarte ușor transpus pe alte calculatoare.

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

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

Îmbinarea avantajelor structurale ale "C"-ului cu OOP au născut C++, care se răspîndește într-un ritm incredibil. Rămîne de văzut în cît timp C++ va reuși 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 câștigător. Dar cine nu vrea să renunțe la creațiile unei din cele mai mari case de soft din lume și vrea să rămînă într-adevăr 100% compatibil, va lucra și 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 reușite ale firmei Borland, noul mediu de dezvoltare se prezintă cu multe noutăți. Cele mai importante dintre ele vi le prezentăm în continuare.*

În primul rând, desigur, ++. Corespunzând 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 ușoară este trecerea de la C la C++ nu numai pentru că C++ este un superset adevărat al C-ului, dar și pentru că propunerile AT&T au fost preluate între timp de mai toate compilatoarele existente pe piață și disponibilizate în bibliotecile adiționale.

Miezul C++-ului îl constituie noțiunea de "clasă". Cele trei aspecte fundamentale ale programării orientate obiect - încapsularea, moștenirea și polimorfismul - sînt realizabile, de altfel, și cu mijloace clasice, dar numai cu prețul unui efort deosebit și într-un mod destul de "fragil" - e de ajuns ca un programator să nu respecte convențiile stabilite, ca tot "OOP"-ul să nu fie decît o frumoasă fațadă. Nu așa se întîmplă în C++. Definițiile de clase permit subdiviziuni consecvente ale funcțiilor și datelor, care nu pot fi modificate decît de funcții interne claselor. Ceea ce duce nu numai la o modularitate mai ridicată, dar ușurează 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 moștenesc în mod automat toate domeniile de date și funcțiile clasei vechi, care în mod automat devine parte a clasei noi. Și în definirea convențională de structuri se pot concepe construcții ierarhizate de tipuri de date;

în OOP însă, încapsularea se păstrează, rrai mult, funcțiile își păstrează valabilitatea și permit în continuare prelucrarea datelor clasei. În C-ul convențional, ar fi necesare o serie întregă de "acrobații" lingvistice pentru a putea prelucra cu o singură funcție toate tipurile derivate de date.

Ca o consecință directă a "moștenirii" apare polimorfismul funcțiilor. "Polimorfism" vine din greacă și înseamnă "multe forme": este proprietatea de a da unei funcții un nume care va fi folosit uniform în toată ierarhia de obiecte, fiecare obiect fiind însă liber să și implementeze funcția în modul cel mai adecvat. Astfel, "moștenirea" duce la preluarea numelui funcției, nu neapărat însă și la cea a implementării. La nevoie, aceasta poate fi refăcută și duce atunci la înlocuirea codului moștenit. Pe de altă parte, se poate păstra și vechea versiune a funcției. Compilatorul de C++ rezolvă aceste referințe complexe la momentul execuției, parcurgînd ierarhia de obiecte de la obiectul actual și pînă (cel mult) la rădăcină pentru a găsi o implementare a funcției necesitate. Cel tîrziu la rădăcină funcția este găsită - altfel, ea nu ar fi putut fi moștenită.

Dar nu toate inovațiile aduse de C++ sînt legate de programarea orientată obiect. Și domeniul convențional este restucturat. Astfel de exemplu biblioteca standard

stdio.h este înlocuită cu stream.h. Comentarii de un singur rînd pot fi marcate prin "//". Parametri funcțiilor pot fi transmiși după dorință ca valoare sau prin adresă. Etc.

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

Noutatea care "sare în ochi", pentru cei familiarizați cu versiunile mai vechi ale compilatorului, este mediul de dezvoltare, numit IDE (Integrated Development Environment). Noua suprafață corespunde standardului SAA și poate fi folosit (în sfîrșit) și 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 îmbunătățiri interne. Astfel, pot fi deschise mai multe ferestre simultan, mărirea lor se poate defini, poziția lor pe ecran se poate fixa după dorință. Texte marcate pot fi copiate dintr-o fereastră într-alta prin intermediul clipboard-ului.

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

O altă noutate o constituie "TEML" (Turbo Editor Macro Language), care permite reconfigurarea sau modificarea editorului și a tastelor sau combinațiilor de taste asociate anumitor comenzi. Adică: dacă în locul editorului standard preferați să lucrați cu un editor ca "Emacs" sau "VI", prin intermediul macrocomenzilor se poate modifica editorul în așa fel încît acesta să se prezinte ca cel cu care sînteți obișnuiți. Destul de puternic, acest macrolimbaj permite asocierea unor mici "programe de editare" la anumite taste sau combinații de taste, fără a ajunge totuși la ca-

## Noutăți la Turbo-C++ 1.0

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

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 încât să lucreze în ferestrele normale cu care operează acesta, păstrând 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 decât la tehnicile uzuale, care "descarcă" module nefolosite, lăsând 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 întreruperi 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 cronometronumărător soft, care poate urmări gestiunea resurselor critice ale calculatorului. Se pot cronometra anumite secvențe de program, se pot număra accesesele 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 câte 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 rămâne 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 a memoriei - debugger-ul nu mai folosește în memoria principală decât 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, permițând lucrul cu clase și păstrând în felul acesta "rotunjimea" acestui mediu C++.

Inovația care în mod cert va câștiga 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, artificio 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

## Comparație: Turbo-C++ - MS-C 6.0

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

\*Driverile pentru SuperVGA trebuiesc achiziționate separat

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

asemănător și 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 performanțe.

Și 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. Și și aici, Turbo-C++ nu are motive să se ascundă: chiar dacă C++ produce un overhead destul de substanțial, codul produs de

Turbo-C++ încă este semnificativ mai rapid decât 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 privește optimizările de cod la momentul execuției, compilatorul Microsoft este, probabil, imbatabil la ora actuală.

Unul din elementele de strălucire ale oricărui compilator Borland a fost și rămîne biblioteca grafică. BGI - "Borland Graphic Interface" - lucrează cu driver-e ce pot conlucra cu aplicații 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 neprețuit. Pe o structură tip hipertext, help-ul permite o găsim foarte rapidă a informațiilor necesitate. Și nu numai atât: exemplele de utilizare a funcțiilor pot fi copiate din help în programele utilizator, prin intermediul clipboard-ului, apăsînd doar cîteva taste - sau făcînd două - trei mișcări și clic-uri cu mouse-ul.

Mai trebuie amintite și documentațiile de utilizare - foarte bune, utilizabile atît de către începători cît și de către profesioniști.

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

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

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

(ing. Iosif Fettich)

### Teste de viteză pentru "large modell"

	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

### Concurenții dintr-o privire

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

# Paradigma Paradox

*Înainte să programați în PAL (Paradox Application Language), trebuie să înțelegeți modelul Paradox.*

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

Acest articol se adresează ambelor grupuri de utilizatori. Paradox are o creștere considerabilă în acest moment, în special în rândul corporațiilor. Neașteptat Paradox este SGBD-ul (Sistemul de Gestionare a Bazelor de Date) relațional, pentru PC-uri, cel mai vândut în rândul corporațiilor importante. Un institut de cercetări afirmă că Paradox a acaparat mai mult de 44% din această piață, în timp ce dBase III împreună cu dBase IV ating doar 35%, iar Fox și R:Base au sub 10%. (Datele se referă la S.U.A.)

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

Dar Paradox a rămas um mister pentru o mulțime de oameni. Mulți programatori și utilizatori ai altor SGBD-uri nu înțeleg cu claritate stilul și modelul Paradox. Paradox are nemeritata reputație că ar fi o unealtă de calibrul redus recomandabilă doar utilizatorilor finali. De fapt, este un mediu de dezvoltare complet, cu posibilitatea de a genera machete (forme) și rapoarte, cu un QbE (Query by Example - interogare prin exemplu) pe care toți doresc să-l copieze și cu un limbaj puternic care permite proiectanților să acceseze, prin intermediul unor construcții de programare tradiționale, orice element al mediului interactiv Paradox.

## Să înțelegem modelul Paradox

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

O comparație a modelului Paradox cu modelul Xbase ne va ajuta să clarificăm diferențele.

## Modelul Xbase

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

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

Mecanismul orientat articol este manipulat direct de limbajul de programare Xbase, care este, în toate privințele, un limbaj tradițional foarte puternic. El, de asemenea, este optimizat pentru a manipula articole, utilizând sau nu indexuri, în funcție de dorința utilizatorului sau programatorului.

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

**Fig. 1**

### Modelul Xbase.

Limbajul Xbase este situat la mijloc între procesorul de articole și interfața utilizator. Programul manipulează articole la nivelul de jos iar interfața utilizator trebuie codificată.

### Interfața utilizator

#### Limbaj de programare

#### Procesor orientat articol

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

Din cauză că interfața utilizator și limbajul de programare sînt legate foarte ușor în acest fel și deoarece interfața utilizator se află în exteriorul stratului de cod, este simplu pentru produsele Xbase să aibă interfețe utilizator diferite. În fapt Ashton-Tate a schimbat interfața utilizator a dBase-ului de cel puțin trei ori, iar diferitele clone, cum ar fi FoxPro și dBase, își au propriile interfețe utilizator.

Funcția interfeței utilizator, în modelul Xbase, este de a crea și de a manipula cod. Cum se ajunge la nucleul limbajului, fie printr-o interfață utilizator (front end), prin intermediul unui limbaj, sau scriind direct într-un editor, practic este lipsit de importanță.

## Modelul Paradox

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

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

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 aplicații cu baze de date utilizînd formatele fișier Paradox, cu un suport complet pentru indexuri, protecție prin parolă și protocoale de blocaj multiutilizator. Paradox Engine nu are capabilități de interfață utilizator, programatorul trebuind să-și scrie propria interfață

sau să utilizeze una dintre bibliotecile puse la dispoziție.

**Orientarea tabelă.** Prima zonă majoră în care Paradox diferă de Xbase este mijlocul liagramei. Imediat în exteriorul stratului procesorului de articole se află un procesor orientat tabelă, care lipsește cu desăvîrșire 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 relațional, 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 relaționale. "Inteligența" înglobată permite Paradox-ului să selecteze o abordare eficientă la nivelul procesării articolelor care stă la baza tuturor operațiilor pe seturi.

Această optimizare analizează strategia corectă pentru o operație pe seturi de fiecare dată cînd este executată. Optimizarea ține cont de memoria disponibilă și de celelalte resurse, mărimea tabelelor, prezența indexurilor, și de o mulțime de alți factori. O aceeași

cerere poate fi executată utilizînd abordări diferite de fiecare dată.

În consecință Paradox execută comenzile relaționale 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 soluție curată și 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 înțelege 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 și machetele și 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 și a lista rapoarte.

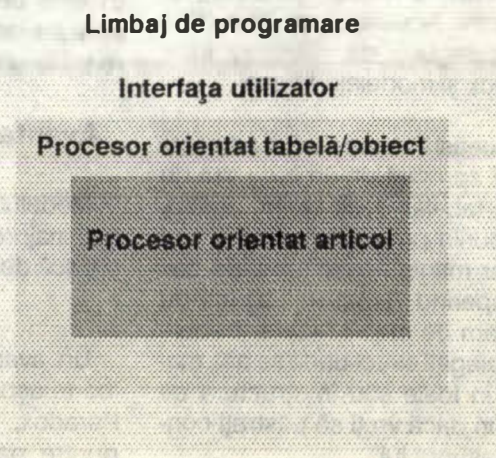
Aceasta este a doua categorie de diferențe între Xbase și Paradox: în Xbase se consumă mai mult timp manipulînd cod pentru a executa operații de bază, care în Paradox sînt implementate ca obiecte predefinite.

Deși Paradox nu are nici una din facilitățile limbajelor orientate

Fig. 2.

### Modelul Paradox.

Modelul Paradox permite PAL să manipuleze construcții de nivel înalt incluzînd tabele (cu operații la nivel de set) și toate facilitățile interfeței 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 și obiecte binare discrete care pot fi manipulate direct în loc să trebuiască procesate sute de linii de cod.

**Interfața utilizator.** Al treilea nivel în modelul Paradox îl constituie interfața utilizator. Aceasta constă din sistemul de meniuri inelar extensiv, incluzînd și capabilitățile query by example, editorul de machete și de rapoarte și diferite module minore, cum ar fi sistemele grafic, de sortare și de protecție prin parolă.

Interfața utilizator Paradox este locul în care se creează obiectele Paradox, incluzînd tabele, indexuri, machete, rapoarte, setări de imagini și condiții 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" interfeței utilizator. Programele Paradox (script) manipulează interfața utilizator ca și cînd ar apăsa tastele corespunzătoare în locul Dvs. A plasa o tabelă în spațiul de lucru presupune a executa aceleași operații cînd programați, care ar trebui executate și interactiv. Aceasta este cea de-a treia categorie de diferențe între Paradox și 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 înțelegi exact unde se află macroul în foaie sau în structura de meniuri dacă vrei să păstrați controlul asupra lui.

În cazul Paradox-ului se întîmplă același lucru. Pentru a executa cu succes programele Dvs., trebuie să înțelegi exact ce situație a creat programul Dvs. în interfața utilizator și în spațiul de lucru.

De ex., adăugînd datele dintr-o tabelă la cele din altă tabelă se schimbă tabela curentă din spațiul de lucru. Trebuie să fii conștient de aceasta și programul Dvs. va trebui fie să îndepărteze această tabelă din imagine, fie s-o utilizeze în continuare ca tabelă curentă. În mod similar dacă tastați `DO-IT!` [`F2`] într-o machetă multitabelară, cursorul sare înapoi în tabela proprietar (master). Interactiv, dacă doriți să rămîneți în tabela de detaliu, va trebui pur și simplu să tastați [`F3`] sau [`F4`] pînă cînd ajungeți din nou în tabela în chestiune. Programul (script-ul) Dvs. va trebui să facă același lucru.

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

De fapt PAL este un limbaj de programare complet, permițînd variabile dinamice, funcții și comenzi definite de utilizator, gestiunea memoriei virtuale și un set complet de comenzi, funcții și construcții aparținînd programării structurate. El este un limbaj foarte complet care are deci acces deplin la oricare trăsătură a mediului interactiv.

### Avantaje și dezavantaje

Modelul Paradox are două avantaje majore față de modelul Xbase și două dezavantaje relativ minore.

Un avantaj este acela că este foarte ușor să creezi prototipuri în Paradox. Puteți naviga interactiv printre pașii necesari, să salvați

cheile tastate și codul rezultat este *același* program pe care l-ați fi putut scrie cu mîna cu ajutorul unui editor. Paradox permite un număr de moduri de a salva ceea ce ați făcut într-un program, care ulterior poate fi executat sau combinat în programe mai mari și 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 spațiul de lucru valori de validare pentru cîmpul curent. Programul navighează prin toate aceste meniuri pînă cînd găsește valorile pe care le caută, le salvează în variabile și le afișează pe ecran.

Un dezavantaj al modelului Paradox îl reprezintă performanțele mai slabe pe care le obțin programele PAL trecînd prin interfața utilizator.

Reprezentarea suplimentară a interpretărilor scade viteza de execuție 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 afișa o tabelă și a utiliza o machetă predefinită sînt:

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

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

(continuare în pag. 43)

(R.M.)

# Static sau dinamic ?

## OOP - Partea a treia

*Tema principală a acestei a treia părți 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 părți ale acestui curs au fost în totalitate statice. Aceasta înseamnă că încă din momentul compilării, se rezervă spațiul de memorie necesar și se rezolvă cu ajutorul acestor metode toate referințele.

Facilitatea de a redefini pentru obiecte derivate, metode noi având același nume ca și cele vechi, permite adaptarea metodelor în funcție de necesități.

Definiția 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, conține toate metodele necesare pentru inițializarea, desenarea, ștergerea și mutarea unui punct.

Această miniierarhie poate fi extinsă după dorință. 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 același nume în funcție de caracteristicile imaginii grafice se vor ascunde desigur implementări diferite.

Se poate spune și că fiecare obiect grafic posedă caracteristica de a se putea desena singur pe ecran. Cuvântul Deseneaza este utilizat pentru a desemna toți algoritmi posibili de desenare. Acest lucru este ușor de înțeles. Dacă din obiectul Gr\_Punct este derivat, de ex., obiectul Gr\_Cerc, atunci metodele InIt și Deseneaza vor trebui reprogramate, fără îndoială. Un cerc necesită ca parametru pentru inițializare și raza. Rutina de dese-

nare a unui punct este alta decât cea pentru desenarea unui cerc.

Dacă se programează static atunci și metodele Muta\_Abs sau Muta\_Rel trebuie redefinite, cu toate că secvența de cod nu se modifică. Acest lucru nu s-ar întâmpla dacă în momentul compilării, adresele rutinelor **Sterge** și **Deseneaza** ar rămâne deschise. Atunci ar exista o posibilitate ca în momentul execuției să se stabilească metodele **Sterge** și **Deseneaza** care să fie folosite. Tocmai această posibilitate o oferă programarea orientată obiect. Turbo-Pascal, începând din versiunea 5.5, sprijină mecanisme pentru tratarea metodelor virtuale.

{ Listing LIST13.PAS }

Pentru o mai bună înțelegere vom mai numi odată diferențele dintre metodele statice și cele virtuale. În cazul metodelor statice, în momentul compilării se stabilește exact ceea ce trebuie să facă. În cazul metodelor virtuale decizia asupra a ceea ce trebuie făcut rămâne deschisă.

Cum funcționează aceasta ? Unit-ul Gr\_Obj conține noile definiții. Cuvântul rezervat **Procedure** înaintea declarațiilor Punct.Init și Cerc.Init este înlocuit cu cuvântul **Constructor**. Un constructor este un antet special de procedură, care rezolvă câteva inițializări interne pentru mecanisme virtuale. El este necesar deci pentru a putea lucra cu metode virtuale. În afară de aceasta constructorul trebuie apelat neapărat înaintea primului apel al unei metode virtuale. El stabilește referințele necesare cu ajutorul așa numitei **VMT** (Virtual Methods Table - tabela metodelor virtuale). În VMT pe lângă mărimea obiectelor sînt conținuți și pointeri la codul metodelor obiectului.

Dacă nu este apelat Constructor, în locul de ramnificație spre o metodă virtuală se află o altă

adresă. Urmarea poate fi o blocare a sistemului.

Compilerul nu are nici o posibilitate de a verifica succesiunea apelurilor metodelor. Este deci sarcina programatorului de a rezolva acest lucru. Singurul ajutor aflat la dispoziție în Turbo\_Pascal este comutatorul { \$R+ } .

Acesta, gîndit de fapt pentru o verificare a domeniilor, verifică la toate metodele virtuale starea de inițializare a instanțelor. În cazul în care anumite inițializări lipsesc, în timpul execuției, va fi dat un mesaj de eroare domeniu și execuția programului va fi întreruptă. Acest comutator este foarte util în faza de dezvoltare a unei aplicații. 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 același 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 declarațiile metodelor pentru toate metodele cu același nume ale unei ierarhii de obiecte.

Listingul LIST13.PAS clarifică avantajele pe care le aduce utilizarea metodelor virtuale. Obiectul Punct definește toate metodele care sînt necesare pentru manipularea unui obiect grafic. Pe lângă Constructor pentru inițializare sînt implementate și rutinele pentru desenarea, ștergerea și 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 și o metodă specială **Sterge**. Metodele pentru mutarea absolută și relativă însă nu mai trebuie definite și pentru urmași.

Obiectul **Punct** conține, să zicem așa, o metodă mamă pentru mutarea tuturor obiectelor grafice care sînt urmașii obiectului **Punct**. Pentru acest scop în rutina **Punct.Muta\_Abs** și **Punct.Muta\_Rel** numele metodelor **Deseneaza** și **Sterge** nu conțin nici o referință 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 execuției 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 același mod în care pot fi tratate și variabilele. Unit-ul **Gr\_Obj** demonstrează cum pot arăta astfel de declarații. Tipurile de pointeri **Gr\_PozitiePtr**, **Gr\_PunctPtr** și **Gr\_CercPtr** permit crearea și îndepărtarea unor obiecte din heap (memoria disponibilă). Pentru acest scop procedurile Turbo-Pascal **New** și **Dispose** au fost astfel extinse încît să se poată referi și la obiecte. Pentru utilizarea lui **New** există două posibilități:

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

În ambele cazuri se rezervă suficientă memorie în heap pentru a permite o instanțiere 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 spațiului de memorie necesar, este apelat simultan și constructorul **Init**, și ne asigurăm în acest fel că inițializările necesare vor fi efectuate. În afară de aceasta programatorul economisește o instrucțiune, deoarece în cazul primei versiuni mai trebuie declarat și constructorul.

Ca și toate celelalte tipuri de date tradiționale, care sînt depuse în heap, și obiectele pot fi îndepărtate din memorie cu ajutorul procedurii

**Dispose**. Dar nu întotdeauna apelul lui **Dispose** este cea mai bună soluție. Un obiect individual poate conține pointeri la alte structuri de date sau obiecte, care ar trebui eliberați într-o anumită ordine. În acest caz este de bun simț ca această eliberare a memoriei să fie cuprinsă într-o singură metodă.

Turbo-Pascal oferă această metodă prin intermediul lui **Destructor**. Cuvîntul rezervat **Destructor** înlocuiește cuvîntul **Procedure**. Apelul lui **Destructor** unește eliberarea zonei de memorie rezervate cu toate celelalte operațiuni de "curățenie" necesare.

Cu aceasta devine evident că destructorii acționează asupra obiectelor dinamice. Ei își găsesc întrebuițarea 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 cunoaște mărimea zonei de memorie ce va trebui rezervată. Aceasta explică de ce utilizarea individuală a lui **Dispose** este interzisă.

Un destructor consultă VMT-ul instanței apelante. Acolo este memorată dimensiunea în bytes a obiectului. O eliberare reală a memoriei nu se obține 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 instrucțiunea **End** este atinsă, dimensiunea instanței este transmisă procedurii **Dispose** care preia munca de eliberare propriu-zisă. Metoda **Destructor** însăși poate să fie goală, eliberarea memoriei se produce totuși. Ea se realizează practic în fundal.

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

Exemplul de program "OOP\_Incercare Virtuala" scoate în evidență mecanismele virtuale. Toate declarațiile de obiecte sînt plasate în Unit-ul **Gr\_Obj**. Variabilele poin-

ter "UnPunct", "UnCerc" și "OLinie" pot indica instanțe ale acestui tip de obiect. Ele trebuie deci create cu **New**. În liniile:

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

este repetată sintaxa extinsă a procedurii **New**. Cu ajutorul acestor instrucțiuni se rezervă spațiul necesar de memorie pentru instanțe și în același timp este apelat constructorul corespunzător. Apelul virtual propriu-zis este realizat prin sarcina **Muta\_Rel** a instanței "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 moștenită de la obiectul **Gr\_Punct**. În rutina **Gr\_Punct.Muta\_Rel** sînt prezente metodele **Sterge** și **Deseneaza**. În acest caz se stabilește în momentul execuției dacă vor fi utilizate metodele **Sterge** și **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 părți ale acestui curs pot fi rezumate în cele ce urmează: Obiectele sînt tipuri de date definite de utilizator, care înglobează structuri de date și metodele corespunzătoare lor. Accesul la date pe o altă cale decît prin intermediul metodelor nu este posibil. Datele sînt deci încapsulate. Tipurile de obiecte pot lăsa moștenire altor obiecte toate caracteristicile lor. În acest mod pot fi construite structuri ierarhice arborescente.

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

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

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

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

### Metode virtuale

Ele sînt declarate tot la fel ca și metodele statice, doar că se

adaugă cuvîntul Virtual. Toate metodele care vor purta acest nume vor trebui declarate de asemenea virtuale. Ele prezintă față de metodele statice avantajul că abia în momentul execuției se decide care metodă anume trebuie utilizată. Acest lucru este dependent de tipul instanței apelante.

### Constructor

este un cuvînt rezervat, care înlocuiește cuvîntul rezervat Procedure. O rutină constructor inițializează mecanismul metodelor virtuale și trebuie apelată înainte de apelul lor. El stabilește o legătură între instanța care apelează constructorul și VMT-ul obiectelor.

### Virtual Methods Table (VMT)

memorează dimensiunea obiectelor și conține pentru fiecare metodă virtuală un pointer la codul ei.

### Destructor

Un destructor este o metodă specială, pentru eliminarea de obiecte din memorie și 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 își îndeplinește funcția chiar și atunci cînd nu conține nici o instrucțiune.

(R. M.)

### Paradigma Paradox

(continuare din pag. 40)

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

```
[ F10 ] afișează meniul
"view" selectează această
opțiune a meniului
CLIENT tastează numele
tabelei în dreptul
prompterului
```

```
[ Enter ] acceptă intrarea
** tabela apare pe ecran
```

```
[ F10 ] afișează meniul
"Image" selectează această
opțiune a meniului
"1" selectează această
măchetă din lista
măchetelor
disponibile
```

\*\* imaginea de pe ecran se modifică de la tabelă la măchetă.

Performanțele obținute lucrînd în mediul interactiv sînt îmbunătățite de un alt punct forte al Paradox-ului, managerul memoriei virtuale, care manipulează eficient aplicațiile care sînt mai mari decît memoria aflată la dispoziție. Paradox ajustează dinamic memoria asignată tabelelor, buffer-elor, stivelor și modulelor programului, în funcție de resursele hardware

disponibile și de operațiile executate de către utilizator.

Plasarea în Paradox a limbajului PAL deasupra interfeței utilizator este restrictivă în alt mod: este dificil pentru Borland să modifice interfața utilizator. Din cauză că programele sînt atît de strîns legate de interfața 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 și delicat.

### Deci care este concluzia ?

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

Proiectanții, oricum, *trebuie* să țină cont de relația cu modul de reprezentare. În Paradox, de ex., un programator poate uza de reprezentarea tabelelor pentru a obține maximum de productivitate. În Xbase trebuie manipulată reprezentarea articolelor pentru a obține aceeași funcționalitate.

Probabil că cea mai mare diferență este aceea că în Paradox programatorul *trebuie* să se plase-

ze în afara mediului interactiv. Paradox este un program "modal". Atunci cînd navigați prin meniuri, selectînd, acționînd și manipulînd obiecte, Paradox navighează prin diferite moduri de operare. Fiecare mod este definit de meniul de opțiuni care este disponibil atunci cînd tastați [ F10 ] .

Anumite acțiuni 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 tastați [ Alt-F9 ], (sau determinați programul să tasteze [ Alt-F9 ], atunci intrați în modul "CoEdit", în care acțiunile posibile sînt cele disponibile în acest meniu; copierea unei tabele nu este una dintre aceste opțiuni.

Acest lucru nu este neapărat "mai bun". Avantajele Paradox-ului sînt legate de alte aspecte, cum ar fi crearea de prototipuri și 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 înțeleagă mediul interactiv și 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;

```

{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_Mode,
'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 incepiind
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 : Integer);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 și asociativitate

## Rețele neuronale

### Partea a III-a

*Sistemele expert acumulează fapte și reguli. În bazele lor de cunoștințe; pornind de la acestea și folosind strategii, ele raționează.*

#### Cuprinsul cursului:

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

Sistemele expert simulează cunoașterea umană în anumite domenii. De aceea se mai numesc și sisteme bazate pe cunoștințe. În general sînt compuse din componente diferite. Pe de o parte, baza de cunoștințe, în care sînt stocate cunoștințele ce urmează a fi prelucrate (fapte și reguli). Apoi, mașina inferențială, folosind strategiile de control preprogramate, "raționează", emițînd judecăți proprii bazate pe datele existente și cunoștințele stocate. Pentru o utilizare prietenoasă, acestora li se mai adaugă de regulă o componentă de dialog, care pune întrebări și preia răspunsurile, o componentă de achiziție, care "prepară" știința experților pentru a fi preluată în baza de cunoștințe și o componentă explicativă, care are menirea de a lăsa să se vadă cum anume ajunge programul să își formeze concluziile.

Să considerăm următorul exemplu simplu, care se compune dintr-un fapt și o regulă. Regula este: "orice om merge în poziție verticală", faptul este "Bogdan este un om". Se poate concluziona de aici

că Bogdan merge în poziție verticală, deși acest fapt nu este specificat în mod explicit. El este conținut implicit în baza de cunoștințe.

Spre înțelegere, încă două noțiuni. Orice regulă se compune dintr-o parte condițională și o parte de acțiune:

DACĂ condiție ATUNCI acțiune

În exemplul dat, condiția este "cineva este om" iar acțiunea corespunde "mersului în poziție verticală". De aceea, regula se mai poate formula și

DACĂ cineva este om,  
ATUNCI el merge în poziție 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 rețelelor neuronale în acest domeniu.

Este clar că situații care pot fi descrise exact cu reguli și 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 dificultăți: regula dată nu este general valabilă. Pentru bebeluși de exemplu, ea nu este adevărată. Și mai dificilă devine situația în care ar trebui prelucrate cunoștințe 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 proporții (merele de exemplu nu sînt 100% roșii sau verzi, ci, de obicei, "și-și") sau prin anumite mărimi (1,94m în loc de "înalt").

Cercetarea unor forme adecvate pentru reprezentarea de cunoștințe nesigure în calculator

constituie fundamentul dezvoltării de sisteme IA (inteligență artificială).

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

Cînd și cum se folosesc însă rețelele neuronale pentru sisteme expert? Folosirea lor se recomandă atunci cînd partea condițională a regulilor se compune din mai multe proprietăți sau cerințe (de exemplu: "cineva este om" și "vîrsta lui e mai mare de 2 ani") și cînd pentru partea de acțiune sînt disponibile mai multe variante (de exemplu "merge aplecat", "merge tîrîș", "aleargă", "sare" etc.). Diversele condiții ale regulilor se interpretează ca neuroni de intrare ai unei rețele neuronale, unde fiecare proprietate se reprezintă printr-o valoare numerică (de exemplu numărul de ani pentru vîrstă). Variantele posibile ale părții de acțiune sînt văzute ca și clase diferite și li se atribuie neuroni de ieșire diferite.

Pentru clarificarea modului de lucru vom prezenta aplicarea metodelor neuronale la un sistem expert exemplu. <sup>(\*)</sup>

Scopul: trebuie clasificate niște obiecte pe baza unor proprietăți. În afară de aceasta, se dorește recunoașterea unor reprezentanți ai claselor care nu îndeplinesc 100% condițiile de apartenență. Ca obiecte, fie date următoarele tipuri de fructe: cireșe, portocale, mere, pere, prune, banane, struguri, lămii. Ca proprietăți

	cireașă	portocală	măr	șară	prună	banană	struguri	lămie
are simburii	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 și nu	nu
dulce	da	da	da	da	da	da	da	nu
acru	nu	da	da	nu	nu	nu	da	da
culoare roșie	da	nu	da și 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 și nu	da și nu	nu	da	nu	da
culoare verde	nu	nu	da și nu	da și nu	nu	nu	da	nu

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

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

La culori contează numai cele de la fructul copt, astfel că "verde" de exemplu nu intră în discuție pentru lămii și banane. Bazată pe dimensiunea tabelii se construiește o rețea neuronală care să reprezinte corect proprietățile și clasele. Cele 11 proprietăți sînt interpretate ca 11 celule de intrare. Celor 8 obiecte de clasificat li se asociază cîte o celulă de ieșire, și anume celula de ieșire nr. 1 îi corespunde obiectului "cireașă", celula de ieșire nr. 2 obiectului "portocală" ș.a.m.d. Pentru rețeaua neuronală sînt stabilite astfel structurile de intrare și de ieșire. Rețelele 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 rețea. 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 cunoștințelor. Cum să fie reprezentat "și da, și nu" ca valoare numerică cu care să lucreze rețeaua neuronală?

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

Rețeaua se "antrenează" cu 19 modele de antrenament, care se compun din 8 modele diferite pentru măr, 4 pentru pară, două pentru struguri și cîte un vector pentru cireașă, portocală, prună, banană și lămie. Obiectivul fazei de antrenament constă în modificarea tuturor ponderilor în așa fel încît fiecare din cele 19 modele de învățare să fie clasificat corect. Căci fiecărui vector de intrare îi corespunde un vector de ieșire: astfel de exemplu vectorului de intrare "cireașă" (255, 255, 255, 255, 0, 255, 0, 0, 0,

0) îi corespunde vectorul de ieșire (1, 0, 0, 0, 0, 0, 0, 0, 0) care reprezintă prima clasă de ieșire.

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 tabelii indexate. În acest caz, valorile de ieșire nu ar fi calculate, ci s-ar regăsi ca într-un dicționar. Foarte mici abateri ale "cuvîntului" căutat față de cuvîntul memorat ar duce însă la eșecul căutării.

Deoarece valorile de intrare sînt ponderate la rețelele neuronale, numerele individuale contribuie mai mult sau mai puțin la calculul rezultatului, al vectorului de ieșire. De aceea, valori asemănătoare la intrare produc valori asemănătoare la ieșire. Se vorbește astfel de capacitatea asociativă a rețelelor neuronale. Modele de intrare incomplete se pot astfel clasifica corect sau completa.

Proprietățile asociative trebuie dezvoltate în faza de învățare. Este limpede că acest lucru devine cu atît mai dificil cu cît trebuie reprezentați mai mulți parametri de intrare (neuroni de intrare) și mostre de antrenament. Dacă în plus modele foarte diferite aparțin unei aceleiași clase, procesul de convergență este îngreunat suplimentar. De aceea, mai ales începătorilor li se întîmplă deseori

ca sistemul să fie divergent, deci învățarea să nu aibă succes. Ca rezultat al unui proces de convergență nereușit, rețeaua reprezintă modelul pe o celulă de ieșire greșită, neîndeplinindu-și astfel scopul.

Ca proiectant începător de rețele neuronale, ce-i de făcut? Este iarăși evident că selecția regulii de învățare corecte, mărirea parametrilor potriviți, structura rețelei (numărul de neuroni) și reprezentarea cunoștințelor sînt totdeauna dependente de problemă. Dacă ar exista pentru 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 experiență în utilizarea acestor tehnici pentru a atinge obiectivele dorite. Ceea ce nu e o noutate, desigur: și pentru a bate un cui în perete, trebuie să știi să umbli cu ciocanul, deoarece altfel apar probleme cu degetele.

Pentru a rezolva problema de mai sus a clasificării fructelor, a dus la reușită regula de învățare 1, predefinită, cu parametri predefiniți  $\beta_1 = 1.0$  și  $\beta_3 = 1.0$ , toleranța predefinită  $T = 850$  și condiția de abandon 3, care abandonează în mod automat procesul de învățare cînd pentru fiecare din modelele de învățare sînt satisfăcute condițiile de toleranță impuse.

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

Pentru a "învăța" cele 19 modele, sistemul a avut nevoie de 130 pînă la 270 de iterații. 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 optime menționate mai sus.

După terminarea cu succes a fazei de învățare se poate și trebuie verificată "capacitatea" rețelei. Ceea ce se face modificînd, cu editorul incorporat, valorile unui model existent sau producînd pur și simplu modele noi care sînt preluate apoi spre evaluare (identificare) rețelei, prin simplă apăsare a unei combinații de taste. În felul acesta se pot face rapid și comod studii particulare. Pentru verificarea profesională a capacităților învățate se pot testa și fișiere întregi, în mod automat, și ieși cu rezultatele fie la ecran, fie la imprimantă. Se vede că rețeaua "învățată" e "testată" cu modele necunoscute (ei). Rezultatele nedorite se corectează, preluînd mostrele greșit identificate în fișierul de antrenament și refăcînd faza de învățare, pentru a îmbunătăți astfel performanțele rețelei. Prin aplicarea repetată a combinației învățare/verificare, prin intermediul setului (fișierului) de date de antrenament, se obține o apropiere din ce în ce mai mare de caracteristicile optime dorite ale rețelei.

Proprietatea de asociativitate mai sus pomenită a fost testată pe un caz "dur". Se cere antrenarea unei rețele care să recunoască cuvinte tastate greșit. Pentru acest lucru au fost selectate 12 comenzi DOS: CHDIR, CHKDSK, COPY, DEL, DIR, FORMAT, MKDIR, MODE, PATH, PRINT, RMDIR, TYPE. Ca intrare în rețea se folosește valoarea corespunzătoare ASCII a literei, deci de exemplu pentru CHDIR vectorul (67, 72, 68, 73, 82). Pentru a îngreuna sarcina rețelei, se permit cuvinte avînd pînă la 7 caractere, astfel că CHDIR se poate scrie >\_CHDIR\_<, >\_\_CHDIR< sau >CHDIR\_\_< cu vectorii de intrare total distincți (20, 67, 72, 68, 73, 82, 20), (20, 20, 67, 72, 68, 73, 82), (67, 72, 68, 73, 82, 20, 20). Fiecărui din cele 12 cuvinte i se asociază o celulă de ieșire a rețelei. Scopul este de a antrena rețeaua în așa fel încît celula de ieșire corespunzătoare lui CHDIR să se activeze de exemplu și cînd la intrare

este >CHDIR<. În felul acesta pot fi eliminate erorile de tastare. Pentru a putea aprecia modul de punere al problemei, ar trebui ținut seama și de asemănarea între trei din comenzile selectate: CHDIR, RMDIR și MKDIR. Cafîșier de antrenament s-a pornit pentru început cu cîte două variante deplasate pentru >FORMAT< și >CHKDSK<, respectiv cîte 3 variante pentru celelalte cuvinte, astfel că în total s-au obținut 34 de modele pentru antrenament. Fișierul de test în schimb conține toate variațiile posibile, dar cu o singură literă greșită - în total, 3.856 de cuvinte.

După o învățare reușită, cu 16 neuroni ascunși și regula de învățare 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 fișierul de antrenament, rezultatul a putut fi îmbunătățit: cu 123 mostre de antrenament (și 18 neuroni ascunși) au fost clasificate corect 2818 cuvinte; cu 198 mostre de antrenament (și 20 de neuroni ascunși), 3191.

Pentru memorarea ponderilor necesare la 20 de neuroni ascunși și a informațiilor relative la topologia rețelei sînt necesari numai 4 kbyte de memorie.

Cine își face gînduri relativ la numărul mare de mostre de antrenament, va realiza că interesantele proprietăți ale rețelelor neuronale ar putea fi îmbunătățite prin combinare cu alte tehnici. Cum anume se face acest lucru, veți afla în episodul următor.

(I.F.)

(\*) Acest curs ar trebui desigur secondat de simulatorul la care se fac referiri. Sperăm că experiențele descrise pot arunca totuși o geană de lumină asupra domeniului. chiar și așa, "pe uscat".(N.red.)

## Parolă variabilă

N-ați fost niciodată în situația de a porni sistemul cu un musafir nepoftit în spatele Dvs ? Nu vi s-a întâmplat, mai ales dacă nu tastați foarte repede, ca acesta să "prindă" parola de acces și să trebuiască s-o modificați ?

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

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

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

PASSW nu afișează caracterele tastate, ci doar câte un punct pentru a facilita controlul introducerii lor. Ați greșit un caracter ? Nici o problemă: puteți utiliza liniștit backspace pentru corectarea erorii.

Deci: creați fișierul PASSW.ASM cu ajutorul unui editor, apoi, în ordine, executați comenzile DOS

> tasm passw

> tlink passw

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

Pentru ca PASSW.EXE pe care l-ați obținut să se execute automat după pornirea sistemului, trebuie ca fișierul AUTOEXEC.BAT să conțină liniile exemplificate în PASSW.BAT

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

(Daniel Buleu, Iași)

```

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  dx,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

```

(in) AUTOEXEC.BAT

## Protecția programelor

Cu filtrul prezentat în cele ce urmează se poate proteja orice program EXE sau COM. Se inserează ca primă instrucțiune în fișier instrucțiunea "ret", ce are codul pe un singur octet (0c3h). Efectul este că la apelul programului se execută acest "ret", astfel că programul se termină și se întoarce în DOS. Prin această inserare, sistemul de operare este determinat să trateze și fișierele EXE ca pe cele COM. Prin urmare, fișierele EXE mai mari de 64kbyte care au fost protejate prin MODIF vor anunța printr-un mesaj de eroare că nu încap în memorie, ceea ce altfel oricum însă nu deranjează. Repetarea comenzii MODIF pentru un fișier îndepărtează protecția pusă. Fișierul 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 aplicație BASIC pentru a produce resetarea imprimantei. Pentru aceasta, programul conține o scurtă rutină în asamblor, care este introdusă în memorie folosind instrucțiuni POKE. La lansare trebuie specificată imprimanta (portul paralel) ce trebuie resetat.

(Computer Persönlich, 10/91)

## Ieșire în DOS

De multe ori în unele aplicații am vrea să putem executa comenzi DOS, fără a ieși din aplicația 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.

Îmbunătățirea față de EXECDEMO.PAS este afișarea prompterului (ca pentru \$p\$g). 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 îmbunătățirea programului DOS.PAS prin interpretarea unei comenzi PROMPT cu sintaxa: PROMPT [\$litera], unde prin [ ] am marcat o secvență 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 (Get Env ('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

```

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

### PRNRESET.BAS

## Există o dischetă în unitate ?

Ideea din următoarea funcție 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 + ':\NULL';
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) = 0xf67;
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 virușilor

Cu ajutorul unității SELFCHK ce v-o prezentăm mai jos, un program oarecare Turbo Pascal poate să-și dea singur seama dacă este sau nu atacat de un virus. Unit-ul, care nu exportă nimic, constă dintr-o parte de inițializare care depune o sumă de control în program și 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 diferențe, rămîne la latitudinea utilizatorului să oprească lucrul sau să ignore pericolul. În caz de coincidență, programul se desfășoară 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 PRACITCA';
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 (cicit < 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 (cicit = 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 și HUGE în C

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

Atît pointerii "far" cit și cei "huge" permit adresarea întregului spațiu de memorie. Diferența 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 folosește o rutină mai complexă, care să controleze depășirile de segment. Aceste diferențe sînt demonstrate de programul care urmează. În timp ce pointerul "far" nu "se prinde" la ieșirea din limitele segmentului, pointerul "huge" modifică valoarea de segment.

(Computer Persönlich, 8/91)

Outsourcing

## Păcatele scumpe ale conducătorilor oficiilor de calcul

Președintele avea o bănuială: oficiul de calcul lucrează ineficient. De-abia comanda către o întreprindere externă pentru preluarea prelucrării datelor ("outsourcing") a relevat potențialul de scădere a costurilor. Outsourcing este în multe cazuri calea corectă pentru a evidenția scumpele păcate ale conducătorilor oficiilor de calcul și 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 știut imediat ce este de făcut: i-au dat un alt job, în care nu mai avea nici o legătură cu patronul lui de până acum. Iar oficiul de calcul (secția de prelucrare a datelor) al ramurii din Germania a concernului scandinav Nokia a fost preluat de Programator, de personal care nu a avut până 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 angajați). Programator promite o scădere drastică a costurilor prelucrării datelor după preluare. Conform experienței scandinavilor, cu personal extern pot fi evitate dezvoltări greșite în oficiile de calcul ale întreprinderilor.

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

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

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

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

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

Cu acest concept, Programator vrea să devină și în Germania o casă de soft de frunte. În Suedia, întreprinderea s-a afirmat deja ca prestator de servicii informatice. Cu 2500 de angajați, Programator este cel mai mare consilier de întreprindere și de prelucrarea datelor din Scandinavia. În concernul înființat în 1964, de care aparține și firma de consultanță pentru con-

strucții Jacobsen & Widmark, lucrează circa 4600 de persoane.

De faptul că prelucrarea datelor este mai ieftină ca soluție conformă cu piața decît printr-o secție proprie s-au lăsat convinse pînă acum printre altele societatea de stat pentru telefoane, Svenska Televerket, societatea de asigurări Skandia, concernul electrotehnic Ericsson precum și filiala-mamă a concernului Nokia din Finlanda. Ideea s-a născut în 1979. Atunci Programator a preluat prelucrarea datelor pentru un producător de oțeluri superioare din Karlstad/Suedia. Între timp peste 20 de concerne se numără printre clienți, de la uzina de țesut celular pînă la specialistul în tehnica frigului Frigoscandia.

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

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

(I.F.)

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

## Din scrisorile primite la redacție

„...la CAS' 85 s-a discutat și despre faptul că n-avem timp să așteptăm 20-30 de ani, pînă cînd se recunoaște că neologismele respective s-au încetățenit în limba curentă, astfel ca termenii să figureze în DEX și să nu mai fim acuzați de barbarism tehnic.

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

Cristian Malide,  
București

„Aș dori informații asupra posibilității de achiziționare de soft și documentație de firmă la prețuri rezonabile în lei, dacă există furnizori oficiali de soft în România și care ar fi aceștia.

De asemenea ar fi bine ca revista dvs. să aibă posibilitatea contactării unor eventuali beneficiari ai produselor și aplicațiilor noastre (un fel de piață de soft). În această întreprindere ar fi interesați și furnizorii de hardware, care fără aplicații nu pot exista. Anunțurile noastre nu sînt prea eficiente deoarece revista nu este citită de prea mulți doritori de aplicații ci mai degrabă de specialiști (furnizorii).

### Mica publicitate

Ofer programe și jocuri pentru calculatoare ATARI. Garanțez calitatea înregistrărilor. Tel. 985/10.609.

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

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

Constatînd cu tristețe că nici anunțurile 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 ocupați ca să citească și mica publicitate ne vedem în situația de a nu putea oferi (chiar licita) programe utile uneori multora.

Aș sugera, dacă s-ar putea, să existe o "bursă" a programelor, prin care conducătorii unităților economice să fie atenționați și interesați în achiziționarea de software. Altfel în curînd vom muri de foame."

Năstase Ileana,  
București

*Nu sînt atît de sceptic ca și corespondenta noastră, dar evident nici mulțumit de ce (nu) am reușit. Sînt multe cauzele știute sau bănuite; o mare parte din ele sînt inevitabile, sînt "datul problemei", dar o altă mare parte sînt vina noastră. Dacă este adevărat că anunțurile de vînzare/cumpărare hard și cumpărare soft (aplicații) sînt de regulă mult prea "strînse" în timp ca să nu devină total neinteresante prin publicarea la un termen incert și oricum la distanță mare în timp - lucru pe care încă sperăm că vom reuși să-l contracarăm, astfel încît revista noastră să ajungă în chioșcuri și la abonați la dată fixă în fiecare lună - oricum lipsa anunțurilor de vînzare soft mă nedumerește. Dacă am scris o aplicație și am vîndut-o deja odată sau de cîteva ori, ea a ajuns deci la o anumită maturitate și rotunjime, nu poate fi decît neglijență sau dezinteres în a nu repeta cel puțin de cîteva ori un anunț care să o facă cunoscută în țară, permițîndu-mi să o vînd în zeci sau sute de locuri! Investiția financiară și de timp este derizorie în comparație cu câștigurile probabile. Ideea că revista nu este ci-*

*tită de utilizatorii finali este, după părerea mea, doar parțial adevărată: chiar dacă e clar că nu am să fac reclamă unui program care-l concurează pe al meu, eu voi fi interesat să știu cît mai multe despre programe care să-l completeze sau să aducă alte foloase beneficiarilor pe care îi deservesc, și care vor veni cu atît mai încrezători la mine cu cît li s-a întîmplat mai des să fie mulțumiți 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 ceilalți realizatori de aplicații 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 - profesioniștii în informatică - va trebui să facem un pas sau chiar mai mulți înspre beneficiari, și nu să așteptăm să învețe ei lucrurile pe care noi le știm, 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 obișnuiți să lucrăm - să nu ne mulțumim să vindem o aplicație scump, de cîteva ori, așteptînd apoi ca beneficiarii să se înghesuie bazat doar pe reclama implicită pe care și-o face produsul. Cu "bursa", vom mai vedea, vrem, dar n-avem putere să promitem.*

*Și, în încheiere, scuze tuturor acelor care ne scriu și cărora - deși ar trebui să o facem - nu ajungem să le răspundem în mod individual.*

Să auzim (numai) de bine!

Ioșif Fettich

### În numărul viitor:

- La ce e bun codul de bare
- EMS - referință
- Și totuși, Cobol?...

## *Important !*

Revista "if" își propune editarea unui catalog al întreprinderilor (de stat sau particulare) a căror sferă de preocupări are tangență cu informatica (producători hard, producători soft, depanatori, distribuitori, comercianți, etc.).

Catalogul va conține adresele întreprinderilor, o scurtă descriere a obiectului de activitate al acestora și o prezentare succintă a produselor și serviciilor oferite. Catalogul va fi distribuit prin rețeaua de difuzare a revistei "if", putând ajunge astfel direct la persoanele interesate. Întreprinderile interesate de publicarea acestui catalog sînt rugate să ia legătura cu redacția.

## *Important !*

Căutăm difuzori/distribuitori autorizați pentru difuzarea revistei "if" !.

Condiții avantajoase !

Rugăm persoanele interesate să ia legătura cu redacția.

**Informații la zi din  
lumea calculatoarelor  
personale puteți  
obține numai citind  
regulat revista  
"if" !**

Micro ATCI

C.P. 64, O.P. 1

RO - 4300 Tîrgu Mureș

**Aveți ceva de vîndut?  
Doriți să  
cumpărați ceva ?  
Vreți să vă oferiți  
serviciile sau aveți  
nevoie de ajutor într-o  
problemă ?**

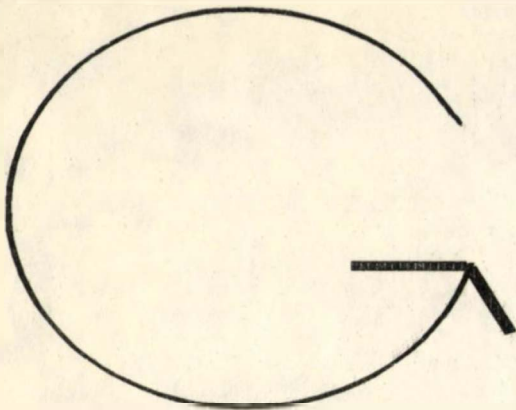
**Folosiți mica publicitate  
de specialitate din  
"if" !**

Micro ATCI

C.P. 64, O.P. 1

RO - 4300 Tîrgu Mureș





# GENESYS

SZINTEZIS  
COMPUTER GYÖR  
UNGARIA

GENESYS  
ORADEA  
ROMANIA

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

Denumire	Preț (N) (Valută)	Preț (Lei)	Denumire	Preț (N) (Valută)	Preț (Lei)
<b>Calculatoare XT</b>			GM6 mouse	\$ 21	5299.7
GF XTT,,640k,,Mo	\$ 311	78486.3	MICROSCAN	\$ 186	46940.4
GF XT,360k,640k,20M,Mo	\$ 693	174890.7	COPROCESOR 287-8	\$ 168	42397.7
4D XTTbill,,640k,,Arcn.	\$ 600	151420.5	COPROCESOR 387-33	\$ 904	228140.2
			MIDI tower+sursa	A 1747	37326.8
<b>Calculatoare AT 286</b>			<b>Discuri Winchester</b>		
GF AT-12,,1M,,Mo	\$ 366	92366.5	20 MB st225 (65ms)	\$ 233	58801.6
GF AT-12,1.2M,1M,40M,Mo	\$ 938	236720.7	40 MB st251-1 (28ms)	\$ 331	83533.6
GF AT-12,1.2M,1M,80M,Mo	\$ 1366	344734.0	80 MB st4096 (28ms)	\$ 669	168833.9
GF AT-16,,1M,,Mo	\$ 393	99180.4	1.2 GB CDC Winc.	\$ 5505	1389283.1
GF AT-16,1.2M,1M,80M,Mo	\$ 1394	351800.3	controller SCSI	\$ 138	34826.7
OFF AT-16,,1M,,Mo,Arcn.	\$ 948	239244.4			
AT-16N,,1M,,Mo	\$ 518	130726.4	<b>Memorii și accesorii</b>		
AT-16N,1.2M,1M,40M,Mo	\$ 1090	275080.6	1 MB	\$ 66	16656.3
			IC 41256-08	\$ 7	1766.6
<b>terminale DTK</b>			<b>Elemente de rețea</b>		
DTK AT-16,1M,1.2,40M,Mo	A 10440	223063.6	ARCNET 8bit STAR-BUS	\$ 61	15394.4
DTK 386SX,1M,1.2Mf,noMO			Ethernet NE1000 8bit	\$ 138	34826.7
- 16Mhz,VGA,D-Top,noKey	A 9132	195116.6	Etherlink 3COM 8bit	\$ 304	76719.7
- 20 Mhz,VGA,D-Top,noKey	A 10361	221375.7	Novell SFT 2.15	\$ 5654	1426885.8
DTK486-33M w 64K C.TOW					
- 4M,noW,FD,VGA,noMo,noKey	A 45077	963126.4			
<b>Calculatoare AT-386</b>			<b>Imprimante</b>		
AT386,mainb,33,0M,nC	\$ 842	212493.4	EPSON LX 800	D 235	35301.4
386,12/25,1.2M,1M,40M,Mo	\$ 1526	385112.8	EPSON FX 1050	D 852	127986.4
386,20/25,1.2M,2M,80M,Mo	\$ 2139	539814.1	EPSON LQ 2550	D 2760	414603.9
CH.LT3600 16/20,40W,VLCD	\$ 3036	766187.7	HP LaserJet III	\$ 2553	644294.2
LT.LA30,AT,1M,20W,MLCD	\$ 1960	494640.3	HP LaserJet II/P	\$ 1604	404797.5
			STAR NX1001	\$ 226	57035.1
			Cartridge FX1050	\$ 5	1261.8
<b>Monitoare</b>			<b>Surse neîntreruptibile</b>		
12" mono FUJIT,VALTR	U8.1		UPS 550 W	\$ 311	78486.3
14" mono SUPERTRON	\$ 110	27760.4	APC UPS 1200 W	\$ 1228	309907.3
14" EGA SUPERTRON	\$ 407	102713.6			
14" VGA (1024x768) SUPERT	\$ 414	104480.1	<b>Atele</b>		
20" VGA monitor+cartela	A 86400	1846043.9	Streamer Wangtech 60 MB IN	\$ 704	177666.7
15" MAG allo A4	\$ 869	219307.4	Modem INT 1200/SM12H	\$ 94	23722.5
			3.5" DS/DD floppy 3M	\$ 32	8075.8
<b>Adaptoare grafice</b>			3.5" DS/HD floppy 3M	\$ 40	10094.7
cartela mono	\$ 25	6309.2	5.25" DS/DD floppy 3M	\$ 9	2271.3
cartela EGA	\$ 81	20441.8	5.25" DS/HD floppy 3M	\$ 16	4037.9
VGA,8b,256k(800x600)	\$ 81	20441.8	Leporello 240x1x12x1700	\$ 16	4037.9
VGA,16b,512k(1024x768)	\$ 138	34826.7	Leporello 382x2x12x900	\$ 53	13375.5
			<b>Telefoane</b>		
<b>Subansamble</b>			PHIL. D9026 (10 mem)	\$ 54	13640
FD 1.2Mb	\$ 81	20441.8	CANON FAX 80	D 973	146162.9
FD 1.44Mb	\$ 81	20441.8	SHARP 7850	\$ 2835	715461.9
FD contr. 2 driv. AT	\$ 25	6309.2			
WD 1003 contr.	\$ 62	15646.8			
controller SCSI	\$ 138	34826.7			
Keyboard 101	\$ 46	11608.9			

KENIX

OS/2

AIX

BS/2

UNIX

386 ix

XENIX

386

BS

OS/2

Sun OS

PC-DOS

MS-

