

**CALCULATOARE
PERSONALE**

6 / 91

(9)

Revistă lunară editată de Micro ATCI S.R.L. Tîrgu Mureș

57 Lei

**Stațiile de
lucru Next**

Memento:
Specificațiile XMS

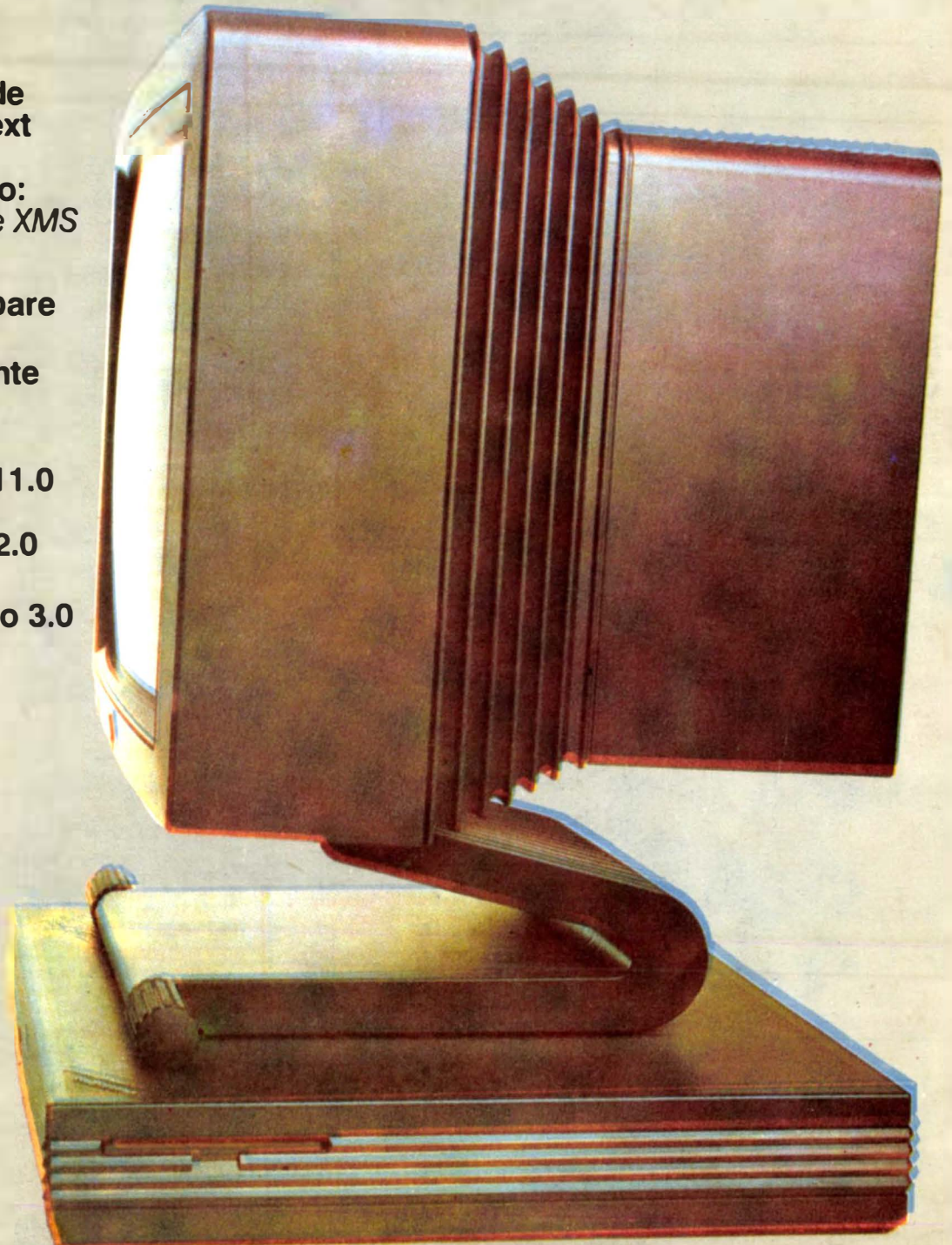
Cod de bare

**Imprimante
laser**

Autocad 11.0

Foxpro 2.0

Quattro Pro 3.0



o NOU o NOU o NOU o NOU o NOU o NOU o NOU o NOU o NOU o

actiune **EGA** la

AbMod



SISTEME COMPATIBILE IBM/PC la numai

239.000 lei

AT 286

12 MHz, 1 MB RAM, HD 40 MB, 1 unitate floppy 1.2 MB 5.25", 1 unitate floppy 1.44 MB 3.5", Monitor color EGA, 1 Mouse, 1 set dischete (10 buc) 5.25", 1 set dischete (10 buc) 3.5", 1 interfata seriala, 1 interfata paralela, tastatura 101 taste

ATENTIE !

Pentru comenzi en-gros firma acorda reduceri substantiale.

Pentru alte echipamente de **calcul** si **birotica**, solicitati lista noastra completa de preturi.

GARANTIE 1 AN DE LA LIVRARE !

ASTEPTAM COMENZI FERME LA UNA DIN ADRESELE:

3400 CLUJ-NAPOCA MAGAZIN AbMod

B-dul 22 Decembrie Nr.135 Tel: 95/156350

3400 CLUJ-NAPOCA Sediul central AbMod

Str. Mitor Nr.11 Tel, Fax: 95/111090 Telex: 31445

3700 ORADEA Str. Dacia Nr.40

Tel: 991/60278 991/44476 Fax: 991/56881



NU UITATI ! Stocul este limitat, si oferta este valabila pina la epuizarea stocului

o NOU o NOU o NOU o NOU o NOU o NOU o NOU o NOU o NOU o

if

revistă de Informatică
editată de firma Micro ATCI

Director: ing. Dumitru Dunca

Redacția:

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

Colaboratori externi:

ing. Ioan Cozac
ing. Alin Flaidăr
ing. Alin Udrea

Tiparul: tipografia Tîrgu Mureș

Tipografi:

Halațiu Mihai,
Cormoș Mircea

Revista apare lunar.

Preț: 57 lei

Manuscrise originale sau listin-
guri de programe sînt permise
cu plăcere de redacție, cu con-
diția să nu fi fost publicate și în
altă parte. Prin expedierea unui
manuscris pe adresa redacției,
autorul consimte implicit la pu-
blicarea materialului său în ca-
drul revistei. Onorariul se
negociază cu redacția. Materia-
lele nepublicate nu se îna-
poiază și nu se rețin.

Revista noastră vă oferă
spațiu pentru reclamă și publici-
tate. Pentru amănunte vă
rugăm să luați legătura cu re-
dacția.

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

Adresa și telefonul redacției:

Micro ATCI,
RO-4300 Tîrgu Mureș,
C.P. 172, O.P. 1,
tel. 954/31660 (direct),
33.612, 24.158, 20.057, 33.511,
int. 134 sau 189
41417 sau 46885 (după ora 19)
fax 954/35208,
telex 65354.

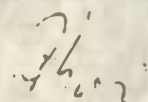
Pictures at an Exhibition

Am fost și noi la TIBCO '91. Am fost, am văzut și ne-am întors acasă. Am văzut, am întreat, am ascultat, am simțit. Căldură mare. Înghesuială. Bunuri de consum. Deocamdată mai mult pe la tîrguri. Cine știe, poate ... Nu îndrăznesc să-mi duc gîndul mai departe. Simt că devin mai ușor. Vîntul liberalizării îmi umflă buzunarele. Dar să revin ... Ce ne interesează pe noi ? Pe noi ne interesează calculatoarele personale. La TIBCO '91 am numărat cel puțin 30 de întreprinderi care se ocupau (și) cu problema în cauză.

Pozitiv este faptul că se poate cumpăra aproape orice: calculatoare, periferice, soft-uri, materiale consumabile. Bani să fie ! Dar apropo de bani, este de remarcat faptul că în timp ce prețurile la mai toate produsele cresc după curbe exponențiale, la tehnica de calcul scad. Cu toate acestea prețurile practicate sînt accesibile doar întreprinderilor. Mai așteptăm. Sper să nu-mi creez antipații afirmînd că expoziția mi-a evocat un "Ballet of the unhatched chickens". După cum ne sfătuiește institutul pentru prospectare a pieții Frost & Sullivan din Marea Britanie, (vezi articolul "O piață dificilă !"), bobocii va trebui să-i numărăm abia în toamna lui 1993. Sîntem optimiști. În loc să ne supărăm pentru faptul că ne situăm pe ultima poziție în grafic, ne bucurăm că figurăm. Albania nu apare ! Sperăm, deci, că în 1993 în România se va realiza în tehnica de calcul și informatică o cifră de afaceri de 135,63 milioane de dolari. Și mai mult decît atît !

Tot pentru că sîntem optimiști vă propunem spre lectură și articolul despre codurile de bare. Articolele despre limbajul de programare Cobol, vă vor determina, poate, să renunțați la asocierea: Cobol - preistorie, iar prezentarea celor mai noi versiuni Foxpro, Quattro și Autocad vă va da, sperăm, o imagine a amețitorului ritm în care se trăiește astăzi în lumea informaticii.

Romulus Maler





Cuprins

Știri

- PC Tools al VII-lea pag.4
- Upper Memory pentru DOS 5.0 pag.4
- Basic sub Windows pag.4
- O piață dificilă ! pag.5
- Oglinda prețurilor (mai 1991) pag.5

Hardware

- 486SX - Putere pentru toată lumea pag.6
- IBM în frunte pag.7

Doar la o zi de la anunțarea oficială a procesorului Intel 486SX, IBM a și dat o soluție pentru modelele 90 și 95. În plus a propus un viitor produs posibil cu o cartelă 486/50 MHz pentru cele două modele.

Vă prezentăm

- Next please pag.8
- Quattro Pro 3.0 pag.10

CAD

- Computer Aided Design pag.12
- Test: Auto - CAD 11.0 pag.14

Versiunea 11 a clasicului "Auto-CAD" a fost anunțată încă în urmă cu un an. Acum este disponibilă, în sfârșit, prima versiune, fapt care ne prilejuiește prezentarea unui prim test al produsului.

Aplicații

- Cod de bare pag.16

Imprimante laser

- Cum funcționează imprimantele laser? pag.24
- Sinteză: Imprimante laser la prețuri sub 5.000 DM pag.26
- Imprimanta HP Laserjet IIIP pag.27

Software

- Specificatia Lotus/Intel/Microsoft/Ast (XMS) pentru memoria extinsa supliment
- Cobol = Cobol ? pag.33
- MS - Cobol V 4.0 PDS pag.36

Cel ce-și aduce aminte de primul compilator Cobol pentru PC va rămîne perplex în fața celei mai noi versiuni. Editoarele confortabile și multele programe utile ușurează foarte mult munca programatorului profesional. În sfârșit te poți concentra asupra problemei principale - programarea. Multe munci suplimentare care înainte cădeau în sarcina programatorului sînt preluate acum de compilator.

- Foxpro 2.0 pag.38

În cursa bazelor de date, Foxpro 2.0 a luat din nou un avans în fața concurenților dBase IV și Paradox 3.5.

Cursuri

- Rețele neuronale partea a IV-a pag.40

În ultimii ani, recunoașterea scrisului prin calculator s-a perfecționat mereu. Recunoașterea de caractere prin intermediul rețelelor neuronale este un pas într-o direcție promițătoare.

- OOP pag.44

Practică

- Protecție prin parolă pag.50
- Vă place muzica ? pag.52
- Ștergere rapidă pag.52
- ChDir confortabil pag.53

Rubrici

- Caseta redacției pag. 1
- Editorial pag. 1
- Poșta redacției pag.54
- Mica publicitate pag.54

PC Tools al VII-lea

Versiunea 7.0 a lui PC Tools a început o nouă rundă în confrun-tarea pentru titlul de cel mai bun pro-gram utilitar. Noile atu-uri: suport rețea și suport Windows.

Săptămânile trecute, Central Point Software, din SUA, a anunțat a șaptea versiune a programului utilitar PC Tools. Noua versiune conține, după declarațiile lui Corey Smith, președintele CPS, 80 de deta-lii îmbunătățite și 10 module noi, care, ca de obicei, pot fi apelate de sub o suprafață utilizator unitară. PC Tools devine primul program utilitar care poate fi rulat atât sub MS-DOS cât și sub Windows și a cărui mod de operare este identic în ambele medii.

Printre cele mai importante noutăți se numără un program de transfer de date, asemănător cu Laplink, cu ajutorul căruia, prin in-

termediul unui modem, a unei rețele, sau a unui cablu, poate fi comandat un calculator de la dis-tanță. Transferul de date între cele două calculatoare este posibil cu ajutorul programului Commute. Posibilitatea lucrului în rețea joacă un rol important atât în cazul lui Commute cât și în cazul celorlalte module. Astfel "Undelete" poate re-cupera acum și fișiere aflate pe un server Novell, iar Filefind utilizează atributele extinse de rețea. Printre celelalte noutăți se numără un pro-gram antivirus și alte utilitare cu-noscute deja din programele concurenței, cum ar fi "Wipe" care șterge un fișier astfel încât să nu mai poată fi recuperat, sau "FileFix" care poate recupera fișiere dBase sau Lotus distruse.

PC Tools costă 179\$ în SUA și trebuia să fie livrat începând din lu-na mai.

Basic sub Windows

Visual Basic, produs de Micro-soft, permite elaborarea unor apli-cații orientate obiect sub suprafața utilizator grafică Win-dows.

La un an de la prezentarea lui Windows 3.0, Microsoft prezintă primul limbaj de programare orientat obiect care să lucreze sub această suprafață utilizator: Visual Basic. Acest software a fost prezentat la târgul de primăvară Comdex din Atlanta, USA.

Analiztii pieții din SUA esti-mează că această prezentare este o mișcare strategică importantă pentru Microsoft. Da-vid Buyer, analist la Montgomery Securities: "Visual Basic va deter-mina o și mai mare răspîndire a lui Windows 3.0 în întreprinderi și va duce la o creștere a cifrei de afa-ceri Microsoft. Dezvoltarea de

aplicații Windows este simplifi-cată semnificativ. Visual Basic va deveni fundamentul multor pro-grame Windows". Visual Basic constă, în mare, din două compo-nente centrale: "Forms", pentru cuplarea suprafeței grafice cu cu-noscutele obiecte Windows, și "Program Units" care conțin pro-cedurile limbajului.

Rick Sherlund, analist la Gold-man Sachs, comentează: "Mo-mentan, cu doar 90 milioane dolari, limbajele de programare reprezintă doar o mică parte din cifra de afaceri Microsoft. După lansarea pe piață a lui Visual Ba-sic se așteaptă rate de creștere de 30 până la 50% la limbajele de programare.

În SUA, Visual Basic va costa cca. 250\$.

Plăci de extensie

Upper Memory pentru DOS 5.0

Întreprinderea Dawicontrol din Göttingen produce o placă de extensie de memorie pen-tru MS-DOS 5.0, "Upper Me-mory Board", care poate fi montată în toate PC-urile, de la XT și pînă la 486. Cea mai nouă versiune a sistemului de ope-rare MS-DOS recunoaște, în gestionarea memoriei, așa-numitele Upper Memory Blocks (UPB). Cu ajutorul lor driver-ele și programele rezide-nente pot fi încărcate în memo-ria superioară (Upper Memory). În acest mod utiliza-torului îi rămîn disponibili pînă la 600 KByte din memoria de bază.

Cartela Upper Memory pu-ne la dispoziția MS-DOS 5.0 spațiul fizic RAM necesar. În plus, spațiul disponibil în ex-tensia de memorie poate fi uti-lizat, la nevoie, pentru extinderea memoriei de bază. Memoria disponibilă pe placa de extensie, poate fi împărțită în ferestre de cîte 16 KByte care pot fi inserate, la dorință, în interiorul spațiului de adresare cuprins între 512 KByte și 1 MByte.

Programul de instalare sta-bilește domeniile RAM libere din sistem și preia, orientat dia-log, întreaga configurare. Re-glarea prin comutatoare DIP sau jumperi a fost înlăturată în întregime. În configurația de bază, placa este echipată cu 256 KByte și costă 295 DM. La cerere ea poate fi echipată cu pînă la 512 KByte RAM.

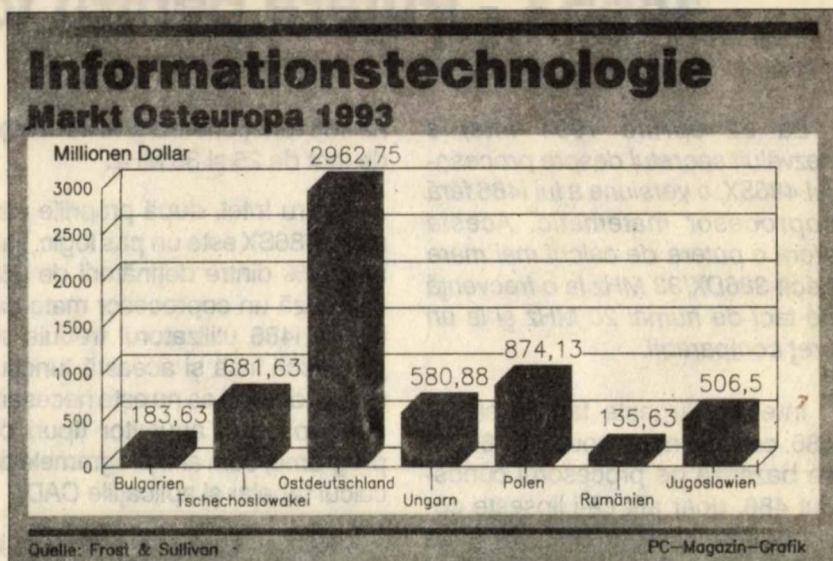
Informatii: Dawicontrol, Leinestr. 36, 3400 Göttingen, Tel. 0551/72075.

O piață dificilă !

Producătorii de calculatoare care doresc să fie prezenți în Europa de Est trebuie să aibă nervii tari. Ei trebuie să investească încă de pe acum, dar nu se pot aștepta la câștiguri decât după 1993, afirmă institutul de prospectare a pieței Frost & Sullivan într-un studiu.

Febra aurului și speranța într-un câștig rapid, după introducerea economiei de piață în țările din estul Europei, sînt deplasate, este de părere institutul de prospectare a pieței Frost & Sullivan, în urma unui studiu despre posibilitățile piețelor esteuropene. "Această piață nu este pentru cei cu nervii slabi", se afirmă în studiu, deoarece șanse au doar aceia care investesc acum pentru a-și asigura imediat prezența pe piață. Câștigul rezultat din aceste investiții s-ar putea să apară, însă, abia după anul 1993.

Frost & Sullivan a stabilit că pentru anul trecut cifra de afaceri realizată de industria occidentală de tehnică de calcul și informatică în țările din estul Europei, incluzînd fostul RDG, se ridică la 1,68 miliarde de dolari. Această cifră ar putea să crească în anii următori la 4,86 miliarde de dolari. Luînd în calcul și o posibilă rată de eroare se pronostichează că în varianta pesimistă această cifră ar putea urca



doar pînă la 3,8 miliarde de dolari, în timp ce în varianta optimistă această cifră de afaceri ar putea atinge în 1993 6 miliarde de dolari (vezi graficul alăturat).

Cea mai mare piață esteuropeană se va situa în următorii ani pe teritoriul fostului RDG. Aceasta se explică și prin faptul că în fostul CAER RDG-ul dispunea de cea mai puternică economie socialistă. Frost & Sullivan: "În fostul RDG, motorul principal al dezvoltării industriale în fostul CAER, se vor desface aproape jumătate din livrările următorilor ani".

Pentru întreprinderile care nu vin din vest, dezvoltarea pieței se va face mai degrabă lent: "La început cea mai mare cerere va exista pentru calculatoare personale. Ulterior se vor vinde destul de bine și echipamente Unix și calculatoare de marcă pentru întreprinderi de dimensiune medie. Numai în landurile fostului RDG va exista încă de la început și o piață semnificativă pentru calculatoare mari."

Ritmul lent, previzibil, de dezvoltare a țărilor din estul Europei nu trebuie să-i determine pe ofertanți să tragă concluzia că pot aștepta încă, afirmă Frost & Sullivan. Fundamentul pentru un succes în 1993 trebuie pus încă de pe acum. "Investițiile necesare sînt semnificative", atrag atenția cercetătorii londonezi, dar "importantă este o pătrundere rapidă pe piață, adesea prin înființarea de firme mixte cu întreprinderile locale." Mai multe întreprinderi, cu o mare putere financiară, au pășit deja pe acest drum, printre altele IBM, Siemens și DEC. În ciuda timpilor mari de așteptare, aceste întreprinderi au recunoscut: "Aceia care vor pătrunde cel mai rapid pe aceste piețe vor avea în viitor cele mai bune perspective".

(R.M.)

Oglinda prețurilor (mai 1991)

Sisteme	Preț de stradă (no-name)	Aparate de marcă ca de ex.
AT 80286-CPU, 1 MByte RAM harddisk de 40 MByte	1.248 DM	Compaq 286 N1/SL 3.780 DM
386SX 80386SX-CPU, 1 MByte RAM, harddisk de 40 MByte, o unitate floppy, monitor VGA	2.490 DM	Compaq 386 N1/SL 4.415 DM
386/33 80386 CPU, 2 MByte RAM, harddisk de 180 MByte, o unitate floppy, monitor VGA	4.408 DM	IBM-PS/2-80 A-31 12.350 DM
486/33 80486-CPU, 4 MByte RAM, harddisk de 330 MByte, o unitate floppy, monitor VGA	9.450 DM	Compaq 486 23.695 DM

486SX - Putere pentru toată lumea

La 22 aprilie 1991 Intel a dezvăluit secretul despre procesorul 486SX, o versiune a lui i486 fără coprocesor matematic. Acesta oferă o putere de calcul mai mare decât 386DX/33 MHz la o frecvență de tact de numai 20 MHz și la un preț comparabil.

Intel își lărgeste familia sa de 486: cea mai mare noutate, 486SX, se bazează pe procesorul cunoscut 486, doar atât că-i lipsește coprocesorul matematic. Ca și în cazul procesoarelor 386 și 486, și în acest caz este vorba despre un procesor pur de 32 de biți, compatibil integral soft cu predecesorii săi din familia Intel x86.

Performanțele de calcul ale noului venit, date de Intel sînt de 16.5 MIPS (milioane de instrucțiuni pe secundă), la o frecvență de tact maximă de 20 MHz. Comparativ 386DX ajunge la 11.3 MIPS, la o frecvență de tact de 33 MHz. În ciuda saltului calitativ clar, procesorul 486SX va costa doar cu 20 pînă la 30 DM mai mult decât un 386DX. David House, de la Intel, comentează intrarea pe piață astfel: "Procesorul Intel 486SX deschide utilizatorului drumul spre calculatoarele care-și merită banii, posedînd performanțele superioare ale tehnologiei 486".

Cu 486SX, specialistul în procesare copiază strategia de succes a procesorului 386SX. După produsul inovator 486 apare deci o variantă mai redusă, dar mai accesibilă ca preț.

Pentru 486SX, Intel oferă și coprocesorul matematic 487SX, care face calculatoarele SX comparabile cu versiunea completă 486. Combinația 486SX - 487SX costă totuși mai mult decât un 486. O altă limitare este faptul că 486SX se produce deocamdată numai în varianta 20 MHz, în timp ce proceso-

rul 486 funcționează și la frecvențe de tact de 25 și 33 MHz.

Pentru Intel, după propriile afirmații, 486SX este un pas logic. Numai 30% dintre deținătorii de 386 utilizează un coprocesor matematic. La i486 utilizatorul trebuie să plătească însă și această funcțiune, cutoate că ea nu este necesară decât în cazul anumitor tipuri de programe, cum ar fi programele de calcul tabelar și aplicațiile CAD.

486SX scoate mai pregnant în evidență diferențele între arhitecturile lui 386 și 486, deoarece i486 este mai mult decât "doar" un procesor 386 cu cache și coprocesor. Ultimele modele ale familiei x86 au fost îmbunătățite ca design față de 386, ceea ce explică și avantajul vitezei sporite în cazul lui 486SX.

După datele furnizate de Intel, arhitectura 486, la același tact, permite o putere de două pînă la trei ori mai mare decât cea a procesoarelor 386. Cu toate că familia x86 aparține familiei procesoarelor CISC (Complete Instruction Set Computer), 486 și 486SX prelucrează multe comenzi ca un procesor RISC (Reduced Instruction Set Computer) în cadrul unui ciclu de tact.

Un avantaj hotărîtor vis à vis de arhitectura 386 este timingul modificat al procesorului: i486 utilizează fiecare impuls al cuarțului, în timp ce 386 necesită cuarțuri de două ori mai rapide pentru a putea atinge aceeași rată de tact efectivă. În toate procesoarele 386 la 33 MHz bate deci inima unui cuarț de 66 MHz, ceea ce implică măsuri speciale de ecranare. Din această cauză crearea unor procesoare 386 cu o frecvență de tact mai ridicată devine o problemă, deoarece un cuarț de 80 sau de 100 MHz devine tot mai greu de ecranat. Arhitectura 486 folosește mai economic ciclurile de tact și din

acest motiv poate lucra mai ușor la un tact de 50 MHz sau chiar de 100 MHz. Procesorul 486 la 50 MHz nu a fost încă anunțat oficial, dar diverși producători de calculatoare, printre care și IBM, vorbesc deja despre el, și își planifică să-l folosească cît mai rapid posibil.

Tactul de 20 MHz mai aduce un avantaj lui 486SX: producătorii de PC-uri pot adăuga componente periferice convenabile, deoarece o placă de bază 486SX, per total, ar trebui să fie mai ieftină decât o placă de bază 386DX. Viteza procesorului 486SX este influențată mai ales de cache-ul integrat, care lipsește la 386.

În ciuda tuturor avantajelor, 486SX le va spori clienților chinul alegerii: să te decizi pentru un 386, sau mai bine să aștepți un 486SX ? Merită să-ți cumperi un 486 sau pentru anii următori ajunge și un 486SX ? Răspunsurile la aceste întrebări vor fi date în lunile următoare de către producătorii de PC-uri. De prețurile pe care le vor practica va depinde cît de repede se va impune 486SX.

Specialiștii apreciază că vânzările de 386 vor scădea sensibil, dacă producătorii vor transfera avantajul prețului mai scăzut și asupra produselor finale. Dacă însă vor pretinde un preț suplimentar pentru puterea sporită, atunci 386 mai are încă șanse bune de a se menține pe piață. Cît de rapid va reuși deci 486SX să-l înlocuiască pe 386 pe piață rămîne de văzut.

Politica de prețuri agresivă, practică de Intel, lasă să se întrevadă faptul că în Santa Clara schimbarea este dorită mai degrabă astăzi decât mîine. Motivul ar putea fi "copierea" lui 386 de către AMD, căruia 486SX i-ar putea lua piuitul.

(I. M. & R. M.)

IBM este în frunte

Doar la o zi de la anunțarea oficială a procesorului Intel 486SX, IBM a și dat o soluție pentru modelele 90 și 95. În plus a propus un viitor produs posibil cu o cartelă 486/50 MHz pentru cele două modele.

IBM vrea să fie din nou în fruntea dezvoltării tehnice, acest lucru l-a demonstrat Big Blue (Marele Albastru) la 23 aprilie în München. Doar la o zi după ce Intel și-a prezentat noul procesor 486SX IBM și-a prezentat modelele PS/2 90 și 95 dotate cu acest procesor.

Sistemele introduse în octombrie anul trecut folosesc sisteme complexe de procesoare interschimbabile, astfel încât clientul să-și poată completa mai târziu calculatorul cu un 486 de 25 sau de 33 MHz. "Vrem să protejăm investiția clienților noștri" afirmă Dr. Wolfram Ischebeck directorul comercial de la IBM. Clientul nostru va trebui doar să înlocuiască modelul CPU în loc să-și cumpere alt calculator de capacitate mai mare.

IBM este destul de sigur pe sine când afirmă că aceste modele PS/2 nu vor îmbătrâni în următorii 5-7 ani, datorită posibilității de actualizare (upgrade) pe care le au, cu toate că pentru 1992 se așteaptă de la Intel procesoare 586.

Cele 2 variante noi reprezintă configurațiile inițiale ale modelelor PS/2 90 și 95, pe care IBM le-a oferit pînă în prezent doar cu ver-

Vedere de ansamblu	
Calculator	PS/2, Model 95 XP
Producător	IBM
Preț	încă necunoscut
Procesor	486SX
Frecvența de tact	20 MHz
Memoria principală	8 MByte
Adaptor grafic	XGA
Harddisk	160 MByte
Controller	SCSI
Sistem magistrală	Microcanal
Cuploare (slot-uri) libere	6

siunea completă a procesorului 486.

Modelul 90 este un calculator desktop, avînd 3 cuploare de microcanal libere. El este dotat cu o memorie de 4 MByte (care poate fi completată pînă la 32 MByte) și o placă grafică modernă XGA, care oferă o rezoluție de 1034x768 pixeli la 256 de culori, sau de 640x400 pixeli la 300.000 culori (16 biți).

Modelul 95 se aseamănă în mare măsură cu modelul 90 în ceea ce privește datele tehnice, doar că dispune de o carcasă tower mult mai mare și mai robustă. Calculatorul are aproape 22 kg, dispune de pînă la 8 cuploare de microcanal (două din cele 8 sînt deja ocupate de placa grafică și de controller-ul de harddisk) și dispu-

ne de spațiu pentru inserarea a încă 7 plăci, de exemplu alte 4 harddiskuri SCSI, care pot mări capacitatea de memorare pînă la peste 2 GByte. Tower-ul se pretează din acest motiv ca server de capacitate mare. Tocmai aici oferta de actualizare (upgrade) a lui IBM devine interesantă, deoarece capacitatea serverului poate fi mărită doar prin cîteva operațiuni, fără ca să fie necesară schimbarea calculatorului.

În legătură cu strategia de dezvoltare a IBM-ului, s-a prezentat o placă 486 de 50 MHz pentru modelele 90 și 95. Peter Richard, susține că ar fi o demonstrație de tehnologie și nu de un produs terminat. Placa de 50 MHz ar putea fi în scurt timp o extensie a modelelor 90 și 95.

Conform afirmațiilor IBM primele aparate cu procesorul 486SX vor fi livrate în iunie 1991. Prețul nu este încă stabilit.

(I.M.)

Familia x86 dintr-o privire

Date comparative

Procesor	I386	I486SX	I486
Anul apariției	1986	1991	1989
Frecvența de tact maximă	33	20	33 (50)
Puterea (MIPS)	11,4	16,5	27

Next please

Acum există calculatoare Ferrari doar la jumătate de preț: Next, noua firmă a vizionarului care a creat calculatorul Macintosh, Steve Jobs, vrea să cucerească piața mondială cu stații de lucru (Workstation) de 11.000 DM.

Ele sînt domnițoarele rețelilor, maestrele simulărilor complexe, stăpînele graficii color - cu stațiile de lucru începe lumea de dincolo de PC-uri.

Din exterior stațiile de lucru arată ca niște calculatoare uzuale, inofensive, de birou, dar afară-i vopsit gardul și înăuntru Microprocesoare speciale se întrec în a-i asigura performanțe maxime, ceea ce este bine pentru lucrul în rețea, este bine pentru baze de date mari, bine pentru aplicații grafice și color, bine pentru simulări, dar rău pentru pungă: stațiile de lucru uzuale ale firmelor Hewlett-Packard, Sun sau IBM costă toate peste 20.000 DM.

Această limită a prețurilor se va clătina în curînd. Steve Jobs, excentricul vizionar al lui Apple, de la care a fost nevoit ulterior să plece, intră din nou în arenă. Noua sa firmă Next Computer a adus pe piață trei noi stații de lucru la prețuri tentante: "Nextstation" (cca. 10.800 DM), "Nextstation Color" (cca. 17.300 DM) și "Nextcube" (cca. 17.300 DM).

Pentru utilizatorii obișnuși aceasta este ca și cum le-ai oferi o mașină Ferrari la jumătate de preț:

senzațional de ieftin, dar cu toate acestea de neplătit.

Toate cele trei variante sînt echipate cu microprocesoare Motorola 68040 care lucrează cu o frecvență de tact de 25 MHz, ceea ce le permite atingerea unei puteri de calcul de 15 MIPS (million instructions per second - milioane de instrucțiuni pe secundă). Pentru comparație să amintim că "Macintosh IICI" nu atinge decît 4 MIPS iar calculatoarele MS-DOS echipate cu 386 nu depășesc nici ele prea mult această valoare.

Memoria principală a calculatorului "Next Station" este de 8 MByte și poate fi extinsă pînă la 32 MByte.

Fiecare Next dispune de un lector de dischete de 3,5" care lucrează cu "Extended Density" pe care se pot stoca de două ori mai multe date decît pe dischetele "High Density" ale calculatoarelor compatibile IBM; dischetele au o capacitate de 2,44 MByte și costă 8 DM/bucată. Lectorul recunoaște și dischetele formatate și înregistrate pe un calculator compatibil IBM, pe care le poate atîta citi cît și scrie fără să necesite în plus nici un program special (așa cum este cazul calculatoarelor Macintosh).

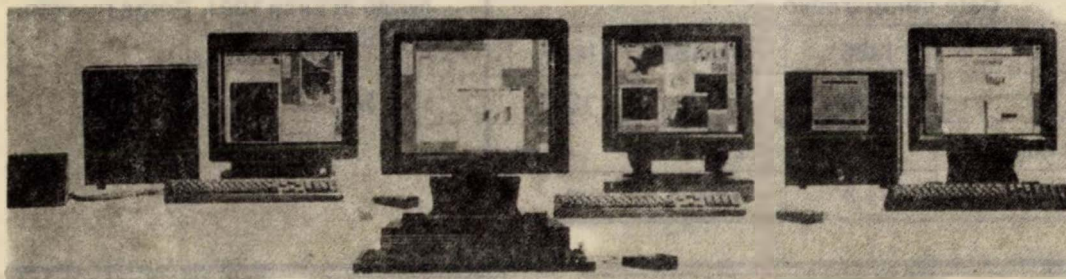
Versiunile de bază "Next Station" și "Next Station Color" sînt livrate cu harddisk-uri avînd capacitatea de 105 MByte (timp de acces 17 ms, rată de transfer 4 MByte/sec). Optional poate fi livrat și un hard-

disk de 340 Mbyte (timp de acces 15 ms, rată de transfer 8 MByte/sec). "Nextcube" poate fi echipat chiar și cu un harddisk de 1,44 GByte (timp de acces 13 ms).

Puterea de calcul și memoria ridicată recomandă utilizarea lui "Next Station" în special pentru aplicații grafice, CAD și DTP: pe monitorul "Megapixel" (diagonală 17"=43 cm) poate fi vizualizată o pagină de format DIN A4 întregă - doar monocrom - la o rezoluție de 832x1120 pixeli, fiecare punct putînd fi reprezentat în 4 nuanțe de gri. "Next Station Color" însă la aceeași rezoluție poate reprezenta 4096 culori.

Dacă totuși nu este suficient, atunci cu "Nextcube" se pot obține imagini și mai împestrîtate: pentru aceasta este necesară însă placa de extensie "Next Dimension" (cca. 8.700 DM) care permite reprezentarea a 16,7 milioane de culori (magistrala este de 32 biți, 24 de biți permit generarea a 16,7 milioane de culori, iar cei 8 biți rămași permit reprezentarea a 256 de nivele de transparență), rezoluția rămînd aceeași.

Dar Next nu încîntă doar văzul ci și auzul cu cipul Motorola special "56001 DSP" (DSP = Digital Signal Processor), utilizatorul poate înregistra, prelucra și reda vorbirea, muzică și zgomote; microfoanele și difuzoarele stereo sînt înglobate în monitor. În acest fel utilizatorul



Familia Next.

poate înzestra documentele și fișierele și cu comentarii verbale.

Sisemul de operare utilizat de Next este Unix, sistem de operare pe care îl folosesc de altfel și majoritatea celorlalte tipuri de stații de lucru. Unix este recomandabil în special pentru lucrul în rețea și pentru a permite rularea simultană a mai multor programe pe același calculator (multitasking). Dezavantajul: Unix este destul de incomod în operare. Pentru a contracta acest minus Next utilizează suprafața utilizator grafică "Nextstep" asemănătoare suprafeței utilizator "Finder" a calculatoarelor Macintosh. Utilizatorul nu mai trebuie să tasteze comenzi complicate (cum trebuie s-o facă de ex. în cazul sistemului de operare MS-DOS) ci poate opera manipulând simboluri cu ajutorul mouse-ului.

Next produce și o imprimantă laser (8 pag./min.), care permite o rezoluție selectabilă de 300 sau 400 dpi (dot per inch). Cu toate că imprimanta laser nu este o imprimantă Postscript (așa se explică și prețul de doar 3.900 DM), datorită

faptului că sistemele dispun de soft-ul "Display Postscript" incorporat, imaginea de pe monitor va fi reprodusă în mod identic și pe hîrtie (WYSIWYG = what you see is what you get).

Dar nici cel mai bun calculator și nici cea mai bună suprafață utilizator nu-i sînt de nici un folos utilizatorului dacă nu există programare care să poată fi utilizate. Aceasta a fost una din cele mai serioase critici aduse primului calculator Next (vechiul "Nexcube"). Oamenii de la Next au încercat să elimine acest reproș, pe harddisk-ul fiecărui Next livrîndu-se în mod standard un număr de pachete soft.

În magazine există deja pachete soft produse și de alte firme. În toate domeniile există deja programe interesante: procesoare de texte, grafică, Desktop Publishing, animație, matematică și statistică, calcul tabelar, baze de date, programare.

Angajamentul unor firme de renume oferă garanția faptului că nici în viitor nu va lipsi softul pentru

Model: Next-Station
Producător: Next Computers
Preț: începînd de la 8200DM, (monitor monocrom 2100 DM, set accesorii 500 DM)
Procesor: Motorola 68040 (25 MHz)
Sistem de operare: Unix, Nextstep 2.0
Monitor: "Megapixel", rezoluție 1120x832, 4 nivele de gri
Memoria principală: 8 MByte (max. 32 MByte)
Memoria de masă: 105 MByte (pînă la 400 MByte), lector de dischete de 3,5" (2.88 MByte pe dischete Extended Density)
Dimensiuni: 6,4x36,5x39,8 cm fără monitor. Greutatea 5,5 kg.

Next: Wordperfect, Lotus, Ahton Tate și Oracle se ocupă de dezvoltarea de programe pentru Next.

Cocluzionînd "Next Station" este un calculator foarte puternic, iar prețul de 10.800 DM (complet cu monitor) este pe măsură.

(R.M.)

Software pentru Next

Next livrează fiecare calculator cu o selecție de programe cu care utilizatorul se poate apuca imediat de lucru. Cu "Next Stations" cu harddisk de 100 MB se livrează printre altele:

- **Nextmail**, cu ajutorul căruia utilizatorul poate expedia documente în rețea. Cu ajutorul unui modem-telefax acesta își poate utiliza Next-ul și ca telefax.
- **WriteNow**, un procesor de texte (fără funcție de scriere de scrisori în serie).
- **Draw**, un program de desenare.
- **Digital Webster**, un dicționar englez, și Webster's Thesaurus.
- **Date**, o agendă de lucru.

Cu toate calculatoarele avînd harddisk-uri mai mari, Next livrează și alte soft-uri, printre altele și un compilator C.

Vă prezentăm în continuare o selecție de soft-uri pentru Next scrise de alte case de soft:

- **Wordperfect**, cunoscutul procesor de texte (Preț: 500\$).
- **Topdraw**, un superprogram de desenare al firmei Media Logic (Preț: 595\$).
- **Lotus Improv 1.0**, program de calcul tabelar, urmașul lui Lotus 1-2-3, are toate șansele să impună un nou standard pentru programele din categoria sa (Preț: 695\$).
- **Soft PC**, emulator soft. Cu ajutorul lui pot fi rulate pe Next toate programele scrise pentru MS-DOS. (Preț încă necunoscut).
- **Oracle**, sistem de gestiune a bazelor de date (Preț încă necunoscut).
- **Framemaker**, un program Desktop-Publishing pentru Next (Preț: 995\$).

A calcula și a prezenta

Quattro Pro 3.0

Borland încearcă să bată toate recordurile la prezentarea unor noi versiuni de programe. De-abia a fost lansat Quattro Pro 2.0 pe piață și deja a fost prezentată noua versiune Quattro Pro 3.0.

A devenit deja un obicei ca la puține săptămîni de la lansarea unui produs, să se prezinte o nouă variantă, în care să fie corectate "scamele" apărute și care uneori era ornată cu câteva noi extensii. Cu toate acestea ritmul, în care lucrează proiectanții lui Quattro Pro, îți taie respirația: în decurs de un an și jumătate au fost produse două versiuni ale programului de calcul tabelar Quattro Pro. Ca și în cazul versiunii anterioare, nici în cazul versiunii 3.0 nu este vorba doar despre modificări cosmetice ci de un program cu funcțiuni noi, impresionante, care-i v-a determina desigur pe utilizatori să opteze pentru noua versiune.

Pe lângă suprafața orientată caracter este disponibil acum și un mod de lucru WYSIWYG care permite virtualizarea tuturor fonturilor, a atributelor de scriere, a liniilor și a umbririlor, așa cum vor arăta după imprimarea lor pe hîrtie. Dar aceasta nu este totul, ecranul cu care produsul se prezintă în modul grafic a fost prelucrat în întregime. Etichetele inscripționărilor liniilor și coloanelor, butoanele meniurilor și cutiile de dialog sînt prezentate acum în trei dimensiuni.

Asemănător ca la Lotus 1-2-3, în modul grafic se poate face o scalare (zoom) cu un număr nelimitat de trepte între 25% și 200%.

Deoarece vă prezentăm una din primele versiuni beta ale produsului, nu vă putem da încă date exacte referitoare la viteza de calcul. Ceea ce vă putem spune deja este

că în modul WYSIWYG Quattro Pro atinge o viteză acceptabilă. Defilarea pe ecran și funcțiile la nivel de bloc au devenit mai lente, din cauza modului grafic, dar aceste dezavantaje se păstrează în limite acceptabile.

Și la imprimarea unei tabele pe hîrtie sînt oferite acum posibilități de a scăpa de multe din necazurile obișnuite. Quattro Pro 3.0 oferă mai multe posibilități de a imprima pe hîrtie tabele complexe: dacă de exemplu tabela conține mai multe linii și coloane decît ar încăpea pe hîrtie, se poate apela funcția "Print to fit" și programul va imprima tabela la o astfel de scară încît aceasta să nu depășească marginile hîrtiei. La cerere, raportul de micșorare poate fi stabilit și manual. O tabelă poate fi și mărită de pînă la 10 ori pentru a fi imprimată.

Această metodă ajută puțin, totuși, atunci cînd este vorba despre tabele cu peste 100 de coloane. În acest caz poate fi folosită o nouă tehnologie integrată în Quattro Pro: asemănător ca în cazul auxiliarului (add-in) Lotus "Sideways", o tabelă poate fi listată ca "Banner". În acest caz ea este imprimată orizontal, fără întreruperi de pagină, astfel încît pe hîrtie fără sfîrșit (perforată) se poate obține o copie 1:1 a tablei.

Și în editorul grafic au fost incluse noutăți. Ca și în cazul programelor de grafică profesională desenul poate fi construit avînd la bază un rastru, care poate fi folosit pentru poziționarea și pentru dimensionarea obiectelor. Simultan poate fi activată și funcția "snap-to" caz în care Quattro Pro va începe unui obiect grafic întotdeauna exact într-un punct al rastrului și îl va sfîrși într-un alt punct al acestuia.

Ca și pînă acum cîmpurile de text pot fi definite ca butoane și la

selectarea lor pot apela o altă diagramă sau un macrou. Nou este așa-numitul buton "background" care poate fi de asemenea cuplat cu un alt grafic. Astfel este suficient un clic pe mouse într-un anumit loc de pe o diagramă pentru a fi afișată următoarea imagine.

Modul de prezentare al unei expuneri (slide-show) poate fi îmbunătățit prin folosirea efectelor speciale din versiunea 3.0. La schimbarea unei imagini pot fi folosite efecte ca "defilare" sau "spargere" a imaginii pentru a capta atenția spectatorului. Extrem de utilă este și funcția "autosave" în cazul graficelor: cu ajutorul ei putem fi siguri că orice modificare efectuată într-o diagramă va fi salvată pe harddisk.

Pe lângă noutățile enumerate, Quattro Pro 3.0 dispune și de câteva mici funcții care în versiunea beta nu erau încă definitive. În versiunea finală va fi posibilă atît citirea cît și scrierea de fișiere CGM. În acest caz graficele vor putea fi preluate în toate programele de prezentare uzuale.

Meniul "File" a fost extins cu comanda "Save All", cu ajutorul căreia pot fi închise, cu o singură operație, toate fișierele deschise la un moment dat. Cîteva comenzi noi pentru construirea de macrouri permit însoțirea prezentărilor cu mici bucăți muzicale.

Quattro Pro dovedește cu versiunea 3.0 că este un concurent serios în competiția pe care o desfășoară împreună cu Lotus 1-2-3 și Excel, pentru titlul de cel mai bun program de calcul tabelar. Cerințele hard (AT, 512 KByte RAM, harddisk) nu vor crește nici în noua versiune.

(R.M.)

Tabel comparativ

	Quattro Pro 3.0	Excel 3.0	Wingz
Preț	495 DM	1847 DM	1704 DM
Date tehnice			
Nr. linii X Nr. coloane X Nr. foi de lucru	8192 x 256 x 0	16384 x 256 x 0	9999 x 256 x 0
Fișiere deschise simultan	32	funcție de mem.	funcție de mem.
Foi de calcul tridimensionale	nu	nu	nu
Lungime câmp	255 caractere	255 caractere	255 caractere
Coprocessor	da	da	da
Spațiu ocupat pe harddisk	4,5 MByte	5 MByte	3 MByte
Caracteristici de putere: Calcul			
Funcții autodefinită	nu	da	da
Căutare pt. obținerea unei valori finale impuse	da	da	nu
Solver	da	da	nu
Tabele ce-se-întâmplă-dacă cu X variabile	2	2	2
Consolidare inteligentă	nu	da	nu
Cuplare cu fișiere de pe harddisk	da	da	nu
Caracteristici de putere: Publicistică			
Combinare de tabele și diagrame	da	da	da
Suport Postscript	da	da	da
Import formate grafice	CLP, CGM	clipboard	clipboard
Funcții de desenare în tabelă	nu	da	da
WYSIWYG	da	da	da
Previzualizare	da	da	da
Imprimare la dimensiunea hîrtiei	da	da	nu
Caracteristici de putere: Baza de date			
Mască pentru baza de date	nu	da	nu
Chei de sortare	5	3	255
Interogare bază de date externă	da	da (cu Q+E)	nu
Actualizare bază de date externă	nu	da (cu Q+E)	nu
Compatibilitate cu Lotus 1-2-3			
Structura meniurilor	da	Help special	nu
Taste	da	nu	nu
Macroui	da	cu conversie	nu
Import / Export			
WKS	da	da	da
WK1	da	da	da
WK3	nu	da	nu
DBF	da	da	da
SYLK	da	da	da
DIF	da	da	da
BIFF	nu	da	da

Computer Aided Design

Zilele planșetei de desen sînt numărate. Pentru efectuarea de desene tehnice PC-urile cu soft corespunzător au nevoie de mai puțin loc, risipesc mai puțină hîrtie și oferă utilizatorului mai multă putere și confort decît creionul.

Prescurtarea CAD este folosită pentru Computer Aided Drafting, și respectiv pentru Computer Aided Design. A doua variantă a acestei noțiuni este mai cuprinzătoare și mai uzuală. În timp ce Drafting se referă la desenarea cu ajutorul calculatorului, Design depășește cu mult cadrul lucrărilor grafice efectuate cu ajutorul calculatorului. Din acest motiv ar fi greșit să traducem această noțiune doar prin construcție: ea cuprinde mult mai mult și anume întregul spectru al activităților care trebuiesc depuse pentru crearea și întreținerea unui produs.

Sistemele CAD de azi au, de exemplu, ca sarcini tipice pe lîngă desenare și construire, și simularea unor încărcări, mișcări și caracteristici de material. Programele trebuie să pună la dispoziție și unelte pentru tratarea desenelor și a listelor componentelor din structură.

Un element esențial care trebuie să ghideze decizia achiziției unui sistem CAD este și compatibilitatea formatului datelor cu a altor aplicații. De aceasta depinde posibilitatea ca utilizatorii să poată prelua sau preda desenele lor din/spre alte sisteme de calcul.

De cînd calculatoarele personale au devenit atît de puternice și de accesibile ca preț încît pe ele se pot rezolva sarcini complexe cu un efort acceptabil, CAD s-a impus în multe domenii. Cele mai importante domenii de utilizare sînt: construcții de mașini, arhitectură, electrotehnică și mai nou din ce în

ce mai mult arhitectura de interior. Producerea de softuri pentru crearea de construcții 3D și umbriri și posibilitățile care decurg din acestea, de a crea și calcula imagini spațiale expresive de produse, clădiri sau peisaje, au propulsat CAD-ul în vîrfurile aplicațiilor profesionale de grafică.

Diferența dintre un program CAD și un program de desenare constă în modul în care sînt tratate componentele individuale ale desenului. Programele CAD lucrează orientat obiect. Calculatorul memorează deci pentru fiecare obiect desenat descrierea sa matematică și dimensională exactă și îi atașează diferite atribute, cum ar fi de pildă starea, culoarea, tipul, nivelul (layer), care îi stabilesc apartenența la o anumită grupă de obiecte. În plus mai pot fi memorate date despre materiale, firmele furnizoare, indicații de prelucrare, etc.

Un desen CAD este deci o bază de date matematică din care poate fi apelat oricînd un obiect pentru a i se face adăugiri (inserări), modificări, analize și multe altele.

Programele de desenare nu răspund tuturor acestor necesități. În cazul lor imaginea este memorată într-un fișier ca o imagine raster, punct cu punct. Elementele individuale nu pot fi identificate.

Fiecare sistem CAD se compune din cinci componente de bază fără de care un astfel de program nu ar putea lucra:

- - masca ecran
- - unitate de intrare (introducere date)
- - reprezentarea desenului
- - baza de date CAD
- - ieșire și export

Masca ecran

Monitorul (ecranul) este împărțit în mai multe domenii de lucru, care trebuie să rezolve probleme diferite. Domeniul principal este întotdeauna domeniul grafic: în el se desenează. De jur împrejurul lui se ordonează cîmpurile de comandă, care sînt rînduite ca liste text la marginea superioară sau inferioară a ecranului, ca și la marginile sale laterale.

Deoarece un program CAD are foarte multe comenzi, cel mai adesea meniurile sînt aranjate pe mai multe nivele și trebuie "răsfoite" pentru a putea apela comanda dorită.

Diferitele programe utilizează tehnici de operare diferite. O anumită normare a modului de operare transpune doar de curînd, din momentul în care anumite programe au fost transpuse în mediul Windows 3.0. Întrucît modul de operare sub această suprafață utilizat este precis stabilit, toate programele care funcționează sub acest mediu vor avea o deservire asemănătoare.

Din punct de vedere hardware pretențiile referitoare la monitor și la adaptorul grafic sînt ridicate. Deoarece în urma ocupării unei părți de ecran cu cîmpurile de control rămîne relativ puțin spațiu pentru desenul propriu-zis, diagonala ecran trebuie să fie cît mai mare. Dimensiunea ei este limitată însă din considerente financiare și tehnice. Standard sînt utilizate monitoare de 19" cu o rezoluție de 1024x768 puncte. În acest caz se poate lucra profesional. Costurile se păstrează și ele între limite acceptabile. Dacă însă se desenează numai din cînd în cînd, atunci 10.000 DM este mult prea scump. În acest caz poate fi folosit

mulțumitor și un monitor de 14" cu un adaptor grafic VGA.

Unitatea de intrare

În timp ce pe ecran sînt prezentate diferite domenii de control, pentru introducerea datelor posibilitățile sînt limitate. Cea mai simplă, dar și cea mai incomodă, este introducerea datelor de la tastatură. Cu toate acestea pentru introducerea valorilor numerice nu ne putem dispensa de tastatură. În plus aproape toate programele CAD folosesc tastatura ca ultimă soluție în cazul unor erori sistem.

Un mod de comandă mai comod este controlul cursorului prin intermediul mouse-ului. Aproape toate programele CAD permit operarea cu un mouse. Marele avantaj al acestui mod de operare constă în faptul că în acest caz privirea operatorului nu trebuie să se plimbe între mai multe aparate.

Tocmai această comutare a privirii de pe ecran pe aparatul de introducere este singurul dezavantaj al unui aparat de introducere utilizat în domeniul profesional: tabela digitizoare cu lupă. Pe aceasta pot fi aplicate măști specifice soft, pe care să fie reprezentate simbolic toate comenzile.

Reprezentarea desenului

Desenarea este sarcina principală a unui sistem CAD. Utilizatorul trebuie să aibă la dispoziție comenzi de desenare elementare ca-

re să permită desenarea de linii, curbe, cercuri, poligoane, corpuri 3D elementare sau texte. În plus mai trebuie să existe comenzi de ușurare a lucrului; cum ar fi: "capturarea" cursorului într-un anumit punct, la intersecția a două linii. Doar în acest mod pot fi executate desene exacte și la scară.

Mai sînt și alte comenzi care pot ușura lucrul: elementele desenate trebuie să poată fi deplasate, copiate, ogindite, întinse sau scurta-te. Dar aceasta nu este totul: caracteristicile unui element pot fi modificate, mai multe elemente se pot grupa fiind tratate în continuare ca un singur element, elementele grupate pot fi separate din nou, sau pot fi utilizate în continuare ca macrouri, și, și...

Această înșiruire ar putea continua la nesfîrșit, dar se poate observa deja cît de complex este un soft pentru CAD și cîte exerciții sînt necesare pînă să se învețe modul de operare.

Baza de date CAD

Toate elementele și respectiv obiectele unui desen sînt memorate și prelucrate de către programul CAD într-un anumit format. Printre aceste prelucrări se numără, în primul rînd, regăsirea unei construcții deja desenate. Pentru ca acest lucru să fie posibil trebuie respectate anumite convenții la creare și memorare (stocare): la stocarea unui desen pe lîngă numele acestuia mai trebuie stabilite și alte elemen-

te de bază, cum ar fi formatul hîrtiei, unitățile de măsură, nivelul, tipul liniilor, directorul, ș.a.m.d.

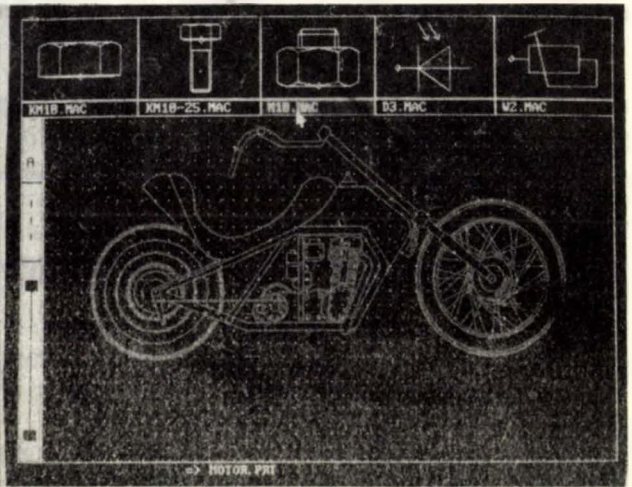
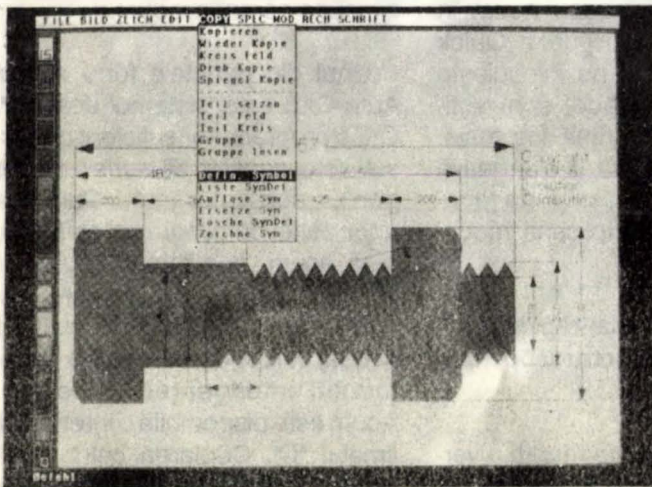
Ieșire și export

Rezultatul unui proces de desenare sau de construcție, în zilele noastre, mai trebuie încă să fie prezentat pe hîrtie. Pentru reprezentarea pe hîrtie sînt disponibile diferite variante: pentru utilizări profesionale și industriale se utilizează un plotter, pentru amatori este suficientă o imprimantă.

A doua interfață importantă spre lumea exterioră o reprezintă formatele de ieșire (export) cunoscute de program. Există mai multe formate de ieșire standard, recunoscute de majoritatea programelor: DXF (format specific Auto CAD), IGES și HPGL.

Există o paletă foarte bogată de programe CAD, cu performanțe, dar și la prețuri, mult diferite. O scurtă enumerare: Easycad 2 (565 DM), Megacad 2.0 (598 DM), Designcad (1140 DM), Generic CADD Level 3 (2000 DM), Geddy Cad (515 DM), Caddy Junior (789 DM), Powercad 1.23 (1298 DM), Auto-sketch 2.0 (350 DM), sub Windows: Drafix Win-CAD 1.1 (2274 DM), Technobox CAD/2 1.2 (2498 DM). Numărul 1 în top, cel care impune standardul, este Auto CAD program a cărui versiune 11.0 v-o prezentăm în continuare.

(R.M.)



Lucrurile bune se obțin cu răbdare

Test: Auto - CAD 11.0

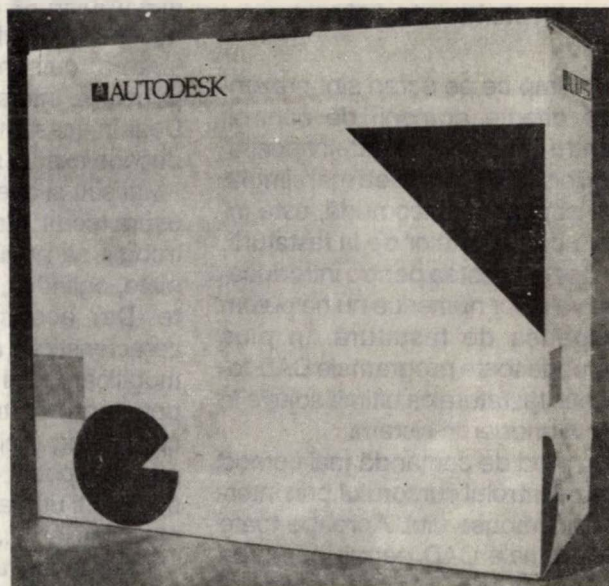
Versiunea 11 a clasicului "Auto-CAD" a fost anunțată încă în urmă cu un an. Acum este disponibilă, în sfârșit, prima versiune, fapt care ne prilejuiește prezentarea unui prim test al produsului.

Producătorii de soft se joacă cu plăcere "de-a șoarecele și pisica": nici nu apucă bine cineva să prezinte un produs nou sau îmbunătățit, că deja concurența anunță un program încă și mai bun. Penibil este însă că atunci când cu toată vîlva care se face, sertarul este gol. Ceva asemănător s-ar fi putut întîmpla și în cazul producătorului de soft Autodesk. Acest lucru ar explica, în orice caz, de ce între anunțarea și momentul lansării pe piață a versiunii "Auto-CAD 11.0" a trecut aproape un an.

Ca întotdeauna, se pare că așteptarea a meritat. Abundența noutăților versiunii Auto CAD 11.0 este de luat în seamă. Pe primul loc ar fi de semnalat facilitățile 3D. Cuvîntul magic pentru acestea este "AME" (Advanced Modeling Extension). În spatele acestei noțiuni se ascunde o nouă procedură de tratare a construcțiilor tridimensionale orientate pe volume, un progres semnificativ față de modelele anterioare orientate pe muchii și suprafețe.

Diferența între reprezentările 3D ale versiunilor 10 și 11 probabil că vor deveni mai clare în urma unui exemplu: pînă acum un corp putea fi creat doar asemănător modului în care se face o construcție din hîrtie. Din afară arăta ca un corp "adevărat" dar în interior nu se putea privi. Noua metodă - orientată pe volume - descrie corpurile și în interior. Întregul mediu cu geometria sa, caracteristicile de material și rezultînd de aici modul de comportare al rezistenței inerției și cen-

Auto-CAD 11.0:
Nu numai designul ambalajului este nou, și în program s-au făcut multe modificări



trului de greutate, sînt descrise prin metoda AME.

Acum, cu ajutorul unei comenzi, pot fi afișate toate valorile relevante și pot fi preluate pentru calcul în aplicațiile învecinate. În afară de acestea se pot executa secțiuni la înălțimea dorită. Construcțiile complicate pot fi realizate prin îmbinarea unor corpuri simple și la nevoie pot fi descompuse din nou.

Strîns legată de reprezentarea 3D este și posibilitatea de umbrire a suprafețelor. Pînă acum Auto-CAD dădea posibilitatea pregătirii unor astfel de "umbriri", dar pentru umbrirea propriu-zisă era necesar programul "Autoshade". Acum în Auto-CAD există funcțiunea "Quick Shade", care deși nu înlocuiește programul Autoshade, este suficientă pentru o primă impresie. "Quick Shade" oferă o alternativă binevenită, pentru comanda "acoperit", tocmai la complicatul "model plasă din sîrmă".

Umbrirea este mai rapidă decît calculul necesar pentru acoperirea (îmbrăcarea) liniilor.

O altă noutate importantă a versiunii 11 o prezintă ecranul de lu-

cru. Acum există posibilitatea definirii unor așa-numite domenii model și hîrtie. În domeniul model este creat desenul prototipului. Aici este prezentată descrierea completă a modelului volumetric, indiferent că este vorba despre un model mic cît o cutie de chibrituri, sau mare cît un teren de fotbal. Cu totul altfel se prezintă situația în domeniul hîrtiei. Aici pot fi definite sau mascate diferite ferestre care să conțină numai anumite detalii ale modelului.

Ferestrele pot fi aduse împreună pe formatul de hîrtie dorit și listate prin intermediul unui plotter sau a unei imprimante. În acest mod pot fi create cu ușurință desene de execuție pentru producție.

Unul din punctele forte ale lui Auto-CAD este "sistemul deschis". Orice programator suficient de versat este capabil să scrie utilitare pentru acest program. Cuplarea unor astfel de programe se făcea, pînă acum, prin intermediul unui limbaj propriu de programare "Autolisp", un limbaj înrudit cu Lisp (limbaj utilizat cu precădere în domeniul inteligenței artificiale). Acum este disponibilă o interfață la limajul "C". Cuplarea celor două limbaje de programare se dove-

dește a fi aproape ideală. Acest lucru fiind mijlocit de 'ADS' (Auto-CAD Development System). Prin intermediul ADS pot fi integrate programe externe, indiferent de faptul că ele sînt programe proprii, scrise de utilizator, sau sînt pachete soft deja existente, cum ar fi: programe de calcul tabelar, baze de date sau soluții specifice branșei.

Auto-CAD 11.0 poate fi utilizat și în rețea, funcționînd pe toate sistemele de operare de rețea, cum ar fi Novell, 3Com sau NFS. Posibilitatea de a fi instalat și pentru un post individual de lucru, a rămas. În cazul ambelor moduri de instalare este disponibilă o procedură de logare (de deschidere a unei sesiuni de lucru) extinsă. Acum pot fi definite parole care să dea certitudinea că la anumite date au acces

numai utilizatorii specificați. De asemenea pot fi blocate fișierele, ceea ce dă siguranță că un anumit desen nu este prelucrat simultan de mai mulți utilizatori. Extensiile pentru lucrul în rețea implică un efort de instalare mai mare, dar o parte semnificativă a acestui efort este preluată de procedura de instalare.

Ținînd cont de multitudinea noutăților, nu este de mirare că Auto-CAD 11.0 nu poate fi rulat decît pe calculatoare cu 386 cu coprocessor și respectiv pe calculatoare cu 486. Memoria minimă necesară este de 2 MByte. Dacă se dorește utilizarea AME sau a funcțiilor de umbră atunci sînt necesari cel puțin 4 MByte. Auto-CAD 11.0 știe să lucreze și cu memorie virtuală pe harddisk, dar aceasta este mai lentă decît memoria RAM. Deci: cu cît există mai multă memorie, cu

atît este mai bine. Auto-CAD recunoaște standardul VCPI (Virtual Control Program Interface). Acest standard include convențiile stabilite de mai mulți producători pentru stabilirea modului în care sistemul de operare DOS trebuie să trateze memoria "extinsă" și "expandată". Pentru aceasta sînt necesare drivere de memorie care să respecte standardul VCPI. Dintre acestea enumerăm: "Compaq CEMM 4.01", "Quarterdeck QEMM 4.01" și "386-MAX" al firmei Qualitas System.

Concluzionînd așteptarea lui Auto-CAD 11.0 a meritat. Noile facilități nu sînt dedicate în primul rînd utilizatorilor obișnuiți, dar le sînt indispensabile utilizatorilor profesionali.

Preț: Auto-CAD 11.0 7296 DM, cu AME 7866 DM.

(R.M.)

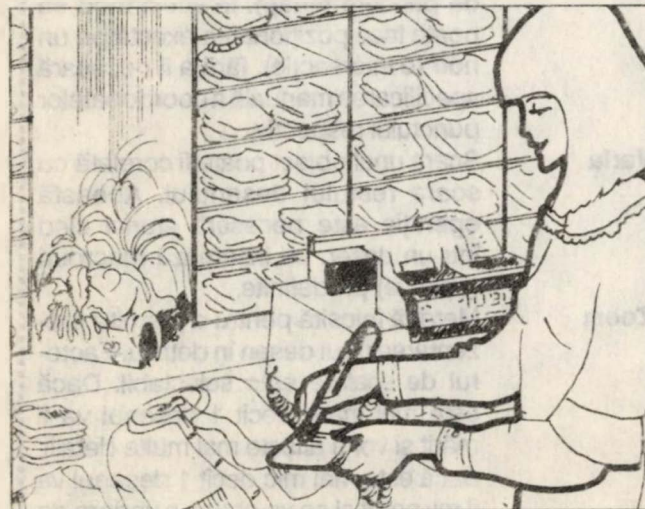
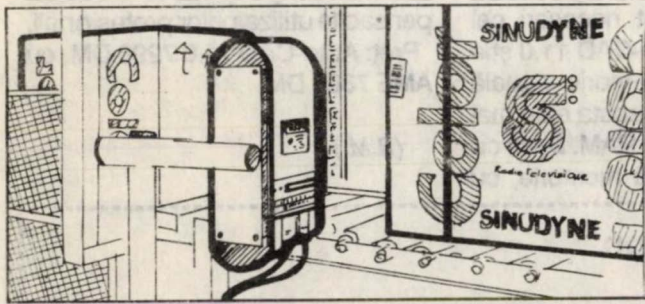
Lexicon

Element	Cea mai mică unitate de desen, de ex. o linie, un arc de cerc sau un cerc		reprezentat în tuș, sau carioca, pe hîrtie
Fișier DXF	Format fișier pentru schimbul de desene între diferite programe CAD	Polyline	Trasarea unei linii care constă din mai multe segmente de linii, cu o singură operație
Grid	Rastru. Pe monitor este afișat un rastru pentru orientare. Acesta nu va fi listat pe plotter sau imprimantă	Printplot	Reprezentarea desenului pe o imprimantă de rezoluție ridicată. Nu se atinge însă rezoluția unui plotter.
Grupă	Obiectele individuale pot fi incluse în grupe pentru a putea fi prelucrate împreună. Grupările pot fi descompuse din nou, la nevoie.	Simbol (Macrou)	Detaliile utilizate frecvent pot fi memorate ca macroui și ulterior pot fi inserate la nevoie și în alte desene.
Hatch	Hașură. Hașurarea trebuie să respecte normele DIN	Snap	Cursorul poate fi poziționat numai în anumite puncte definite prin valoarea de prindere (snap). În acest mod se poate face poziționarea exactă pe un nod (o intersecție), fără a fi necesară specificarea manuală a coordonatelor punctului respectiv.
HPGL	Limbaj de comandă pentru controlul plotter-elor (Hewlett Packard Graphics Language).	Varia	Scara unui obiect poate fi corelată cu scara restului desenului. Această operație este necesară atunci cînd într-un desen se inserează macroui (obiecte) predefinite.
Layer	Nivel desen: asemănător modului de lucru cu folii pe un diaproiector. Un nivel poate fi mascat, aceasta înseamnă că nu va fi afișat pe ecran.	Zoom	Metodă folosită pentru a permite prezentarea unui desen în detaliu. Factorul de scalare este selectabil. Dacă este mai mare decît 1 desenul va fi mărit și vor fi afișate mai multe detalii, dacă este mai mic decît 1 desenul va fi micșorat și se va obține o vedere de ansamblu mai bună.
Listă de componente	Lista simbolurilor care apar într-un desen și cantitatea în care apar.		
Ortho	Nu mai pot fi desenate decît linii orizontale sau verticale. Unele programe permit specificarea unui unghi cu care să se lucreze în continuare.		
Pan (Panorama)	Deplasarea secțiunii vizibile din desen		
Plotter	Masă de desen automată. Aparat de ieșire cu ajutorul căruia desenul este		

Cod de bare

Printre cele mai mari probleme ale întreprinderilor industriale și comerciale moderne se numără și felul în care funcționează magazinele, felul în care se realizează fluxul tehnologic. Cît anume din timpul de realizare al unei mașini mai complexe este timp de lucru/asamblare efectiv și cît anume este timp "de așteptare" în diverse faze ale trecerii respectivului produs prin fabrică este un factor pînă la urmă extrem de important, care se reflectă direct atît în productivitate cît și în preț. Cerințele din ce în ce mai dure în această direcție au determinat dezvoltări în două direcții principale:

- utilizarea unor sisteme de transport moderne



- utilizarea calculatoarelor la urmărirea fluxului de materiale

Pentru o utilizare consecventă a calculatoarelor în urmărirea fluxului de materiale în toate fazele, este inevitabilă identificarea directă a încărcăturii transportate, în fiecare ipostază (pachet, palet, container etc.).

Datorită progreselor însemnate din opto-electronică, în prezent este posibilă citirea automată a datelor înregistrate (direct) pe încărcătura transportată și transmisia lor, printr-o interfață standard, spre calculator, pentru a fi prelucrate. Datele de identificare de pe fiecare încărcătură trebuie să fie codate sau înregistrate într-un fel oarecare; cel mai utilizat sistem, în prezent este codul de bare. Datele înregistrate sub formă de cod de bare se tipăresc de regulă direct pe ambalaje, avize de expediție, fișe de magazie, bilete de avion, legitimații, etichete etc.

În repaus, datele astfel codate pot fi citite manual, folosind un "creion de citire" pentru a le prelua și disponibiliza calculatorului. Unde acest lucru nu are sens - de exemplu la instalații unde se lucrează la bandă, cu încărcături mobile - datele de identificare pot fi preluate (de la o anumită distanță) prin intermediul unor laser - scannere cu He și Ne sau al unor camere de luat vederi CCD.

Preluarea automată a datelor tipărite oferă utilizatorului diverse posibilități:

- generarea tuturor documentelor ce privesc depozitarea și expediția (aviz de expediție, fișă de urmărire, chitanță, etc.)

- controlul automat al întregului sistem de transport și depozitare

- inventar permanent la zi și (prin aceasta) date exacte relativ la stocurile existente (de produse și/sau materii prime); ca o bază pentru deciziile conducerii.

Nenumărate sînt exemplele de utilizare ale aplicațiilor construite pornind de la codul de bare:

- în biblioteci (și/sau videoteci): toate cărțile sînt identificate prin intermediul unui cod de bare. De asemenea, fiecare utilizator are o legitimație, identificată printr-un cod de bare. Toate împrumuturile se fac prin intermediul calculatorului, fără hîrtie.

- în depozitele de componente electronice (sau alte articole mărunte) intrările și ieșirile de marfă se înregistrează prin intermediul calculatorului. Prelucrarea codului fieșării articol se face prin intermediul codului de bare, aplicat pe ambalaje, și al "creionului de citire".

- legitimațiile sau abonamentele se recunosc folosind cititoare cu fantă; astfel, se poate da acces la calculatoare, intrare la biblioteci, schi-lift, domenii miștate etc.

- pe benzi de fabricație, codurile de bare aplicate pe cartoane permit preluarea datelor - prin laser-scanner montat fix, într-o anumită poziție, astfel încît citirea să

se facă de la o distanță de 200 - 800 mm - și transmiterea lor la nivelul superior.

Codarea



La concurență cu alte sisteme de codare care să permită identificarea automată sau semiautomată a diverselor entități ce trebuie urmărite, codul de bare are un avantaj indiscutabil: este relativ simplu de produs și relativ simplu de recunoscut.

Sistemul OCR se bazează pe recunoașterea caracterelor alfabetice cu care sîntem obișnuiți, cu restricția că forma caracterelor trebuie să fie standardizată pentru a permite citirea automată. Avantajul clarității - datele nu trebuie codate - este anulat de aparatele de citire și recunoaștere complicate, precum și de cerințele ridicate vizavi de calitatea imprimării. La acesta se adaugă rate de erori mari: cercetări făcute în aplicațiile tip supermarket au evidențiat peste 30% citiri nereușite din prima încercare și peste 1% citiri eronate.

Sistemul magnetic cere și el un efort tehnic și financiar deosebit. Cînd codul trebuie produs în cantități numerice importante, limita economicității este foarte repede depășită.

Codul de bare poate fi aplicat direct pe orice marfă (tub, carton, cutie, sticlă, etc.) sau poate fi aplicat ulterior, ca etichetă. În domeniul alimentar, codul EAN

Terminologie

Linie	elementul întunecat al unui cod	cu un caracter de stop. Astfel citirea se poate face în ambele direcții.
Spațiu	elementul deschis dintre două linii ale unui cod	
Interspațiu	spațiul dintre ultima linie a unui caracter și prima linie a următorului caracter în cadrul unui cod de baze discret	Cod cu autoverificare - cod de bare care permite verificarea fiecărui caracter, după un algoritm prestabilit. Erorile de substituție sînt posibile
Element	expresie folosită pentru a descrie o linie sau un spațiu component al unui cod	Cod discret - cod de bare în care fiecare semn începe și se termină cu o linie. Interspațiul nu face parte din cod.
Modul	elementul cel mai îngust dintr-un cod. Liniiile sau spațiile mai late se calculează ca multipli ai modulului	Cod continuu - cod de bare la care interspațiul face parte din cod
Lărgimea modulului X:	definește lărgimea celui mai îngust element (de ex. $X=0,33$ mm) Zona "de liniște" - zona de culoare deschisă dinaintea caracterului de start și de după caracterul stop. Zona de liniște este necesară pentru a da cititorului informația de "început de cod". Minimul este de 10 ori lărgimea modulului, dar cel puțin 2,5 mm. La aplicații ce folosesc scannere cu un domeniu de contrast mai mare, zona de liniște trebuie aleasă mai mare ($L = 15 \times$ lărgimea modulului, dar cel puțin 6,5 mm).	Cifră de control - unul sau mai multe semne suplimentare, care permit recunoașterea erorilor
Simbol în cod de bare	un cod de bare complet, începînd cu o zonă deschisă înaintea codului, un caracter de start, unul sau mai multe caractere utile, un caracter de stop și încă o zonă deschisă. Dedesubt, un rînd cu textul în clar.	Eroare de substituție - un semn e înlocuit de un altul. Numai prin utilizarea unei cifre de control, acest tip de eroare poate fi aproape exclus
Caracter de start/stop	- orice cod începe cu un caracter de start și se termină	Citire nereușită - decodorul nu este în stare să decodifice informația citită
		Citire falsă - informația decodată nu coincide cu cea din codul de bare (eroare de substituție).
		Rata de primă citire - numărul de citiri corecte la prima încercare, împărțit la numărul de încercări
		Cod de fotolmpriamare - prin fotoimprimare, pot fi produse coduri de orice densitate
		Cod de densitate mare - cod foarte strîmt, produs prin fotoimprimare sau cu imprimante cu ciocănele
		Cod de densitate medie - cod produs cu imprimante matriceale, termice sau cu jet de cerneală (modul 0,3 - 0,5 mm)
		Cod de densitate mică - cod produs cu imprimante matriceale, termice sau cu jet de cerneală (modul 0,5 mm).

a devenit practic standard european; în domeniul medico-chimical, s-a afirmat codul CODABAR, iar codurile Code 2/5 5 linii sau codul 2/5 interleaved se utilizează în mai toate domeniile industriale.

Cele mai multe din codurile uzuale se bazează pe un principiu binar, cu un număr de linii/spații înguste sau largi. Secvența acestor linii respectiv spații determină un anumit conținut numeric sau chiar alfanumeric.

Citirea se face optic. Prin reflexia diferențiată a liniilor negre și spațiilor albe, în receptorul optic apare un tren de impulsuri care corespunde acestei secvențe de linii și spații.

Prin codare acest tren de impulsuri este apoi interpretat ca secvență de date.

Mărimea codului este determinat în principiu de parametrii aplicației - de exemplu, felul caracterelor de reprezentat (numerice, alfanumerice) prin numărul de caractere, spațiul disponibil, posibilitățile de imprimare, etc.

Toți acești factori au făcut ca în decursul timpului pentru diverse aplicații să apară o multitudine de coduri de bare, care toate își au (sau respectiv aveau) justificarea. Cîteva din cele mai utilizate coduri, inclusiv tabelele de adevăr corespunzătoare, sînt prezentate în paginile următoare.

Cod 2/5 5 linii industrial

Generalități - cod numeric; pot fi reprezentate cifrele 0-9. Acest cod e construit cu două linii late și trei linii înguste. Raportul de imprimare linie îngustă:linie lată (R) este de 1:2 pînă la 1:3. Spațiile nu conțin informații.

Avantaje - codul constă numai din linii, în spații nu este cuprinsă informație. Toleranța e mare (15%), de aceea poate fi produs prin cele mai simple procedee de imprimare

Dezavantaje - densitate mică a informației (4,2 mm/cifră la o lățime a modulului $x = 0.3\text{mm}$ și $R = 1:3$.)

Tabela de coduri:

Caracterul	L1	L2	L3	L4	L5
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0
0	0	0	1	1	0
start	1	1	0		
stop	1	0	1		

L1-L5 = Linia 1 - 5
1 = Linie lată
0 = Linie îngustă

Exemplu:



Cod 2/5 interleaved

Generalități - cod numeric; pot fi reprezentate cifrele 0-9. Acest cod e construit cu 2 linii late și 3 linii înguste. Raportul de imprimare (R): linie îngustă: linie lată 1:2 pînă la 1:3, spațiu îngust: spațiu lat 1:2 pînă la 1:3. Dacă cel mai îngust element e mai mic de 0,5 mm, atunci linie (spațiu) îngustă linie (spațiu) lat = 1:2,25. Prima cifră este reprezentată prin 5 linii, iar a doua cifră prin spațiile ce urmează nemijlocit liniilor primei cifre:

Avantaje: densitate mare de informație. (2,7 mm/cifră la o lățime a modulului de $x 0,3 \text{ mm}$ și un raport de imprimare $R = 1:3$); autoverificabil

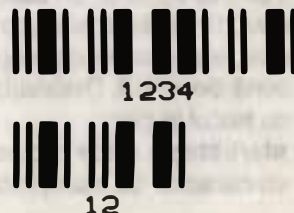
Dezavantaje: toate spațiile poartă informație, de aceea toleranța trebuie să fie mai mică de 10%.

Tabela de coduri:

Caracterul	L1	L2	L3	L4	L5
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0
0	0	0	1	1	0
start	0	0			
stop	1	0			

L1-L5 = Linia / Spațiul 1 - 5
1 = Linie/Spațiu lat(ă)
0 = Linie/Spațiu îngust(ă)

Exemplu:



Code 2/5 3 linii (Matrix)

Generalități - cod numeric; pot fi reprezentate cifrele 0-9. Fiecare caracter constă din 5 elemente, 3 linii și două spații. Două elemente sînt late și 3 sînt înguste. Deci atît liniile cît și spațiile pot fi late sau înguste. Inter-spațiul nu poartă informație. Raportul de imprimare (R) linie îngustă: linie lată = 1:2,25 pînă la 1:3; spațiu îngust: spațiu lat = 1:2,25 pînă la 1:3. Linie deosebit de lată în caracterul de start și cel de stop.

Avantaje: - densitate mare de informație (3,0 mm/cifră la o lățime a modulului x = 0,3 mm și R = 1:3) Nu există informație între cifre

Dezavantaje: toleranță <= 10% Caracter supralat în informația de start/stop.

Tabela de coduri:

Caracterul	L1	S1	L2	S2	L3
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0
0	0	0	1	1	0
start	11	0	0	0	0
stop	11	0	0	0	0

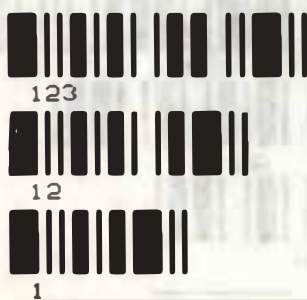
L1-L3 = Linia 1 - 3, S1-S2 = Spațiul 1 - 2

1 = Linie/Spațiu lat(ă)

0 = Linie/Spațiu îngust(ă)

11 = 1,5 x Linie lată

Exemplu:



Cod 2/5 3 linii (Datalogic)

Generalități - cod numeric, pot fi reprezentate cifrele 0-9. Fiecare caracter constă din 5 elemente, 3 linii și două spații. Două din elemente sînt late și 3 sînt înguste. Deci atît liniile cît și spațiile pot fi late sau înguste. Interspațiul nu poartă informație. Raportul de imprimare (R): linie îngustă: linie

lată = 1:2,25 pînă la 1:3; spațiu îngust:

spațiu lat = 1:2,25 pînă la 1:3

Avantaje - densitate mare de informație (3,0 mm/cifră

la o lățime a modulului x = 0,3 mm și R =

1:3) Nu există informație între cifre

Dezavantaje - toleranță mai mică de 10%.

Tabela de coduri:

Caracterul	L1	S1	L2	S2	L3
1	1	0	0	0	1
2	0	1	0	0	1
3	1	1	0	0	0
4	0	0	1	0	1
5	1	0	1	0	0
6	0	1	1	0	0
7	0	0	0	1	1
8	1	0	0	1	0
9	0	1	0	1	0
0	0	0	1	1	0
start	11	0	0	0	0
stop	11	0	0	0	0

L1-L3 = Linia 1 - 3, S1-S2 = Spațiul 1 - 2

1 = Linie/Spațiu lat(ă)

0 = Linie/Spațiu îngust(ă)

Exemplu



Codabar

Generalități - cod numeric cu 6 caractere speciale. Pot fi reprezentate cifrele 0-9, minus (-), dolar (\$), două puncte (:), slash (/), punct (.), plus (+). Fiecare caracter constă din 7 elemente, 4 linii și 3 spații. Se folosesc 2 sau 3 elemente late și 5 sau 4 elemente înguste pentru reprezentarea codului. Interspațiul nu poartă informație. Raportul de imprimare: (R) linie îngustă: linie lată (1:2) pînă la 1:3; spațiu îngust: spațiu lat 1:2 pînă la 1:3 Dacă elementul îngust e mai mic de 0,5 mm, atunci element îngust: element lat = 1:2,25.

Avantaje: în afară de cifre, pot fi reprezentate încă 6 caractere speciale. Interspațiul nu poartă informație.

Dezavantaje: densitate de informație redusă. (5,5 mm/cifră la o lățime a modului de x = 0,3 mm și R = 1:3.)

Tabela de coduri:

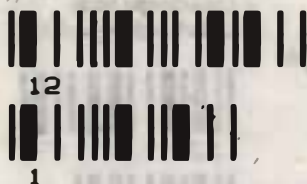
Caracterul	L1	S1	L2	S2	L3	S3	L4
1	0	0	0	0	1	1	0
2	0	0	0	1	0	0	1
3	1	1	0	0	0	0	0
4	0	0	1	0	0	1	0
5	1	0	0	0	0	1	0

Aplicații

6	0	1	0	0	0	0	1
7	0	1	0	0	1	0	0
8	0	1	1	0	0	0	0
9	1	0	0	1	0	0	0
0	0	0	0	0	0	1	1
.	0	0	0	1	1	0	0
\$	0	0	1	1	0	0	0
'	1	0	0	0	1	0	1
/	1	0	1	0	0	0	1
-	1	0	1	0	1	0	0
+	0	0	1	0	1	0	1
A	0	0	1	1	0	1	0
B	0	1	0	1	0	0	1
C	0	0	0	1	0	1	1
D	0	0	0	1	1	1	0

L1-L4 = Linia 1 - 4, S1-S3 = Spațiul 1 - 3
 1 = Linie/Spațiu lat(ă)
 0 = Linie/Spațiu îngust(ă)

Exemplu



Cod 39

Generalități - cod alfanumeric. Pot fi reprezentate cifrele 0-9, 26 de litere, 7 caractere speciale. Fiecare caracter constă din 9 elemente (5 linii și 4 spații). 3 elemente sînt late și 6 sînt înguste, cu excepția caracterelor speciale. Interspațiul nu poartă informație.

Raportul de imprimare (R) linie îngustă: linie lată = 1:2 pînă la 1:3; spațiu îngust: spațiu lat = 1:2 pînă la 1:3. Dacă elementul îngust e mai mic decît 0,5 mm, atunci elementul îngust/element lat = 1:2,25

Avantaje: reprezentare alfanumerică

Dezavantaje: densitate mică de informație (4,8 mm/cifră la o lățime a modului de $x = 0,3$ mm și $R = 1:3$). toleranță mică (10%).

Tabela de coduri:

Caracterul	L1	S1	L2	S2	L3	S3	L4	S4	L5
1	1	0	0	1	0	0	0	0	1
2	0	0	1	1	0	0	0	0	1
3	1	0	1	1	0	0	0	0	0
4	0	0	0	1	1	0	0	0	1
5	1	0	0	1	1	0	0	0	0
6	0	0	1	1	1	0	0	0	0
7	0	0	0	1	0	0	1	0	1
8	1	0	0	1	0	0	1	0	0
9	0	0	1	1	0	0	1	0	0
0	0	0	0	1	1	0	1	0	0
A	1	0	0	0	0	1	0	0	1

B	0	0	1	0	0	1	0	0	1
C	1	0	1	0	0	1	0	0	0
D	0	0	0	0	1	1	0	0	1
E	1	0	0	0	1	1	0	0	0
F	0	0	1	0	1	1	0	0	0
G	0	0	0	0	0	1	1	0	1
H	1	0	0	0	0	1	1	0	0
I	0	0	1	0	0	1	1	0	0
J	0	0	0	0	1	1	1	0	0
\$	0	1	0	1	0	1	0	0	0
/	1	1	0	1	0	0	0	1	0
K	1	0	0	0	0	0	0	1	1
L	0	0	1	0	0	0	0	1	1
M	1	0	1	0	0	0	0	1	0
N	0	0	0	0	1	0	0	1	1
O	1	0	0	0	1	0	0	1	0
P	0	0	1	0	1	0	0	1	0
Q	0	0	0	0	0	0	1	1	1
R	1	0	0	0	0	0	1	1	0
S	0	0	1	0	0	0	1	1	0
T	0	0	0	0	1	0	1	1	0
U	1	1	0	0	0	0	0	0	1
V	0	1	1	0	0	0	0	0	1
W	1	1	1	0	0	0	0	0	0
X	0	1	0	0	1	0	0	0	1
Y	1	1	0	0	1	0	0	0	0
Z	0	1	1	0	1	0	0	0	0
.	0	1	0	0	0	0	1	0	1
'	1	1	0	0	0	0	1	0	0
BLANC	0	1	1	0	0	0	1	0	0
+	0	1	0	0	0	1	0	1	0
%	0	0	0	1	0	1	0	1	0
start/stop,0	0	1	0	0	1	0	1	0	0

L1-L5 = Linia 1 - 5, S1-S4 = Spațiul 1 - 4
 1 = Linie/Spațiu lat(ă)
 0 = Linie/Spațiu îngust(ă)

Exemplu:



Cod EAN

Generalități - cod numeric, pot fi reprezentate cifrele 0-9. Fiecare caracter constă din 11 elemente. Toate liniile și spațiile poartă informație. Pot fi reprezentate numai grupuri de 8 sau 13 caractere împreună.

Avantaje - densitate mare de informație în 10 dimensiuni diferite.

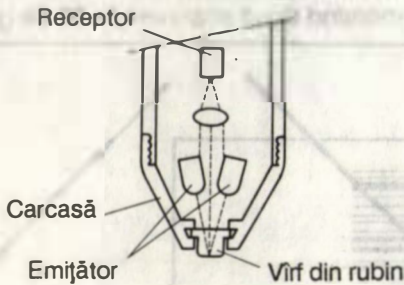
Dezavantaje - toleranțe foarte mici.

Tabela de coduri este prea complicată pentru a mai fi exemplificată acum. Ea poate fi solicitată de la Centrale für Coorganisation, Postfach 19 04 24, Spichernstrasse 55, 5000 Köln 1.

Tehnici de citire

Cititoare statice

Creioanele de citire funcționează pe baza efectului ilustrat în figură. Emițătoarele emit lumină roșie (630 mm) sau infraroșie (950 mm); creionul este "dus" deasupra hîrtiei, atingînd-o în permanență. Suprafața ce conține codul de bare e luminată difuz. Prin intermediul unui sistem optic complex, lumina reflectată ajunge la fototranzistor. La parcurgerea codului de bare, sistemul transformă optic liniile și spațiile într-un tren de impulsuri. Acesta este disponibilizat în formă numerică la ieșire. Un decodor atașat transformă trenul de impulsuri în date pe care le transmite serial (printr-o interfață RS232, RS422 sau 20mA)

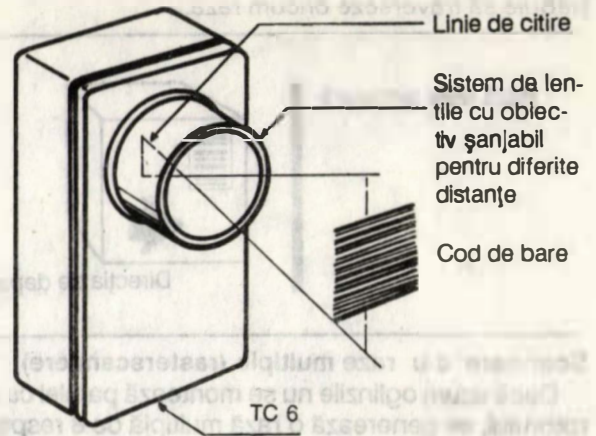


Cititoarele distanțoare folosesc același sistem, cu diferența că citirea se face la o distanță de 10 mm \pm 1 mm. Înălțimea minimă a codului de bare este de 3mm.

Cititoare dinamice

Sistemul cu cameră de luat vederi

Un rînd de fotodiode este dispus în spatele focarului, ca la o cameră suprafața filmului. Prin interogare internă rapidă a punctelor individuale, un cod de bare este transformat astfel într-un rastru de puncte. Cel puțin 4 pixeli trebuie să fie proiecția unei linii înguste pentru ca aceasta să poată fi recunoscută. Utilizînd obiective variabile, pot fi recunoscute coduri de bară la pînă 4 m distanță. Ca funcție, camera corespunde unui scanner cu rază singulară. Numărul de citiri pe secundă (scans) este de pînă în 1000.



Scannere cu He - Ne

Este cunoscută reflexia unei raze de lumină printr-o oglindă netedă, la care unghiul de reflexie coincide cu unghiul de incidență.

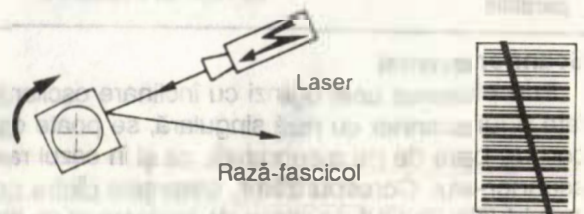
Dacă 4 sau mai multe oglinzi se montează pe un rotor și se reflectă razele luminoase ale soarelui sau ale unei surse de lumină artificiale atunci pe o suprafață se proiectează un punct care se deplasează pe un traseu rectiliniu, repetitiv. Dacă rotorul se învîrte suficient de repede, atunci funcție de turația lui și numărul de oglinzi, punctul luminos face pînă la 800 de parcurgeri ale traseului pe secundă (scans) pe suprafața pe care se proiectează.

De ce laser cu He - Ne?

Laserul are proprietatea de a emite lumină aproape paralelă. Astfel e posibilă reflexia ordonată a luminii cu un efort optic rezonabil (în principal, elemente optice netede, oglinzi). Prin paralelismul dat, citirea se poate face cu multă exactitate în profunzime.

Scanner cu rază singulară

Folosind ca sursă de lumină un laser cu He - Ne, se ajunge la principiul de funcționare al unui scanner cu laser He - Ne. Raza de lumină generată are un diametru de 0,2 - 0,6 mm. Dacă oglinzile se montează apoi paralel cu axa de rotație a rotorului, se proiectează o rază singulară de lumină.



Acest scanner se folosește cînd pe o bandă transportoare liniile de cod sînt paralele cu direcția deplasării iar direcția citirii este în unghi drept față de aceasta. Liniile sînt orientate orizontal, iar raza laser verticală. Avantajul este că nu trebuie realizată o precizie prea mare a înălțimii la care se plasează codul (\pm 600 mm).

Aplicații

În afară de aceasta, prin deplasare, codul de bare trebuie să traverseze oricum raza.

Rază laser singulară



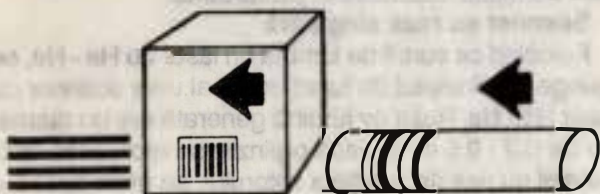
Direcția de deplasare

Scannere c u raze multiple (raster-scannere)

Dacă acum oglinzile nu se montează paralel cu axa rotorului, se generează o rază multiplă de 8 respectiv 10 raze - rastrul.



Raster-scanner-ul se folosește atunci când liniile codului pot fi aplicate numai vertical față de direcția de deplasare a încărcăturii (imprimanta poate tipări numai într-o direcție, eticheta poate fi aplicată numai într-o anumită poziție, etc.). Astfel se asigură o anumită toleranță în ce privește înălțimea codurilor.



Fascicul de raze paralele

Direcția de deplasare

Scanner evantai

Prin plasarea unei oglinzi cu înclinare oscilantă în fața unui scanner cu rază singulară, se poate citi un cod de bare de pe o suprafață, ca și în cazul raster-scanner-ului. Corespunzător, distanțele dintre razele individuale depind de viteza de înclinare și de amplitudinea oscilațiilor

Citirea omnidirecțională

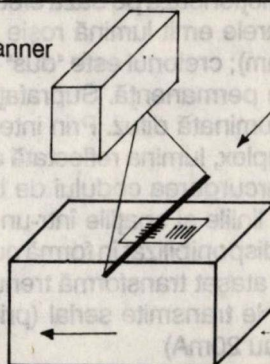
Dacă lucrurile ce se identifică nu au o orientare totdeauna aceeași, folosind un cod adecvat, acesta poate fi citit omnidirecțional.

Eticheta în T

Acest cod se aplică de două ori, sub un unghi de 90. Premisa necesară este ca înălțimea codului (lungimea liniilor) să fie mai mare decât lățimea.



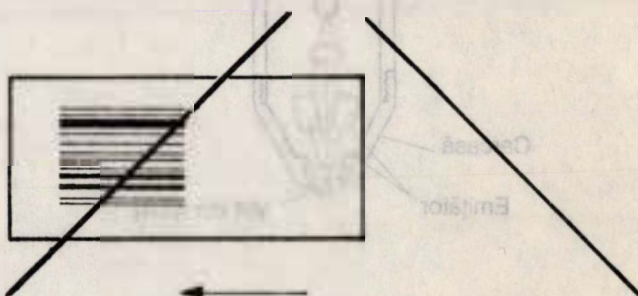
Scanner



Rază de scannare

Direcția de deplasare

Citirea omnidirecțională cu două scannere cu rază individuală este o altă posibilitate, folosind un cod simplu, dar montând două scannere la 90 de grade.



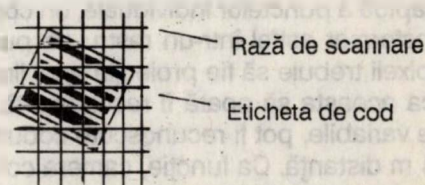
Scanner 1

Direcția de deplasare

Scanner 2

Scannere omnidirecționale

Cu un scanner care nu emite numai unul sau mai multe raze paralele, ci o rețea completă, pot fi citite apoi toate codurile care au o construcție cel puțin pătratică (lungimea liniilor = lățimea lor), indiferent de poziția codului față de scanner. Cel mai adesea, astfel de sisteme se utilizează la casele de marcat. Din păcate, profunzimea în citire este limitată la circa 15 cm.



Rază de scannare

Eticheta de cod

În final, vă prezentăm și câteva dispozitive, împreună cu aplicațiile tipice. Ilustrațiile se bazează pe prospecte ale firmei Datalogic. Relații suplimentare relativ la gama de produse oferite pot fi obținute direct (Data-

Logic International GmbH, Dreisteinstrasse 49 b, A - 2372 Giesshubel bei Wien, Austria). Altă firmă "cu nume" în domeniu: APOG, Paris, 298 av. G'ral de Gaulle, 92.140 Clamart, France.



Creion de citire

Aplicații:

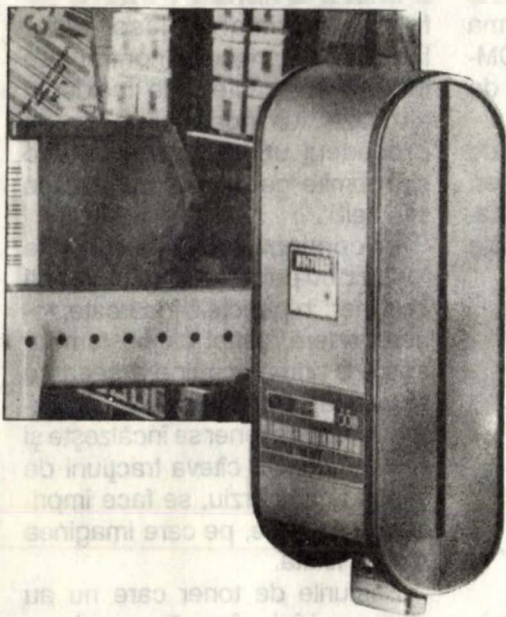
- case de marcaj în magazine cu autoservire
- inventarieri
- identificarea de pachete, paleți, scrisori, acte, cărți, fotografii, probe de date de producție, administrație, laboratoare



Microcalculator pentru introducerea datelor "pe teren" prin tastatură sau creion de citire

Aplicații:

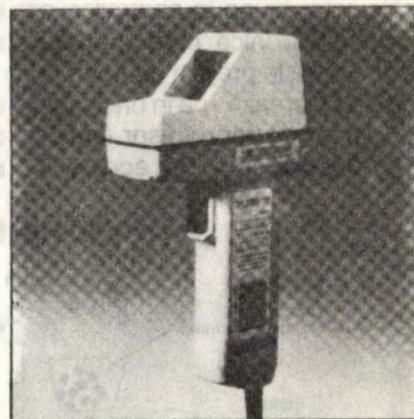
- preluarea și memorarea datelor "pe teren" - inventare - consignație - preluare de comenzi - preluarea unor date de producție - supravegerea producției - pontaj



Laser-scanner cu He-Ne

Aplicații:

- preluarea codurilor de bare fixate pe hîrtie, carton, folii sau etichete
- identificarea automată a bagajelor la aeroporturi
- identificare pachete, paleți etc.



Pistol de citire cod de bare

Aplicații

- citirea fără atingere a diverse coduri de bare, în toate domeniile tehnicii și comerțului
- preluarea de coduri aplicate pe hîrtie, carton, etichete, suport textil, în poziții oarecari
 - casă și preluare de marfă
- identificarea unor părți de montaj pe linii de asamblare

Cum funcționează imprimantele laser?

Gutenberg ar fi impresionat: doar de mărimea unei valioare, imprimantele laser fac adevărate minuni și duc pe culmi noi seculara artă a tipografilor; toate acestea, pentru prețuri în jur de 3.000 DM. Din punct de vedere tehnic, imprimantele laser sînt într-adevăr realizări de excepție: o mecanică lucrînd cu o precizie aproape absolută și o microelectronică extrem de rapidă sînt necesare pentru ca textele să ajungă pe hîrtie într-o manieră care să satisfacă și cele mai exigente gusturi.

Primul în șiragul de "muncitori" cu înaltă calificare este calculatorul, pe care utilizatorul își scrie textul. Din clipa în care utilizatorul vrea să-și imprime textul, interfața calculatorului preia aceste date sub formă de semnale electrice și le transmite direct interfeței imprimantei. Important e desigur ca cele două interfețe să se înțeleagă și să transmită (respectiv să preia) datele fie în șir indian, una cîte una ("serial"), fie în coloană de cîte opt deodată ("paralel").

Ajunse la imprimantă, datele de la calculator ajung în controller, o unitate electronică care prepară datele spre a le pregăti de tipar. În controller lucrează - ca în orice calculator - un microprocesor (creierul calculatorului). Pe acesta îl

ajută, ca și în calculator, așa-numita memorie nevolatilă (ROM - Read Only Memory). În ROM-uri sînt stocate programele care controlează funcționarea imprimantei. Datele preluate de imprimanta laser sînt depozitate temporar în memoria volatilă - RAM - (Random Acces Memory). Aceasta pentru că, spre deosebire de imprimantele matriceale, imprimantele laser trebuie să-și construiască întîi imaginea unei pagini întregi, înainte de a putea imprima chiar și o singură literă. Motivul: în timp ce imprimanta matriceală poate transfera imediat pe hîrtie semnalele de la calculator, deoarece lucrează : "rînd cu rînd", imprimanta laser imprimă numai pagini complete, pe care pune texte, imagini grafice sau ambele.

Deosebit de multă memorie în imprimantele laser necesită imaginile grafice, căci ele trebuiesc reprezentate punct cu punct în memorie. În contrast, literele, deși pe hîrtie constau deasemenea din puncte individuale, au nevoie de foarte puțină memorie. Căci formelor e deja preprogramată în ROM-uri, astfel că la nevoie se preia de acolo.

Din clipa în care toate datele de la calculator au ajuns la controller, acesta începe cu calculele necesare pentru a le transforma în puncte

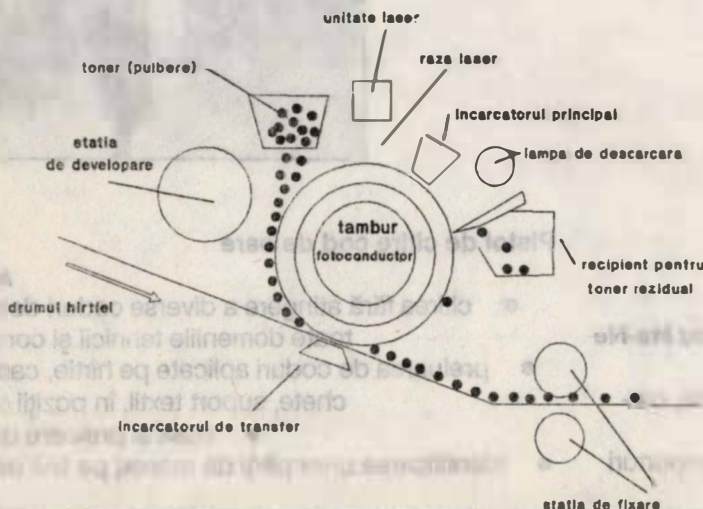
individuale. Căci pentru imprimanta laser, pagina imprimată nu constă din litere, ci din aproximativ 8.000.000 de puncte. Linie după linie de puncte este pregătită de controller și trimisă unității de expunere. Nucleul acesteia este un tambur cu strat superficial fotosensibil, lat cît o pagină tipărită (de regulă A4). Acest tambur se rotește și este încărcat electric de încărcătorul principal. Rotația tamburului face ca fiecare pistă încărcată să treacă prin dreptul unei diode laser, care prin intermediul unui sistem mecanic de oglinzi ghidează raza laser astfel încît să ajungă pe pistă. Pentru a ocoli această mecanică complicată, unii producători folosesc diode luminescente (LED-uri) în locul laserului. Circa 2.000 de LED-uri sînt așezate pe o "stinghie" una lîngă alta și pot fi comandate individual.

Indiferent că e laser sau LED, lumina produsă face ca pe tambur, în locul luminat, încărcătura electrostatică să dispară. Există de altfel și imprimante laser care lucrează pe principiul contrar - se încarcă în punctele de incidență ale razei luminoase. Funcție de procedeu utilizat, imprimantele sînt numite "cescriu negru" sau "ce scriu alb".

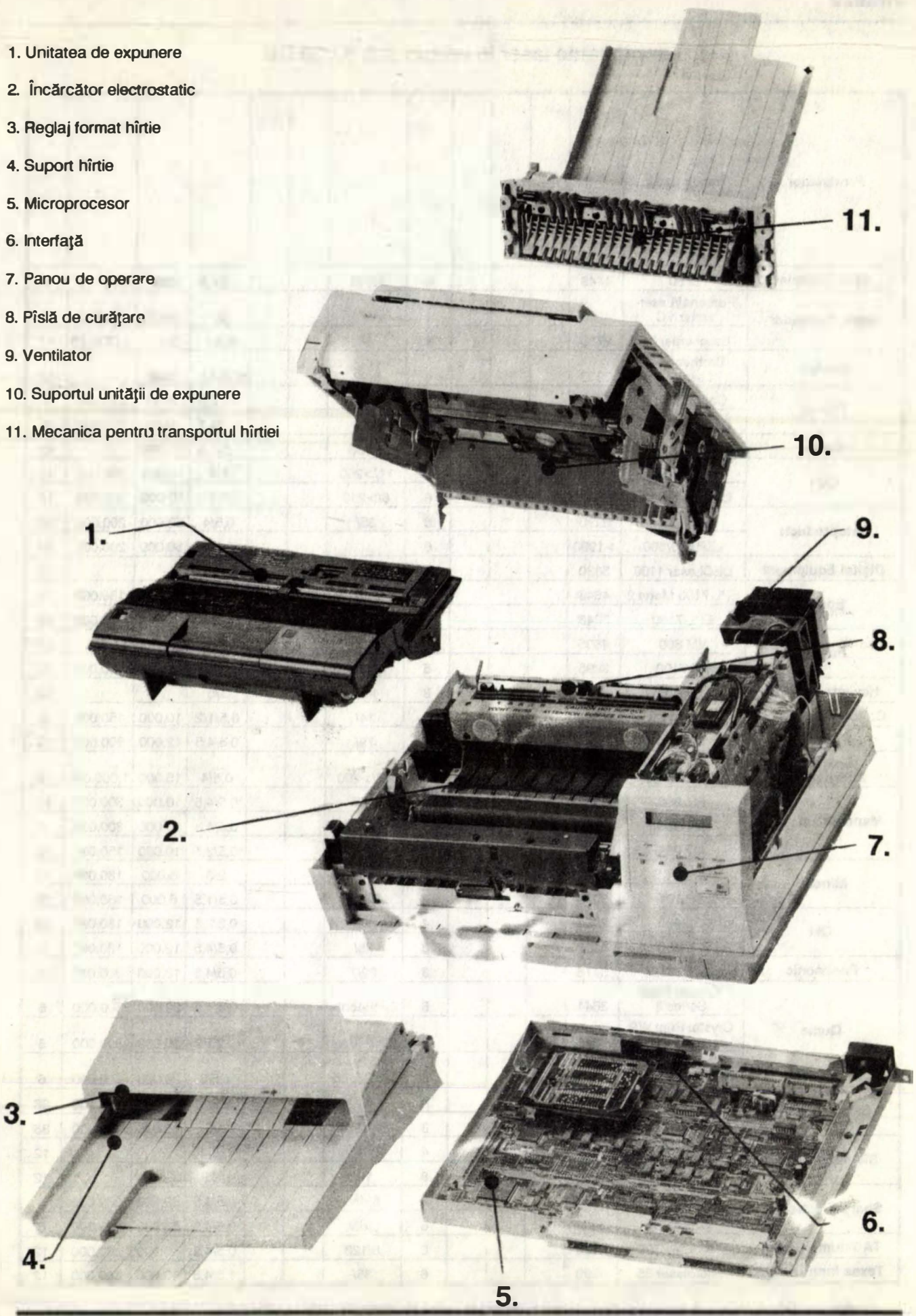
În continuare, tamburul este "prăfuit" cu pulberea extrem de fină de toner; în punctele încărcate, tonerul aderă, astfel că pe tambur apare o primă imagine a ceea ce se va tipări. Pentru aceasta, pulberea uscată de toner se încălzește și se topește. Cu cîteva fracțiuni de secundă mai tîrziu, se face imprimarea pe hîrtie, pe care imaginea rămîne fixată.

Resturile de toner care nu au rămas pe hîrtie sînt măturate de pe tambur și colectate de cîteva mici pensule înainte ca tamburul să fie descărcat. În stația de fixare, tonerul și hîrtia se răcesc, înainte ca o pagină imprimată să iasă, în sfîrșit, din imprimantă.

Principiul de funcționare



- 1. Unitatea de expunere
- 2. Încărcător electrostatic
- 3. Reglaj format hîrtie
- 4. Suport hîrtie
- 5. Microprocesor
- 6. Interfață
- 7. Panou de operare
- 8. Pîslă de curățare
- 9. Ventilator
- 10. Suportul unității de expunere
- 11. Mecanica pentru transportul hîrtiei



Imprimante laser la prețuri sub 5.000 DM

Producător	Imprimantă											
AEG Olympia	LP60	4446			6	49/10			0,5/4,5	6000	180.000	6
Apple Computer	Personal/Laserwriter SC	4350			4	4/			2/	3000	150.000	12
	Laserwriter LS	2730			4	4/			0,5/1	3000	>150.000	12
Brother	Brother HL-4 Laser AS	3933			4	5/22			0,5/4,5	3500		12
Canon	Canon LBP-4	3450			4	4/19			0,5/2,5	3500	200.000	6
	Canon LBP-8III	4950			8	4/19			1,5/4,5	4500	200.000	6
Casio	LCS-1600	1190			6	13/60			1,5/1,5	10.000		12
COT	Laserpro 90/8	4902			8	11/>200			1/5	10.000	300.000	12
	Laserpro EXEX	3409			6	6/>200			0,5/4,5	10.000	300.000	12
Dataproducts	LZR 650	2290			6	35/			0,5/4	50.000	200.000	24
	LZR 650/660	>1990			6				0,5/4	50.000	200.000	24
Digital Equipment	DECLaser 1100	3830			4	36/			0,5/2,5			12
Epson	EPL-7100 Mega 2	4648			6	11/13			2/6	6.000	180.000	12
	EPL-7100	3948			6	11/13			0,5/6	6.000	180.000	12
Fujitsu	VM 800	4695			8	7/8			1/5	8.000		12
	RX 7100	3995			5	6/30			0,64/4	6.000	180.000	12
Hewlett-Packard	HP Laserjet III	4400			8	14/8			1/5			12
C. Itoh Electronics	CI-4	3306			4	14/			0,5/1/2	10.000	150.000	9
Kyocera	F 800 T	4218			8	79/			0,5/4,5	12.000	300.000	12
Lexmark International	IBM 4019-E01	4611			5	10/>200			0,5/4	15.000	1.000.000	12
Mannesman Tally	MT 906	4503			6	6/35			0,5/4,5	10.000	300.000	12
	MT 905	4503			6	6/div.			0,5/4,5	10.000	300.000	12
	MT 904	3352			4	14/35			0,5/2,5	10.000	150.000	12
Minolta	SP 101 S	4891			6	48/			2/3	6.000	180.000	12
	SP 101	3979			6	32/44			0,5/1,5	6.000	180.000	12
Oki	OL 400	2998			4	15/			0,5/2,5	12.000	180.000	12
	OL 800	4198			8	25/			0,5/4,5	12.000	180.000	12
Panasonic	KX-P 4420	3418			8	22/7			0,5/4,5	18.000	300.000	12
Qume	Crystal Print Series II	3644			6	2/HP-fonts			0,5/1,5	20.000	300.000	6
	Crystal Print WP Plus	2845			6	2/HP-fonts			0,25/6	20.000	300.000	6
	Crystal Print Super Series II	4555			6	7/HP-fonts			1,5/6	20.000	300.000	6
Sanyo Electric	SPX 508	3180			8	9/128			0,5/5	15.000	300.000	36
	SPX 608	3408			8	9/128			1/5	15.000	400.000	36
Siemens Nixdorf	9021	3400			4	14/			0,5/4			12
	4810 P	4900			8	14/			1/5			12
Sharp Electronics	JX-9500H	3933			9	6/div.			0,5/4,5	50.000	200.000	6
	JX-9500 E	3408			6	6/div.			0,5/4,5	50.000	200.000	6
TA Triumph-Adler	TA SDR 7706	4320			6	8/128			0,5/4,5	10.000	300.000	12
Texas Instruments	Microlaser 35	4990			6	35/			1,5/4,5	40.000	300.000	12

Imprimanta HP Laserjet IIIP

Noua imprimantă Laserjet IIIP ce va înlocui imprimantele de tip HP Laserjet IIP, îmbină tehnologia seriei III cu prețul avantajos al predecesoarelor sale.

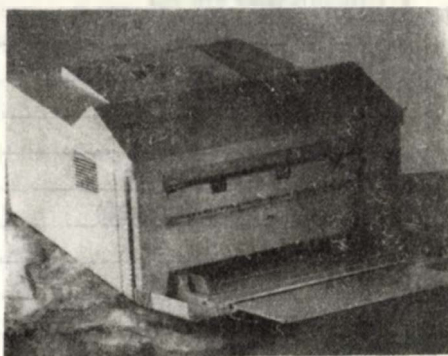
Cu Laserjet IIISi, Hewlett-Packard și-a delimitat poziția în segmentul superior al seriei III. Cu noul Laserjet IIIP, producătorul își rotunjește seria și în jos. La prețul de vânzare de 3500 DM, Hewlett-Packard dorește să-și accentueze poziția sa de lider al pieții și în segmentul prețurilor coborâte.

Cu excepția efigiei, noul Laserjet IIIP nu se deosebește cu nimic, ca aspect exterior, de predecesorul lui IIP. Toate detaliile carcasei au rămas aceleași. Și panoul operator - care în practică s-a dovedit reușit - a fost preluat întocmai.

Ocupînd o suprafață de 34x63 cm (inclusiv caseta de hîrtie), Laserjet IIIP are loc aproape oriunde. Înălțimea este de 21 cm; cu caseta, opțională, de hîrtie, ce poate fi amplasată sub imprimantă, se mai adaugă 6 cm.

În mod normal, alimentarea cu hîrtie se face din față, folosind o casetă de alimentare multifuncțională, ce are o capacitate de 70 de pagini și se potrivește și pentru pluri. Deși Laserjet IIIP nu este concepută ca o imprimantă de mare performanță, ci ca o imprimantă cu volum mediu de tipărire, rezerva de 70 de pagini este mai degrabă insuficientă. Caseta opțională de hîrtie, ce poate fi amplasată sub imprimantă, rezolvă două probleme deodată. Pe de o parte nu mai este necesară caseta obișnuită de alimentare cu hîrtie, suprafața ocupată reducîndu-se astfel la 34x40,5 cm. Pe de altă parte rezerva de hîrtie crește la 250 de pagini. Mecanica pretențioasă utilizată face ca această casetă opțională să fie suficient de scumpă: 400 DM.

Multe detalii constructive evidențiază faptul că IIIP este un pro-



duș matur, bine conceput. Astfel, de ex., mai multe cîrlige mici de ridicare, amplasate pe partea superioară a dispozitivului de antrenare a hîrtiei, împiedică paginile care ies din imprimantă să mai antreneze în mișcare paginile deja tipărite.

Instalarea casetei de toner, care conține toate materialele consumabile, se face prin față. Caseta este surprinzător de mică și poate fi montată dintr-o singură mișcare. Ea este suficientă pentru cca. 3500 de pagini și costă cca. 240 DM. Astfel o pagină ajunge la un cost de 6,9 pfennigi (fără costul echipamentului), respectiv 7,6 pfennigi /pagină (inclusiv echipamentul); ceea ce face ca IIIP-ul să se situeze în domeniul cel mai coborît de prețuri din categoria imprimantelor ce nu sînt capabile de Postscript.

Legarea imprimantei se poate face fie serial, fie paralel. Ambele interfețe fac parte din configurația standard. Tot în configurația standard - lucru neobișnuit la imprimante din această categorie de prețuri - imprimanta este echipată cu 1 MByte de memorie; cu module corespunzătoare, memoria poate fi extinsă pînă la 4 MByte.

Imprimanta Laserjet IIIP este complet compatibilă cu Laserjet III și, ca toate imprimantele seriei III, dispune de noul limbaj de comandă pentru imprimante PCL 5.

Sprijinită de mai toate programele cunoscute (în mod special de

Windows 3.0, cu toate aplicațiile), imprimanta poate fi folosită la parametrii maximi fără drive-re suplimentare.

Cele mai importante noutăți la PCL 5, față de PCL 4, sînt fonturile scalabile, grafica vectorială, direcția de imprimare, modelul de imprimare și îndesirea rastrului.

În afara celor 14 fonturi fixe, IIIP oferă suplimentar 8 fonturi ce pot fi dimensionate, bazate pe "Intellifont Scaling Technology" de la Agfa Compugraphic. Pe baza conturului, aceste fonturi sînt convertite la dimensiuni între 0,25 și 999,75 puncte. Cînd un program solicită o anumită dimensiune, aceasta este calculată și pregătită în memoria imprimantei.

Grafica vectorială integrată în PCL 5 este limbajul grafic propriu Hewlett-Packard, HPGL/2. Prin intermediul acestuia IIIP poate fi adresat și ca plotter. Funcțiile de grafică vectorială pot fi combinate cu celelalte funcții din PCL 5.

Astfel, fonturile scalabile pot fi integrate într-o grafică vectorială și pot fi imprimate într-o direcție oarecare. Suplimentar, sînt disponibile efecte speciale, cum ar fi fonturi înclinate sau lărgite.

Dacă pînă acum totdeauna era problematic să reușești scrieri în direcții distincte pe aceeași pagină, acum acest lucru este ușor realizabil cu PCL 5. Orice text poate fi imprimat în patru direcții, rotite cu cîte 90 grade.

Prin funcția de "model de imprimare" se înțelege, de ex., posibilitatea ca obiectele să fie "umplute" cu tonuri de gri sau modele diferite și de a suprapune imagini diferențiat. Astfel, Laserjet IIIP dispune de tonuri de gri rezidente și de modele de rastu, care duc la imprimări uniforme. În afară de aceasta, folosirea lor duce la o mărire a vitezei de lucru, deoarece imprimanta produce rastul intern,

acesta nemitrebuind "adus" bit cu bit prin interfață de la calculator. Prin posibilitatea suplimentară de a suprapune diverse elemente transparent sau opac, se pot obține efecte interesante, ca scrieri umbrite sau reprezentări inverse.

La imagini scanate, de ex., își găsește aplicație "îndesirea de rastru". Este vorba despre un procedeu de compresie care, aplicat fișierelor-imagini, economisește spațiu pe harddisk. În afară de aceasta, se reduce timpul necesar transferului lor din calculator spre imprimantă.

Un alt avantaj al întregii familii Laserjet III este procedeul RET (Resolution Enhancement Technology), care duce la o calitate vizibil îmbunătățită a imprimării. Un cip special, propriu HP, selectează și prelucurează în timp real pachete de 6x2560 puncte, pentru a corecta, de ex., liniile oblice, curbele sau întretăierile de linii. Deși rezoluția la Laserjet III este de doar 300 dpi (dot per inch), această prelucrare suplimentară duce la o imprimare vizibil mai bună, variind mărimea și poziția punctelor. Printr-o modulare a razei laser, dimensiunea punctului poate fi variată în 5 trepte. Chiar la liniile înclinate, binecunoscutul "efect treaptă" poate fi eliminat aproape complet. La întretăierile de linii, punctele de rastru mai mici evită obișnuitele îngroșări și apariția "bulgărilor" de toner.

Intensitatea procedurii RET poate fi variată în 3 trepte; sau poate fi invalidată complet. Un rastu special pe o imprimare de test face posibilă găsirea variantei optime.

Îmbunătățirile se văd cel mai bine la compararea cu imprimarea fără RET. Avantajul este vizibil și fără lupă și "aruncă" Laserjet III-ul mult în față în ceea ce privește calitatea imprimării, la nivelul celorlalte reprezentante ale seriei III.

În ceea ce privește viteza de imprimare, III marchează progrese considerabile față de modelul II. Procesorul intern (un Motorola 68000) lucrează acum la 16 MHz

Caracteristici	
Nume	HP Laserjet III
Pret (inclusiv MwSt.)	cca. 3500 DM
Carcasa	
Dimensiuni (L x l x I în cm)	34 x 21 x 63
Date tehnice	
Tehnica imprimării	Laser
Unitatea de imprimare	Canon
Rezoluția (text și grafică)	300 x 300 dpi
Tambur	organic
CPU	Motorola 68000
Memoria (min./max.)	1 MByte / 5 MByte
Timpul de încălzire la pornire	45 sec.
Listări într-o lună pînă la	8000 pagini
Durata de viață	nelimitată
Interfețe	
Paralelă	da
Serială	da
Apple Talk	optional
Altele	RS242 opțional
Firmware	
Familii de fonturi	Courier, Line Printer, Times, Univers
Număr de fonturi	67
Emulări	HP Laserjet III
Opțiuni hirtie	
Formate	DIN A4, C5
Greutatea (g/m ²)	60 pînă la 105
Rezerva	70 / 320 pagini
Antrenare	manuală / automată
leșire (face-down / face-up)	70 / 20 pagini
Materiale consumabile	
Toner	-
Tambur	-
Unitatea de încălzire	-
Cartuș toner/tambur	3500 pag., cca. 240 DM
Costuri	
Preț pe pagină (cu / fără prețul aparatului)	7,61 / 6,87 pfenning
Altele	
Garanție	12 luni
Accesorii	Program hardcopy
Manuale	în engleză

(10 MHz la modelul II), iar viteza de transfer a crescut de 4 ori.

Și limbajul de comandă al imprimantei a devenit mai rapid, lucru vizibil mai ales la grafică.

Fără îndoială, cu Laserjet III, Hewlett-Packard confirmă poziția

sa de lider pe piața imprimantelor laser, echilibrînd acum și sub aspectul prețului situația care avantaja, uneori, concurența.

(I.F.)

Specificatia Lotus/Intel/Microsoft/Ast (XMS) pentru memoria extinsa

Specificatiile L/I/M/A pentru memoria extinsa (numite si specificațiile XMS) cuprind, printre altele, unele tehnici de gestiune a memoriei al căror scop principal este acela de a furniza un standard pentru diverși producatori de software care intentionează utilizarea memoriei extinse. Aceasta, la prima ei apariție - pe PC AT - nu era însoțită de nici o metodă care să permită partajarea ei de mai multe aplicații; această regretabilă absență a dus la apariția mai multor metode de gestiune incompatibile între ele, precum desuetul procedeu VDISK.

Chiar dacă sînt prezente într-un fișier care se încarcă ca un "device driver" normal, funcțiile XMS nu sînt apelate prin metodele tipice driver-elor DOS, ci invocînd Int 2fH cu coduri de funcție specifice; datorită acestui fapt "driver"-ul XMS este denumit manager de memorie extinsă (eXtended Memory Manager).

Datorită unei scăpări în proiectarea microprocesorului 80286 există un mod REAL de adresare directă a unei părți din memoria aflată deasupra primului megabyte, care teoretic ar trebui să fie inaccesibilă. După cum este știut, adresele utilizate de 8086 sînt codificate pe 20 biți, fiind compuse dintr-un segment și un deplasament (offset). În exemplul următor acestea se combină astfel:

Offset		9	9	9	9	+
Segment		F	E	D	C	=
Adresa	0	8	7	5	9	

După cum se observă 8086 ignoră transportul pe bitul 20, rezultînd astfel o referință la adresa 08759H. 80286 folosește adrese pe 24 biți și nu ignoră transportul obținînd o referință la adresa 108759H, deasupra primului megabyte.

Pentru a obține compatibilitatea cu programe care au fost scrise pentru 8086 (și care în unele cazuri se bazează în mod explicit pe adresarea cu pierdere de transport), IBM a realizat AT-ul cu linia A20 a magistralei de adrese permanent dezactivată, ignorînd astfel transportul generat de 80286. Linia poate fi totuși activată la cererea programelor aplicative, care în acest caz pot adresa direct primii 64 KB ai memoriei extinse punînd în segment valoarea FFFFH; valorile acoperite pornesc de la FFFFH:0010H la FFFFH:FFFFH, adică 64 KB - 16 bytes de memorie extinsă.

Termeni noi introduși de specificațiile XMS - L/I/M/A:

- **Memorie superioară** (Upper Memory) - cu acest termen se desemnează zona de memorie rezervată pentru a avea acces la alte tipuri de memorie prin intermediul unor driver-e și a unei serii de adrese altfel neutilizate.
- **Memorie înaltă** (HMA - High Memory Area) - memorie deasupra primului megabyte adresabilă direct cu microprocesorul 80286 în mod real; nu poate fi mai lungă de circa 64 KB .

Pentru a permite accesul la memoria înaltă, ea trebuie disponibilizată prin încărcarea driver-ului XMS precizat în CONFIG.SYS. Avantaje:

- - utilizatorul poate decide dacă să instaleze sau nu funcțiile XMS și dacă să permită sau nu utilizarea memoriei înalte;
- - utilizatorul poate asigura instalarea funcțiilor înaintea oricărui program care le-ar putea apela;
- - producătorii pot realiza "device driver"-e diferite.

Specificațiile XMS - L/I/M/A au fost furnizate pentru prima dată de Microsoft printr-un driver numit HIMEM.SYS .



Sumarul funcțiilor XMS

Funcțiile XMS se pot obține apelînd o rutină a cărei adresă este furnizată în urma unui apel la o funcție INT 2Fh, după cum se va ilustra mai departe. Transmiterea parametrilor, rezultatelor se face prin intermediul regiștrilor, specific pentru fiecare funcție în parte.

00h - Citește numărul versiunii XMS

Restituie numărul de versiune al managerului de memorie extinsă XMS.

Input		Output	
AH	00H	AX	Numărul vers. BCD 0123H=1.23
		BX	Versiune internă XMS
		DX	1=Există HMA

01h - Cere acces la HMA

Cere accesul la memoria înaltă (HMA) și o alocă dacă aceasta e disponibilă și dacă dimensiunea cerută este mai mare sau egală cu parametrul /HMAMIN (din linia de comandă device = himem.sys /HMAMIN ...).

Input		Output	
AH	01H	AX	1=HMA alocată
DX	Dimensiunea cerută FFFFH pentru aplicative liberă pentru TSR	BL	cod de eroare dacă AX <> 1

02h - Eliberează HMA

Eliberează memoria înaltă (HMA) ; această funcție trebuie apelată de aplicativele care se folosesc de HMA, înainte de terminarea execuției.

Input		Output	
AH	02H	AX	1=HMA eliberată
		BX	cod eroare dacă AX <> 1

03h - Activează A20 global

Activează linia A20 a magistralei de adrese pentru a permite accesul la memoria înaltă (HMA), care trebuie să fi fost deja alocată ; pe unele AT-uri mai vechi această funcție poate necesita un timp considerabil de lung pentru a fi executată.

Input		Output	
AH	03H	AX	1=A20 activată
		BL	cod eroare dacă AX <> 1

04h - Dezactivează A20 global

Dezactivează linia A20 a magistralei de adrese re-aducînd accesul la modul 8086, în care adresa FFFFh:0010h coincide cu 0000h:0000h; trebuie să fie apelată de programe care au controlul HMA și au activat linia A20, înainte de terminarea execuției; execuția ei poate necesita un timp considerabil.

Input		Output	
AH	04H	AX	1=A20 dezactivată
		BL	cod eroare dacă AX <> 1

05h - Activează A20 local

Activează linia A20 a magistralei de date pentru a permite accesul la memoria extinsă; execuția ei poate necesita un timp considerabil.

Input		Output	
AH	05H	AX	1=A20 activată
		BL	cod eroare dacă AX <> 1

06h - Dezactivează A20 local

Dezactivează linia A20 a magistralei de date, re-aducînd accesul la memorie la modul 8086, în care adresa FFFFh:0010h coincide cu 0000h:0000h; această funcție trebuie să fie apelată de programe care să fi activat anterior linia A20 înainte de terminarea lor; execuția ei poate necesita un timp considerabil.

Input		Output	
AH	06H	AX	1=A20 dezactivată
		BL	cod eroare dacă AX <> 1

07h - Cere starea liniei A20

Verifică dacă linia A20 este corect activată sau dezactivată.

Input		Output	
AH	07H	AX	1=A20 activată

08h - Cere dimensiunea memoriei extinse libere

Returnează dimensiunea celui mai mare bloc de memorie extinsă disponibil precum și memoria extinsă liberă totală, excluzând memoria înaltă dacă aceasta a fost deja alocată.

Input		Output	
AH	08H	AX	Dimensiune bloc în KB
		DX	Memorie liberă totală în KB
		BL	Cod eroare

09h - Alocă un bloc de memorie extinsă

Alocă un bloc de memorie extinsă și restituie identificatorul unui handle XMS, dacă există memorie disponibilă suficientă; de notat că numărul disponibil de handle-uri XMS este limitat, acestea trebuind cerute cu moderație.

Input		Output	
AH	09H	AX	1 = Bloc alocat
DX	Dimensiunea cerută în KB	DX	Handle XMS
		BL	Cod eroare dacă AX < > 1

0Ah - Eliberează un bloc de memorie extinsă

Eliberează un bloc de memorie extinsă alocat anterior; această funcție trebuie apelată înainte de terminarea programului, pentru a evita ca blocul respectiv de memorie să rămână permanent alocat și inutilizabil de către alte aplicații.

Input		Output	
AH	0AH	AX	1 = Bloc eliberat
DX	Handle XMS alocat anterior	BL	Cod eroare dacă AX < > 1

0Bh - Mută un bloc de memorie extinsă

Deplasează un bloc de memorie extinsă între două zone de memorie extinsă sau convențională; linia A20 nu trebuie să fie activată în timpul acestui apel dacă memoria înaltă este implicată în deplasare; pentru a indica o adresă de memorie convențională, fie sursă fie destinație se atribuie valoarea zero handle-ului iar adresa în forma segment:offset se atribuie deplasamentului.

Input		Output	
AH	0BH	AX	1 = Bloc mutat
DS:SI	Adresa buffer parametri	BL	Cod eroare dacă AX < > 1

formatul buffer-ului cu parametri de transfer:

DD	lungimea ce urmează a fi transferată, în bytes
DW	handle-ul blocului sursă
DD	deplasament în interiorul blocului sursă, în bytes
DW	handle-ul blocului destinație
DD	deplasament în interiorul blocului destinație, în bytes

0Ch - Protejează un bloc de memorie extinsă

Împiedică deplasările în cadrul blocului și returnează adresa liniară de 32 biți a unui bloc anterior alocat; această funcție este utilă să fie apelată înainte de comutarea în modul protejat de adresare al procesorului.

Input		Output	
AH	0CH	AX	1 = succes
DX	Handle XMS anterior alocat	DX:BX	Adresă liniară de 32 biți
		BL	Cod eroare dacă AX < > 1

0Dh - Deblochează un bloc de memorie extinsă

Deblochează un bloc de memorie anterior protejat; această funcție trebuie utilizată înainte de terminarea programelor care au blocat porțiuni de memorie extinsă.

Input		Output	
AH	0DH	AX	1 = succes
DX	Handle XMS anterior alocat	BL	Cod eroare dacă AX < > 1

0Eh - Citește informații despre un handle XMS

Furnizează informații relativ la un bloc anterior alocat de memorie extinsă.

Input		Output	
AH	0EH	AX	1 = succes
DX	Handle XMS anterior alocat	BH	1 = bloc de memorie protejată
		BL	Număr handle-le disponibile sau Cod eroare dacă AX < > 1
		DX	Lungime bloc în KB

0Fh - Realocă un bloc de memorie extinsă

Modifică dimensiunea unui bloc de memorie extinsă neprotejată.

Input		Output	
AH	0FH	AX	1 = bloc realocat
DX	Handle XMS	BL	Cod eroare dacă AX <> 1
BX	Noua dimensiune în KB.		

10h - Cere un bloc de memorie superioară

Cere accesul la un bloc de memorie superioară aliniat la o adresă de paragraf (segment)

Input		Output	
AH	10H	AX	1 = bloc alocat
DX	Dimensiune în paragrafe	BX	Segment de început al blocului (de 16 biți) alocat
		BL	Cod eroare dacă AX <> 1
		DX	Lungimea în paragrafe a blocului alocat

11h - Eliberează un bloc de memorie superioară

Eliberează accesul la un bloc de memorie superioară; această funcție trebuie apelată înainte de terminarea programelor care au alocat blocuri de memorie superioară.

Input		Output	
AH	11H	AX	1 = bloc eliberat
DX	Segment de început al blocului alocat	BL	Cod eroare dacă AX <> 1

Verificarea prezenței driver-ului XMS și apelarea lui

Înainte ca o aplicație să apeleze funcții XMS trebuie obligatoriu să verifice prezența driver-ului în memorie. Acesta se poate face conform următoarei scheme:

- - Se utilizează INT 2Fh pentru a verifica prezența driver-ului XMS apelând funcția AX = 4300, care trebuie să returneze AI = 80h în cazul în care driver-ul a fost instalat corect iar accesul la memoria extinsă funcționează corespunzător;
- - Se utilizează INT 2Fh pentru a obține adresa de intrare în driver-ul XMS, apelând funcția AX = 4310h; adresa este returnată în perechea de registre ES:BX
- - Driver-ul XMS se apelează prin transferul execuției la adresa obținută prin tehnica anterioară, după ce în prealabil au fost pregătite registrele pentru transmiterea de parametri.

Sumarul codurilor de eroare

În continuare este prezentat tabelul tuturor codurilor de eroare XMS posibile, furnizate de funcțiile apelate totdeauna în registrul BL când AX = 0.

Cod	Descriere
00h	Nu este eroare ;
80h	Funcție neimplementată ;
81h	Dispozitiv asemănător VDISK instalat ;
82h	Eroare în timpul tratării liniei A20 ;
8Eh	Eroare generică de driver ;
8Fh	Eroare fatală de driver ;
90h	Nu există HMA ;
91h	HMA deja alocată ;
92h	Dimensiune cerută inferioară parametrului /HMAMIN
93h	HMA nealocată
A0h	Nu există memorie extinsă liberă
A1h	Handle-le XMS epuizate
A2h	Handle XMS eronat
A3h	Handle XMS sursă eronat
A4h	Deplasament sursă eronat
A5h	Handle XMS destinație eronat
A6h	Deplasament destinație necorespunzător
A7h	Lungime incorectă
A8h	Deplasament cu depășire
A9h	Eroare de paritate
AAh	Bloc neprotejat
ABh	Bloc protejat
ACH	Depășire număr de blocuri permise
ADh	Protejară imposibilă
B0h	Memorie superioară insuficientă
B1h	Nu există memorie superioară disponibilă.
B2h	Segment eronat

Considerații pentru dezvoltarea de aplicații

Considerațiile următoare ar trebui utilizate în proiectarea oricărei aplicații care face apel la interfața XMS:

- - transferurile de date în DMA nu sînt recomandate în memoria înaltă;
- - nu se recomandă transferul ca argumente pentru INT 21h, INT 25h sau INT26h al adreselor din memoria înaltă;
- - se recomandă ca vectorii de întrerupere să nu se direcționeze în memoria înaltă, întrucît dacă linia A20 s-ar dezactiva ulterior, vectorii respectivi n-ar mai putea fi atinși și gestiunea întreruperilor s-ar pierde.

Pentru nefericiții prizonieri ai modului real de adresare prezentăm în cele ce urmează o bibliotecă completă de funcții XMS pentru Turbo Pascal 6.0., care ar putea fi un colac de salvare pînă la apariția și impunerea unui standard DPMI (Interfață cu modul protejat de adresare).

Biblioteca a fost testată cu Turbo Pascal 6.0 și driver-ul XMS aparținînd lui Microsoft - HIMEM.SYS ver 2.0 (care s-a dovedit a nu suporta toate funcțiile specificației XMS: lipsesc funcțiile de acces la memoria superioară), pe următoarele configurații : 386 cu ISA și 8 MByte RAM și respectiv AT 286 cu 1 MByte RAM.

(ing. Flaidăr Alin , ing. Udrea Alin)

```
{ $B-,D-,F-,L+,R-,S-,V- }
unit XMS;

{ Micro ATCI, mai 1991 }
interface

var XMSError: byte;

function XMSInstalled: boolean;

{ the following routines should be called only after a
  successful call to XMSInstalled
}
procedure XMSInfo( var Version, InternalVersion: word;
  var HMAavail: boolean);
  { major version in lowbyte, minor in high byte }
procedure AllocHMA( bytes: word );
procedure FreeHMA;

procedure EnableAddrA20;
procedure DisableAddrA20;
procedure EnableDataA20;
procedure DisableDataA20;
function A20Status: boolean;

procedure AllocExtMem( KBytes: word;
  var Handle: word );
procedure FreeExtMem( Handle: word );
procedure FreeExtMemInfo( var MaxFreeBlock,
  TotalFreeMem { in KBytes } : word );
procedure ReAllocExtMem( KBytes, handle: word );

procedure MoveExtMem(DestOffs
  DestHandle : word;
  SourceOffs : longint;
  SourceHandle : word;
  bytes      : longint);

procedure ReadExtMem( bytes : longint;
  SourceHandle : word;
  SourceOffs : longint;
  var Dest);

procedure WriteExtMem( var Src;
  bytes : longint;
  DestHandle : word;
  DestOffs : longint);

procedure GetHandleInfo( handle
  var locked : boolean;
  var HLength { KBytes },
  FreeHandles: word);

function GetProtectedMemory( handle: word ): pointer; { 32 bits
  linear pointer }
procedure ReleaseProtectedMemory( handle: word );

procedure AllocSupMem( PagesRequested { 16 bytes paragraphs }
:word;
  var PagesAllocated { 16 bytes paragraphs}:word;
  var BaseAddress :pointer );
procedure FreeSupMem( BaseAddress: pointer );

implementation
uses Dos;

var XMSAddr : pointer;
  _AX, _BX, _DX, _SI : word ;
  var buff : record
  TransferBytes : longint;
  source : word;
  SourceOffset : longint;
  destination : word;
```

```
  DestOffset : longint
end;

function XMSInstalled : boolean;

var regs: registers;
begin
  with regs do:
  begin
    AX := $4300;
    intr($2F, regs);
    if AL $80 then XMSInstalled := false
    else
    begin
      AX := $4310;
      intr($2F, regs);
      XMSAddr := ptr( ES,BX );
      XMSInstalled := true
    end
  end
end; { XMSInstalled }

procedure CallXMS;
begin
  asm
  mov AX,[_AX]
  mov BX,[_BX]
  mov DX,[_DX]
  mov SI,[_SI]
  call [XMSAddr]
  mov [_AX],AX
  mov [_BX],BX
  mov [_DX],DX
  end : longint;
end; { call XMS }

procedure XMSInfo( var Version, InternalVersion: word;
  var HMAavail: boolean);
  { major version in low byte, minor in high byte }
begin
  _AX := 0;
  CallXMS;
  Version := _AX;
  InternalVersion := _BX;
  HMAavail := _DX = 1
end;

{
begin
  : word;
  asm
  mov AH,0
  call [XMSAddr]
  mov word [Version], AX
  mov word [InternalVersion], BX
  end
end;
}

procedure FreeExtMemInfo( var MaxFreeBlock,
  TotalFreeMem { in KBytes } : word );
begin
  _AX := $0800;
  CallXMS;
  XMSError := lo(_BX);
  if XMSError = 0 then
  begin
    MaxFreeBlock := _AX;
    TotalFreeMem := _DX
  end
end; { FreeExtMem }

procedure AllocHMA( bytes: word );
begin
```

Specificațiile XMS pentru memoria extinsă

```
_AX := $0100;
_DX := bytes;
CallXMS;
if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { AllocHMA }
```

```
procedure FreeHMA;
begin
  _AX := $0200;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { FreeHMA }
```

```
procedure EnableAddrA20;
begin
  _AX := $0300;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { EnableAddrA20 }
```

```
procedure DisableAddrA20;
begin
  _AX := $0400;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { DisableAddrA20 }
```

```
procedure EnableDataA20;
begin
  _AX := $0500;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { EnableDataA20 }
```

```
procedure DisableDataA20;
begin
  _AX := $0600;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { DisableDataA20 }
```

```
function A20Status: boolean;
{ true -- enabled; false -- disabled }
begin
  _AX := $0700;
  CallXMS;
  A20Status := _AX = 1
end;
```

```
procedure AllocExtMem( KBytes: word;
  var Handle: word );
begin
  _AX := $0900;
  _DX := KBytes;
  CallXMS;
  if _AX = 1 then
  begin
    XMSError := 0;
    handle := _DX
  end
  else XMSError := lo(_BX)
end; { AllocExtMem }
```

```
procedure FreeExtMem( Handle: word );
begin
  _AX := $0A00;
  _DX := handle;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX);
end;
```

```
procedure MoveExtMem( DestOffs : longint;
  DestHandle : word;
  SourceOffs : longint;
```

```
SourceHandle : word;
bytes : longint );
begin
  move( bytes, Buff, sizeof(Buff) );
  _AX := $0B00;
  _SI := ofs( Buff );
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end;
```

```
procedure ReadExtMem( bytes : longint;
  SourceHandle : word;
  SourceOffs : longint;
  var Dest);
begin
  with Buff do
  begin
    TransferBytes := bytes;
    source := SourceHandle;
    SourceOffset := SourceOffs;
    destination := 0;
    DestOffset := longint( @Dest )
  end;
  _AX := $0B00;
  _SI := ofs( Buff );
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { ReadExtMem }
```

```
procedure WriteExtMem( var Src;
  bytes : longint;
  DestHandle : word;
  DestOffs : longint);
begin
  with Buff do
  begin
    TransferBytes := bytes;
    source := 0;
    SourceOffset := longint( @Src );
    Destination := DestHandle;
    DestOffset := DestOffs
  end;
  _AX := $0B00;
  _SI := ofs( Buff );
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { WriteExtMem }
```

```
function GetProtectedMemory( handle: word ): pointer; { 32 bits
  linear pointer }
```

```
type SegOfs = record
  OfS, Seg: word
end;
var LinPtr: pointer;
```

```
begin
  _AX := $0C00;
  _DX := handle;
  CallXMS;
  if _AX = 1 then
  begin
    SegOfs(LinPtr).seg := _DX;
    SegOfs(LinPtr).ofs := _BX;
    XMSError := 0;
  end
  else
  begin
    LinPtr := nil;
    XMSError := lo(_BX)
  end;
  GetProtectedMemory := LinPtr
end;
```

```

procedure ReleaseProtectedMemory( handle: word );
begin
  _AX := $0D00;
  _DX := handle;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { ReleaseProtectedMemory }

procedure GetHandleInfo( handle
  var locked : boolean;
  var HLength { KBytes },
  FreeHandles: word);
begin
  _AX := $0E00;
  _DX := handle;
  CallXMS;
  if _AX = 1 then
  begin
    locked := hi(_BX) = 1;
    FreeHandles := lo(_BX);
    HLength := _DX;
    XMSError := 0
  end
  else XMSError := lo(_BX)
end; { HandleInfo }

procedure ReAllocExtMem( KBytes, handle: word );
begin
  _AX := $0F00;
  _BX := KBytes;

```

```

  _DX := handle;
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { ReAllocExtMem }

procedure AllocSupMem( PagesRequested { 16 bytes paragraphs }
:word;
  var PagesAllocated { 16 bytes paragraphs }:word;
  var BaseAddress : pointer );
begin
  _AX := $1000;
  _DX := PagesRequested;
  CallXMS;
  if _AX = 1 then
  begin
    PagesAllocated := _DX;
    BaseAddress := ptr(_BX, 0);
    XMSError := 0
  end
  else XMSError := lo(_BX)
end; { AllocSupMem }

procedure FreeSupMem( BaseAddress: pointer );
begin
  _AX := $1100;
  _DX := seg(BaseAddress ^);
  CallXMS;
  if _AX = 1 then XMSError := 0 else XMSError := lo(_BX)
end; { FreeSupMem }
end.

```

{ \$A +, B-, D-, F-, G +, O-, R-, S-, V-, X- }
 { \$M 16384, 0, 655360 }

```

program TestXMS;
{ Micro ATCI, mai 1991 }

uses XMS, crt;
type MaxArray = array[1..65519] of byte;

var
  Ver, IntVer : word;
  HMAOk : boolean;

  MaxFree,
  MemFree : word;
  GctA20Status: boolean;
  { top of conventional memory: }
  HMA TestArray: MaxArray absolute $FFFF:$0010;

  handle : word;
  locked : boolean;
  HandleLength,
  FreeHandles : word;

  i : word;
  j, N : longint;
  T : array[0..255] of longint; { 1 KB long }

  pages : word;
  BaseAddress : pointer;
  type ST2 = string[2];

  function HexString(Number: byte): ST2;
  function Hex_Char(Number: Word): Char;
begin
  if Number < 10 then
    Hex_Char := Char(Number + 48)
  else
    Hex_Char := Char(Number + 55);
end; { Function Hex_Char }

```

```

  Var
  S: ST2;
begin
  S := '';
  S := S + Hex_Char(Number shr 4);
  Number := Number and $0F;
  S := S + Hex_Char(Number);
  HexString := S
end; { Function Hex_String }

procedure Error;
var msg: string[40];
begin
  write('Error ' + HexString(XMSError));
  case XMSError of
    0 : msg := 'No error.';
    $80 : msg := 'Inexisting function.';
    $81 : msg := 'VDISK like device present.';
    $82 : msg := 'A20 handling error.';
    $8E : msg := 'Internal error.';
    $8F : msg := 'Fatal internal error';
    $90 : msg := 'No HMA.';
    $91 : msg := 'HMA already allocated.';
    $92 : msg := 'Cannot allocate less than /HMAMIN.';
    $93 : msg := 'HMA not allocated.';
    $A0 : msg := 'Extended memory totally allocated.';
    $A1 : msg := 'No more handles available.';
    $A2 : msg := 'Bad XMS handle.';
    $A3 : msg := 'Bad source handle.';
    $A4 : msg := 'Bad source offset.';
    $A5 : msg := 'Bad destination handle.';
    $A6 : msg := 'Bad destination offset.';
    $A7 : msg := 'Length not allowed.';
    $A8 : msg := 'Bad length with overlapping.';
    $A9 : msg := 'Parity error.';
    $AA : msg := 'Unprotected block.';
    $AB : msg := 'Protected block.';
    $AC : msg := 'Too many blocks.';
    $AD : msg := 'Cannot protect block.';
    $B0 : msg := 'Not enough upper memory.';

```

Specificațiile XMS pentru memoria extinsă

```

$B1: msg := 'Not available upper memory.';
$B2: msg := 'Bad Segment address.';
end;
writeln;
writeln(msg);
if XMSError in [$8E,$8F] then halt(1);
end;

begin
  if XMSInstalled then
    begin
      XMSInfo(Ver,IntVer,HMAOk);
      writeln('Found XMS driver version ',HexString(hi(Ver)),',',
        HexString(lo(Ver)),
        '; internal version ',HexString(hi(IntVer)),',',
        HexString(lo(IntVer)));
      write('High Memory Area ');
      if HMAOk then writeln(' available.') else writeln(' not available');

      FreeExtMemInfo(MaxFree,MemFree);
      writeln('Free extended memory: ',MemFree,' kbytes. ');
      writeln('Maximum free block: ',MaxFree,' kbytes. ');
      GetA20Status := A20Status;
      if GetA20Status then writeln(' A20 line enabled.')
      else writeln('A20 line disabled. ');

      if HMAOk then
        begin
          AllocHMA(50000);
          if XMSError > 0 then Error
          else
            begin
              writeln('HMA successfully allocated. ');
              if not GetA20Status then
                begin
                  write(' Enabling A20 address line. This may take
                    some time...');
                  EnableAddrA20;
                  if XMSError > 0 then Error
                  else
                    begin
                      writeln('OK. ');
                      write(' Enabling A20 data line. This may take
                        some time...');
                      EnableDataA20;
                      if XMSError > 0 then Error
                      else
                        begin
                          writeln('OK. ');
                          write('Testing HMA...');
                          for i := 1 to 65519 do
                            HMAstestArray[i] := i mod 255;
                            i := 1;
                          while (HMAstestArray[i] = i mod 255) and
                            (i <= 65519) do inc(i);
                          if i < 65519 then writeln('HMA Test failed.')
                          else writeln('OK. ');
                          DisableAddrA20;
                          if XMSError > 0 then Error;
                          DisableDataA20;
                          if XMSError > 0 then Error
                        end
                      end
                    end;
                  FreeHMA;
                  if XMSError > 0 then Error
                end
              end;
            end;
          write('Allocating ',MaxFree,' kbytes of extended memory...');
          AllocExtMem(MaxFree,handle);
          if XMSError > 0 then Error
          else
            begin
              writeln('OK. ');
            end
          end;
        end
      end;
    end
  end;
end;

```

```

GetHandleInfo(handle,locked,HandleLength,
  FreeHandles);
if XMSError > 0 then Error
else writeln('Handle ',handle,' allocated; length: ',
  HandleLength,
  '; ',FreeHandles,' more free handles. ');
writeln('Testing ',MaxFree,' kbytes extended memory. ');
write('Writing ',1024,'
N := 0;
for i := 1 to MaxFree do
begin
  gotoxy(9,WhereY); write(N shl 2 + 1024);
  for j := 0 to 255 do T[j] := N + j;
  WriteExtMem(T,1024,handle,N shl 2);
  inc(N,256);
  if XMSError > 0 then
    begin
      Error;
      i := MaxFree + 2
    end
  end;
end;
if i <= succ(MaxFree) then
begin
  Gotoxy(1,WhereY);
  write('Reading ',1024,'
N := 0;
for i := 1 to MaxFree do
begin
  gotoxy(9,WhereY); write(N shl 2 + 1024);
  ReadExtMem(1024,handle,N shl 2, T);
  if XMSError > 0 then
    begin
      Error;
      i := MaxFree + 2
    end
  end;
else
begin
  for j := 0 to 255 do
    if T[j] <> j + N then
      begin
        writeln;
        writeln(' Error testing extended memory. ');
        i := MaxFree + 2
      end
    end;
  inc(N,256);
end;
if i <= succ(MaxFree)
begin
  writeln;
  writeln(' OK. ')
end
end
end;
write('Deallocating ',MaxFree,' KBytes of extended memory... ');
FreeExtMem(handle);
if XMSError > 0 then Error else writeln('OK. ');
end
else writeln('XMS driver not installed. ');
end.

```

Cobol = Cobol ?

Din cel puțin două puncte de vedere, Cobol-ul are caracteristici singulare: pe de o parte, este, incontestabil, cel mai utilizat limbaj de programare în domeniul comercial, iar pe de altă parte, (surprinzător?) este totodată unul din cele mai contestate limbaje de programare.

Faptul că este utilizat intens în ramurile corespunzătoare, ar trebui să fie un argument puternic în favoarea lui. Atunci, de unde motivele de ceartă? Orice programator cu experiență în COBOL cunoaște desigur punctele slabe care pot servi ca și contraargumente pentru folosirea COBOL-ului - le vom trece pe scurt în revistă, încercând să arătăm că mai puțin limbajul ca atare cît alte circumstanțe sînt cauze ale disputei și că avantajele precumpănesc cu mult.

Să începem cu lucruri elementare: manualele de referință sînt toate, în limba engleză. Mai mult: autorii acestora țin la o completitudine deseori exagerată și informații foarte concentrate, ceea ce, deseori, duce la nenumărate trimiteri. Exemple clare, care să demonstreze descrierile stufoase ale sintaxei în mod practic, lipsesc de obicei cu desăvîrșire.

Chiar dacă unele compilatoare COBOL sînt însoțite de complexe programe demonstrative - rămîne de văzut dacă găsiți acele pasaje care sînt interesante pentru munca Dvs. Chiar dacă ați studiat informatica, dacă e să vă obișnuți cu un nou compilator, ar trebui să vedeți o perioadă de experimentări destul de lungă și mult timp "pierdut".

În afară de aceasta, deseori listări kilometrice cu "algoritmi" îmbîcșiți și total nestructurați după criteriile moderne, sperie cititorul. Critica aproape că nu are sens; pentru că acolo unde se programează strict în COBOL, generații elitiste țin cu dinții de tradiții, iar cei care intră sînt nevoiți să se supună

cu strictețe "standardelor" specifice întreprinderii. Pentru programe regîndite și cu deschidere spre viitor - posibile și în Cobol, în ciuda afirmațiilor contrare! - pur și simplu nu rămîne timp.

Așa că se continuă în ritmul obișnuit. Asemănător microprocesoarelor 80386 și 80486, care, în ciuda arhitecturii celei mai inteligente și unor capacități aproape inepuizabile, sînt nevoite să se lupte cu aplicații DOS cu multiple limitări de memorie, și în Cobol, în ciuda faptului că în permanență se înlocuiesc standarde vechi cu unele mai noi și se implementează extensii de limbaj tot mai funcționale, "bătrînele" programe, care, deși ticșite cu GOTO-uri, au acea calitate de bază care pînă la urmă decide valoarea unui program - merg! - rămîn în uz și trebuie întreținute, pe parcursul a ani și ani de zile de muncă migăloasă. Nu-i de mirare dacă un programator răsfățat în C sau în Pascal nu găsește nici o plăcere în Cobol și refuză să-și lase alterat modul de gîndire - structurat - de reguli de mult depășite. N-ajută la nimic vocabularul simplu, foarte asemănător englezei uzuale, și nici modul vizual foarte "curat" în care se prezintă programele Cobol.

În afară de diversele minusuri ce țin de gusturi individuale, un factor important care explică antipatia ce i se arată este faptul că pentru studenții informaticieni, Cobol-ul este materie obligatorie. Relativ la forma și logica Cobol-ului se poate într-adevăr discuta, căci el este realmente un limbaj de programare orientat pe problemă și ar trebui privit ca atare. A programa calcule științifice - în afara domeniului comercial! - sau aplicații grafice în Cobol este aproape imposibil. Funcții trigonometrice sau alte funcții asemănătoare nu există în vocabularul standard - dar este adevărat că de regulă nici nu sînt necesare în aplicații Cobol.

Dar să trecem și la avantaje. Cînd este vorba de prelucrarea rapidă și exactă pînă la cîteva poziții după virgulă a unor fișiere extrem de mari, cu tabele multidimensionale și indexate, Cobol-ul este una din puținele unelte de programare ce garantează succesul. Un motiv important care a dus la apariția Cobol-ului a fost cerința de a putea face cele mai variate selecții în fișiere oarecare. De aceea, Cobol-ul dispune de un sofisticat mecanism de control pentru fișiere și tabele, cu toate variantele de acces imaginabile, inclusiv chei multiple. Cîmpurile de date ale unui program Cobol sînt ierarhizate în trepte, de la 1 la 49. Astfel e posibilă accesarea unor grupe întregi de date, folosind un singur nume. Variabilele și constantele se divid în alfabetice, alfanumerice, numerice și editate numeric și se detaliază pînă la ultimul octet. Numai definiția inteligentă a unei variabile vă poate deja economisi importante eforturi de programare:

```
77 LEI PIC Z,ZZZ,ZZ9.99 VALUE ZERO.
```

Variabila editată numeric "LEI", de nouă poziții, corespunde nivelului 77, adică nici nu are alte date subordonate și nici nu se subordonează vreunei alte grupe de date. La afișarea pe ecran sau la imprimare, zerourile premergătoare din pozițiile marcate prin "Z" vor fi înlocuite cu blancuri. LEI are două poziții zecimale, e formatată în mii și inițializată cu valoarea 0. (Printr-o clauză în program, scrierea americană a numărului poate fi înlocuită cu scrierea europeană, cu virgulă pentru partea zecimală și punct ca separator de mii).

Există și variabile booleene. Atașate nivelului "88", ele pot corespunde în tabele și mai multor condiții - deci unor mulțimi!

Cu puține excepții sintaxa construcțiilor este identică la aproape toate compilatoarele Cobol. Aproape toate compilatoarele se

```

1 | IDENTIFICATION DIVISION.
2 |
3 | program-id. BENCH.
4 | author. umj,tj.
5 |
6 | ENVIRONMENT DIVISION
7 |
8 | CONFIGURATION SECTION.
9 |
10 | source-computer. AT-DR-DOS.
11 | object-computer. AT-DR-DOS.
12 |
13 | INPUT-OUTPUT SECTION.
14 |
15 | FILE-CONTROL.
16 |
17 |     select File
18 |     assign to 'BENCH.DAT'
19 |     file status IOResult
20 |     organization indexed
21 |     record key DataKey
22 |     access dynamic.
23 |
24 | DATA DIVISION.
25 |
26 | FILE SECTION.
27 |
28 | f d File
29 |     label record standard.
30 | 01 DataRecord.
31 |     05 DataKey pic 99.
32 |     05 DataChr pic x occurs 79 times.
33 |
34 | WORKING-STORAGE SECTION.
35 |
36 | 77 IOResult pic xx.
37 | 77 ZPos pic 99.
38 | 77 SPos pic 99.
39 | 77 Tasta pic x value space.
40 | 77 Pi pic s99(9) value 3.141592654.
41 | 77 x pic s99(10).
42 | 77 s pic s99(10) value zero.
43 | 77 i pic 9(5).
44 | 77 f pic 9(18) value 1.
45 |
46 | 01 Proba pic x.
47 |     88 Eroare value '1'.
48 |
49 | 01 Matrice.
50 |     05 Rlesire pic x occurs 79 times.
51 |
52 | 01 Sinus.
53 |     02 Rind occurs 25 times.
54 |     03 Coloana occurs 79 times.
55 |     04 Car pic x.
56 |
57 | PROCEDURE DIVISION.
58 |
59 | begin.
60 |     move spaces to Proba, Sinus.
61 |     move -3.14 to x.
62 |     display 'Putintica rabdare...'.
63 |
64 |     perform Domeniu until x 3.14.
65 |     open output File.
66 |     if IOResult not = zero
67 |         display '1. Eroare in fisier: ', IOResult
68 |         accept Tasta
69 |         move '1' to Proba
70 |     else
71 |         move zero to ZPos
72 |         perform Memorare until ZPos = 25.
73 |     close File.
74 |     if not Eroare
75 |         open input File.
76 |         if IOResult not = zero
77 |             display '2. Eroare in fisier: ', IOResult
78 |             accept Tasta
79 |         else
80 |             move zero to ZPos
81 |             perform Reciteste until ZPos = 25.
82 |         close File.
83 |     end.
84 |     stop run.
85 |
86 | Domeniu.
87 |     move zero to i.
88 |     perform Calcul until i = 10.
89 |     compute SPos rounded = x / Pi * 39 + 40.
90 |     compute ZPos rounded = s * 11 + 13.
91 |     move 'x' to Car (ZPos, SPos).
92 |     add 0.1 to x.
93 |     move 0 to s.
94 |     move 1 to f.
95 |
96 | Calcul.
97 |     compute f = f * (i * 2 + 1).
98 |     compute s = s + ((-1)**i) * (x** (2 * i + 1)) /
99 |
100 | Memorare.
101 |     add 1 to ZPos.
102 |     move ZPos to DataKey.
103 |     move zero to SPos.
104 |     perform Aduna until SPos = 79.
105 |     write DataRecord invalid key
106 |         display 'Eroare de scriere: ', IOResult
107 |         accept Tasta.
108 |
109 | Aduna.
110 |     add 1 to SPos.
111 |     move Car (ZPos, SPos) to DataChr (SPos).
112 |
113 | Reciteste.
114 |     add 1 to ZPos.
115 |     move ZPos to DataKey.
116 |     read File invalid key
117 |         display 'Eroare de citire: ', IOResult
118 |         accept Tasta.
119 |     move zero to SPos.
120 |     perform Afisare until SPos = 79.
121 |     display Matrice.
122 |
123 | Afisare.
124 |     add 1 to SPos.
125 |     mov DataChr (SPos) to Rlesire (SPos).
126 |
127 | *END OF BENCH.COB

```

Figura 1.

bazează pe vocabularul exact definit prin standardul ANSI-85, care este realmente foarte cuprinzător

și acoperă multumitor cele mai importante variante de programare.

Cu toate acestea, la o privire atentă portabilitatea se dovedește limitată. Puteți desigur porta fără probleme un program Cobol din DOS spre Unix sau OS/2, și după poziționarea corespunzătoare a comutatorilor de compilare și a altor opțiuni, obțineți, într-un timp rezonabil, o variantă care să meargă - totuși, probleme de adaptare apar întotdeauna.

Chiar și respectînd standardul pînă în ultimul amănunt, trebuie să vă așteptați la lungi liste de erori la prima compilare.

De cele mai multe ori este însă vorba de erori în lanț, care de regulă dispar dintr-o singură "lovitură" (dacă găsiți rădăcina răului...) Cum am mai menționat, acest lucru se poate deseori rezolva modificînd o opțiune de compilare - sau cu o redefinire a unei instrucțiuni Cobol speciale.

Un exemplu posibil - un programator liber profesionist întreține pentru diverse firme programe în Cobol pe PC-uri. Firma X lucrează numai cu compilatorul Micro Focus, în timp ce firma Y s-a specializat în Realia-Cobol. Acasă preferă să lucreze în MS - Cobol, versiunea 2.1. Ocazional, prin intermediul unui cunoscut, are de lucru cu MS-Cobol, versiunea 3.0. Cînd a învățat, a învățat utilizînd VS-Cobol de la IBM. Programatorul unei alte firme, care luca în RM-Cobol, a renunțat la servicii, drept pentru care probabil vor apare comenzi și în legătură cu acest compilator. Astfel că numai în mediul DOS, programatorul are de a face cu 6 compilatoare Cobol.

Programul BENCH.COB, scris pentru a permite o comparație între diversele compilatoare, calculează sinusul folosind cele 4 operații aritmetice elementare, face apoi două ridicări la putere, scrie rezultatele sub formă de tabel într-un fișier, recitește fișierul și reprezintă pe ecran o curbă stilizată. Nu e vorba desigur de un program deosebit de util, cît de a folosi, instrucțiuni Cobol cît mai variate și de a observa "reacția" diferitelor compilatoare. Singurul aspect tehnic este

viteza de execuție, care s-a cronometrat. În rest, interesează numai portabilitatea programului.

Dificultăți fundamentale nu există.

Privind însă scurtimea programului, multe mici diferențe se însumează neplăcut. Aplicațiile profesionale în Cobol umplu de regulă bibliorafturi întregi, motiv pentru care "micile" diferențe, chiar dacă stau doar în amănunte "banale", pot duce la timpi de transpunere importanți.

Compilarea cu MS-Cobol, v.2.1 duce cu promptitudine la prima eroare. Numele fișierului "BENCH.DAT" trebuie precizat prin "VALUE of FILE-ID" din "FILE SECTION" și nu se poate face ca la aproape toate celelalte compilatoare prin "ASSIGN TO" din paragraful "FILE CONTROL". Ecranul e șters în mod automat. Compilarea se face prin comenzile menționate în tabelul alăturat; timpul de execuție al modului Runtime ISAM nu a fost luat în considerare.

La MS-Cobol 3.0, ecranul nu se mai șterge automat; programul

funcționează însă impecabil. La fel și la Microfocus și Realia Cobol. Acestea sînt, ambele, protejate prin "dongles", lucru mai puțin stînjitor la Microfocus Cobol decît la Realia Cobol. "Dongle"-ul de la Realia este deranjant de mare: a trebuit reamplasat adaptorul VGA pentru a-i face loc. În afară de aceasta, acest dongle vrea neapărat ca imprimanta să fie legată la prima interfață paralelă și să fie neapărat online; mai mult, nici nu se împacă cu alte "dongle"-uri (ca de exemplu cel de la Microfocus).

MS-Cobol 3.0 și Microfocus Cobol sînt aproape 100% compatibile. Însă în loc de IXSIO, Microfocus la linkeditare are nevoie de biblioteca EXTFX.

Acu-Cobol, exact ca și MS-Cobol v2.1, are nevoie de clauza "ASSIGN TO DISK". în loc de "VALUE OF FILE-ID...", Acu-Cobol acceptă însă numai "VALUE OF LABEL...". Ecranul nu e șters în mod automat.

Nici la MBP, ecranul nu e șters. Instrucțiunea ACCEPT produce automat un semnal sonor.

La RM-Cobol, puțin deranjant este mesajul de terminare, care determină o defilare în sus a întregului ecran.

Probleme ridică, în general, instrucțiunile ACCEPT și DISPLAY. Deoarece aceste instrucțiuni sînt prea puțin standardizate, fiecare producător "condimentează" după gustul propriu aceste instrucțiuni. Plasarea cursorului într-un anumit punct poate provoca cele mai neplăcute surprize. De aceea, ca programator profesionist de Cobol, care este nevoit deseori să porteze aplicațiile scrise, ar trebui să renunțați din principiu la atribuțiile ACCEPT și DISPLAY specifice unui producător și să utilizați în schimb interfața prin CALL. Prin aceasta, producătorii de compilatoare oferă posibilități de programare deosebit de individualizate - astfel, rutinele confortabile de tratare a ecranului rămîn locale și se pot adapta independent de restul programului, "dintr-o singură mișcare".

(I.F.)

Cobol	Comenzile necesare lansării programului din fig.1	Timpul consumat (secunde)	Preț (USA, apr. '91)
MS-Cobol 2.1	cobol bench isam runcob bench isam/f	34,6	
MS-Cobol 3.0	cobol bench.cob link bench ixio bench	8,3	629 \$
MS-Cobol 4.0			1299,60 DM (RFG iunie '91)
Microfocus Cobol	cobol bench.cob link bench extfx bench	8,4	1499 \$ (W/tools)
Realia Cobol	realcob bench link bench,,,realcob bench	6,6	844 \$
Acu-Cobol		29,8	
MBP	viscob bench link vcmain bench,bench,nul,vcobol+vcsort+vcisam bench	12,5	
RM-Cobol	rmcobol bench runcobol bench	14,5	

Figura 2.

Un limbaj gigant cu un nou aspect

MS - Cobol V 4.0 PDS

Cel ce-și aduce aminte de primul compilator Cobol pentru PC va rămâne perplex în fața celei mai noi versiuni. Editoarele confortabile și multele programe utile ușurează foarte mult munca programatorului profesional. În sfârșit te poți concentra asupra problemei principale - programarea. Multe munci suplimentare care înainte cădeau în sarcina programatorului sînt preluate acum de compilator.

Microsoft a realizat cu această ultimă versiune Cobol un pas consecvent în direcția corectă. Fiind echipată cu o multitudine de instrumente de programare utilitare, versiunea 4.0 își merită pe deplin numele de PDS: Professional Development System.

Cel mai băttător la ochi este noul mediu de proiectare, cunoscut deja de la aite limbaje, și anume PWB (Programmer's Workbench). Acesta garantează fără probleme trecerea de la un limbaj de programare la altul. Și cel care lucrează doar în Cobol se va obișnui foarte repede cu noul mediu de editare. Acesta este realizat în standard SAA și se poate deservi cu mouse-ul. Cel care renunță la mouse poate lucra și cu tastatura. Deservirea tastelor este standardizată.

Toate operațiunile, care sînt necesare fiecărui program al pachetului se pot utiliza prin suprafața utilizator. La acestea se adaugă și instalarea propriu-zisă a compilatorului, a linkeditor-ului și a fiecărui debugger.

Există posibilitatea, foarte avantajoasă, de a deschide foarte multe ferestre în același timp. Acestea pot conține de pildă surse program, texte de "help" sau mesaje de eroare.

Dacă totuși lipsește din PWB o funcție des folosită, atunci ea se poate îngloba ulterior în mod foarte simplu. În acest scop trebuie scris doar un macrou corespunzător, care trebuie introdus în fișierul TOOLS.INI - Dacă este necesar, atunci un macrou propriu poate recurge la funcții deja existente în PWS sau la alte macroui.

Programatorii știu să aprecieze proprietatea de memorare a compilatorului la terminarea lucrului. La reluarea unui proiect de program se poate continua cu aceleași condiții.

Help-ul on line este exemplar, el înlocuiește căutarea repetată în manualul de utilizare. Dacă cursorul este plasat pe un cuvînt cheie Cobol, la apăsarea tastei F1 apare o nouă fereastră. Această fereastră conține

toate informațiile importante cu explicații aferente funcțiunii și o descriere detaliată a sintaxei.

În domeniul detectării erorilor, Cobol 4.0 oferă multe instrumente de ajutor. "Source Browser" vizualizează informații importante despre procedurile existente și despre legăturile variabilelor introduse. Singura condiție este folosirea comutatorului corespunzător care duce la crearea unui fișier BSC ("Source Browser

```

$SET ANS85 NOOSVS MF
IDENTIFICATION DIVISION.
PROGRAM-ID. RECUR.
* Acest program demonstreaza utilizarea
* apelurilor recursive in MS-Cobol 4.0
AUTHOR. D-S.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01 tab1 pic x(02) occurs 10.
01 tab2 pic x(02) occurs 10.
LOCAL-STORAGE SECTION.
01 z1 pic 9(02).
LINKAGE SECTION.
01 z2 pic 9(02).
PROCEDURE DIVISION.
MAIN SECTION.
CALL X"E4".
MOVE 1 TO Z1
CALL "UMPLE" USING Z1.
MOVE 1 TO Z1.
CALL "TABELA" USING Z1.
STOP RUN.

UMPLE-TABELA SECTION.
ENTRY "TABELA" USING Z2.
IF Z2 > 10 THEN
DISPLAY "Tabela - 2 umpluta" at 1110
ELSE
MOVE TAB1 (Z2) TO TAB2 (Z2)
MOVE Z2 TO Z1
ADD 1 TO Z1
CALL "TABELA" USING Z1
END-IF.
EXIT PROGRAM.

INITIALIZARE-TABELA SECTION.
ENTRY "UMPLE" USING Z2.
IF Z2 > 10 THEN
DISPLAY "Tabela initializata" at 1010
ELSE
MOVE Z2 TO TAB1 (Z2) Z1
MOVE Z2 TO Z1
ADD 1 TO Z1
CALL "UMPLE" USING Z1
END-IF.
EXIT PROGRAM.

```

* RECUR.CBL Sfirsit

Exemplu de apel recursiv în Cobol 4.0

Cache*). Acest fișier conține toate informațiile necesare ulterior lui Source Browser.

Dacă eroarea nu poate fi depistată astfel, atunci se vor folosi Code view - Debugger (doar sub OS/2) sau Animator - Debugger. El poate fi inițiat direct de pe suprafața utilizator dar el nu poate fi mînuit prin mouse.

Nou în comparație cu versiunea anterioară este și generatorul de măști integrat (screen-uri). Astfel măștile pentru programare se pot crea mult mai ușor. Nici generatorul de măști nu poate fi deservit cu mouse.

Compilerul generează un cod program mult mai compact și mai simplu decît în versiunea precedentă. Aceasta se poate remarca mai ales la importul și exportul de imagini (ecrane) și la utilizarea fișierelor. Timpul de răspuns a scăzut în cazul I/E ecran cu 50%.

La fel de simplă ca elaborarea aplicațiilor DOS este și elaborarea aplicațiilor pentru managerul de prezentare sub OS/2. Programatorul poate alege între o linkeditare statistică și una dinamică a bibliotecilor sau a subprogramelor. Acest lucru este valabil și pentru DOS.

Avantajele sistemului OS/2 sînt clare. Linkeditarea dinamică face ca toate subprogramele linkeditate corespunzător să devină părți ale DLL (Dynamic Link Library) a OS/2. Astfel fișierele executabile obținute sînt mult mai compacte și mai rapide, deoarece părțile necesare sînt activate doar în timpul execuției progra-

mului. Astfel se face și o economie serioasă de memorie.

Pentru simplificarea programării de aplicații pentru Presentation Manager există programul suplimentar H2CPY. Acesta traduce fișierele de header C în echivalentele lor Cobol. De multe ori merită încercată transformarea unui program C în unul Cobol. Astfel se poate economisi mult efort de scriere.

Totodată Cobol 4.0 oferă și posibilitatea lucrului cu SQL. Arhitectura client-server a fost îmbogățită în noua variantă cu posibilitatea de comunicare cu eventuali serveri de baze de date SQL. Compilerul permite integrarea instrucțiunilor SQL în programele Cobol cît și interogarea directă a bazelor de date SQL.

Pentru orice proiect suprafața utilizator generează o bază de date proprie proiectului, iar Cobolul memorează și administrează toate informațiile importante despre proiectul respectiv.

Această bază de date poate fi apoi deservită cu ajutorul lui Source Browser. Astfel manipularea programelor Cobol deosebit de voluminoase devine mult mai simplă. Mai ales Source Browser ajută programatorul să se descurce foarte repede într-un program Cobol necunoscut.

Programul răspunzător pentru aceasta se numește NMAKE. El recunoaște imediat ce părți de programe dintr-un anumit proiect au rămas nemodificate și se ocupă doar de părțile care au suferit o modificare de la ultima prelucrare. O altă versiune specială a acestui program, denumită NMK reușește să genereze și programe voluminoase, care necesită mai multă memorie decît memoria disponibilă. Acest lucru are sens cînd se lucrează în Real Mode sau sub OS/2.

Programatorului i se oferă și funcțiuni noi pentru programarea mouse-ului. Modul de inițializare a mouse-ului ni-l arată programul MOUSE.CBL. Un dezavantaj îl constituie faptul că în manual se fac mereu trimiteri de la un capitol la altul și faptul că exemplele de programare nu sînt grupate. Nu este chiar simplu, deci, să te orientezi, cu toate că în rest manualele nu lasă aproape nici o dorință neîndeplinită. În manuale există chiar și o descriere a modului în care trebuie făcut transferul programelor sursă de pe sistemele IBM/370 pe PC-uri (Compatibility Guide).

Noi utilitare: ILINK - un linkeditor incremental, QH - poate vizualiza chiar și helpuri online la nivelul liniei de comandă. Sistemul de HELP-uri este foarte bine construit, texte de help existente se pot modifica cu ajutorul utilitorului HELPMAKE, putînd fi adăugate și texte noi.

(I.M.)

```
IDENTIFICATION DIVISION.
PROGRAM-ID. MAUS.
* Acest program demonstreaza modul
* de initializare al unui mouse
AUTHOR. D-S.
ENVIRONMENT DIVISION.
DATA DIVISION.
WORKING-STORAGE SECTION.
01  MOUSE-FUNCTIE PIC 9(02) COMP-X VALUE 64.
01  MOUSE-PARAM PIC 9(02) COMP-X.
01  MOUSE-DETAILS-G PIC 9(02) COMP-X VALUE 67.
01  MOUSE-DETAILS.
05  MOUSE-X-POZ PIC 9(04) COMP-X.
05  MOUSE-Y-POZ PIC 9(04) COMP-X.
05  MOUSE-STATUS PIC 9(04) COMP-X.
01  CHAR PIC X.

PROCEDURE DIVISION.
CALL X"E4".
* Activare mouse
MOVE 1 TO MOUSE-PARAM.
CALL X"AF" USING MOUSE-FUNCTIE MOUSE-PARAM.
ACCEPT CHAR AT 1010.
* Dezactivare mouse
MOVE 0 TO MOUSE-PARAM.
CALL X"AF" USING MOUSE-FUNCTIE MOUSE-PARAM.
STOP RUN.
```

* MOUSE.CBL Sfsirit

Inițializarea mouse-ului în Cobol

Încă o dată mai rapid

Foxpro 2.0

În cursa bazelor de date, Foxpro 2.0 a luat din nou un avans în fața concurenților dBase IV și Paradox 3.5.

Sistemul de gestiune al bazelor de date (SGBD) Foxpro 2.0 conține o serie de îmbunătățiri, comparativ cu versiunile anterioare, care ne îndreptățesc să afirmăm că nu este vorba doar despre o actualizare cu modificări pur cosmetice. În primul rând ne referim la viteza de lucru, care a fost sporită din nou. Memoria de lucru necesară a fost minimizată și au fost adăugate facilități SQL, noi tipuri de indexuri și utilitare pentru programatorii de aplicații.

Foxpro 2.0 va fi disponibil în două versiuni, care vor fi livrate însă împreună fără ca pentru aceasta să se perceapă o taxă suplimentară: o versiune pentru PC/AT și o versiune extinsă (Extended Edition) pentru procesoare 80386. Utilizarea versiunii extinse este recomandată în cazul bazelor de date cu mai mult de 500.000 de articole, pentru a îmbunătăți sensibil performanțele, încă o dată. Cu aceasta Foxpro 2.0 nu numai că doboară două muște cu o singură lovitură, dar folosește și puternica arhitectură a magistralei de 32 de biți.

În pas cu moda în sectorul bazelor de date și Foxpro 2.0 lucrează acum cu instrucțiuni SQL (Structured Query Language). Foxpro 2.0 include instrucțiunea "Select" și a fost extins și cu alte instrucțiuni.

Interogările Select se autooptimizează în funcție de dimensiunea tabelului, indexurile disponibile, mărimea memoriei de lucru și a spațiului disponibil pe harddisk etc.

Comanda poate crea indexuri temporare pentru a mări performanțele. Acest SGBD poate fi astfel recomandat și pentru utilizare în cazul lucrului în rețea.

Pe lângă interogările SQL, Foxpro 2.0 dispune și de "RQBE" (Relational Query By Example), o metodă puternică de interogare a bazelor de date prin exemplu. Chiar și interogările SQL pot fi testate astfel comod de către începători și poate fi învățat astfel modul de comportare al unor comenzi.

Sistemul de management al bazei de date mai aduce o noutate care vorbește de la sine despre inventivitatea programatorilor de la Foxsoftware: tehnologia Rushmore. Este vorba despre o procedură de optimizare a interogării unei baze de date, dezvoltată în propria casă. După datele furnizate de Foxsoftware, această versiune atinge viteze de interogare de până la 1.000 de ori mai mari decât în cazul versiunilor anterioare, datorită acestei tehnici. Chiar și bazele de date uriașe cu milioane de articole vor putea fi prelucrate astfel pe PC-uri la viteze comparabile cu cele de pe mainframe-uri (calculatoare mari). Avantajele tehnologiei Rushmore devin evidente abia în cazul utilizării versiunii extinse pentru volume uriașe de date.

Foxpro 2.0 cunoaște acum trei tipuri de indexuri: indexul standard (.IDX) al versiunilor Foxbare și Foxpro 1.XX, noul "Compound Index" (.CDX) analog indexului "Multiple Index File" din dBase și indexul compact. Fișierele de indexuri conțin mai mulți pointeri. Suplimentar există un așa-numit "structural compound index file", care este deschis automat în momentul deschiderii bazei de date complementare. În cazul indexului compact este vorba despre o formă

comprimată a indexurilor sus-numite. Foxpro 2.0 utilizează, în acest caz, o tehnică proprie de comprimare care reduce fișierele index la cca. o șesime din dimensiunea lor inițială. Chiar și acest lucru mărește viteza de execuție, deoarece fișierele care trebuie citite de pe harddisk sînt mai mici iar decomprimarea se face în memorie.

Per ansamblu acest program lucrează cu o viteză care-i poate face invidioși pe concurenți ca dBase IV sau Paradox 3.5. Pentru acei care pun preț pe viteza de răspuns la interogarea unei baze de date acesta ar putea fi un criteriu decisiv.

Foxpro s-a orientat pînă în prezent pe standardul dBase căruia a încercat să-i dea o putere sporită. Din acest motiv este asigurată compatibilitatea nu numai cu versiunile anterioare de Foxpro ci și cu dBase III și IV.

Comparativ cu standardul pieței, versiunea beta a lui Foxpro 2.0 nu a strălucit la modul de comandă al imprimantelor. În meniul de imprimare există câteva comenzi rudimentare pentru stabilirea marginilor, a orientării, etc., dar driver-urile speciale pentru imprimante matriciale și laser lipsesc. Dacă se dorește, de exemplu, listarea din Foxpro 2.0 pe o imprimantă HP-Laserjet, terminatorii de linie și alte secvențe de întreruperi trebuie realizate cu secvențe de comenzi. Pentru profesioniști acest lucru nu constituie o problemă, dar începătorilor nu le este tocmai ușor.

După informațiile de la Foxsoftware, pînă la lansarea pe piață, în această vară, vor mai fi aduse câteva îmbunătățiri la acest modul.

Generatorul de rapoarte, controlabil în întregime prin meniuri, poate lucra cu variabile, cu câmpuri suplimentare de calcul (abatere, varianță) și poate repeta antetul de grup pe fiecare pagină. Salutară este și opțiunea care permite restartarea numărului paginii la schimbarea grupului. Definiția unui raport poate fi memorată.

Introducerea și poziționarea de câmpuri și de texte în interiorul unui raport se poate face comod cu ajutorul mouse-ului.

Pe lângă tehnica ferestrelor cunoscută din versiunile anterioare, Foxpro dispune de o serie de utilitare care în viața de zi cu zi se pot dovedi a fi foarte folositoare.

Pentru operațiile obișnuite cu fișiere utilizatorul poate utiliza managerul fișierelor care permite copierea, ștergerea, mutarea, fișierelor sau a unor întregi direcții.

Un calculator de buzunar ajută la efectuarea unor calcule intermediare, de asemenea mai fiind disponibile și o tablă ASCII și un calendar. Pentru destindere este oferit un joc de puzzle.

Foarte folositor se dovedește a fi și clipboard-ul (suportul, magazia intermediară). Părți ale conținutului ecranului pot fi astfel "captate" și inserate astfel în alte tabele. În acest mod se elimină necesitatea retastării unor date. Același lucru este valabil, într-o anumită măsură și pentru descrierile de rapoarte. Părțile standard pot fi reutilizate cu ajutorul comenzilor "Cut and Paste".

După cum se știe, Foxpro nu este doar un SGBD foarte ușor de învățat și foarte rapid ci și un instrument confortabil pentru proiectanții de aplicații cu baze de date. Pentru aceștia din urmă Foxpro 2.0 are implementate câteva noutăți interesante.

Printre acestea se numără structura deschisă a programelor, care permite programatorilor cuplarea prin intermediul API (Application Program Interface) unor rutine externe scrise în C sau Assembler.

În acest mod pot fi realizate și funcții pe care propriul limbaj de programare - un fel de dBase extins - nu le poate realiza. În plus pot fi utilizate acum și biblioteci "externe" standardizate, ceea ce duce desigur la ridicarea productivității

muncii. Avantajoasă este și utilizarea generatoarelor de machete și meniuri.

Generatorul de machete lucrează cu obiecte care pot fi înglobate în ecrane și în meniuri. Sursa program care trebuie executată pentru prezentarea machetelor și meniurilor este generată fără complicații. Suprafața utilizator Foxpro 2.0 poate fi astfel extinsă cu ușurință cu puncte suplimentare în meniuri și cu ecrane suplimentare și poate fi combinată cu alte programe și rapoarte.

Proiectarea este supravegheată de managerul de proiecte care permite documentarea tuturor programelor, măștilor, meniurilor, etc., care sînt necesare pentru generarea unei aplicații. Fișierul proiect poate fi utilizat pentru crearea unei aplicații (.FXP) sau a unui program executabil (.EXE).

Foxpro 2.0 este un produs reușit, noua versiune fiind recomandabilă nu numai ca o îmbunătățire pentru fanii de Foxpro ci și ca o alternativă pentru dBase.

(R.M.)

Tabel comparativ

Program	Foxpro 2.0	dBase IV/1.1	Paradox 3.5
Producător	Foxsoftware	Ashton-Tate	Borland
Preț (DM)	Un post de lucru 2502, LAN (pt. fiecare server) 4098	2679	2451
Meniuri pull-down	da	da	nu
Număr articole	1 miliard	1 miliard	2 miliarde
Dimensiune maximă articol	4000 Byte	4000 Byte	4000 Byte
Număr câmpuri	255	255	255
Limbaj de programare	da	da	da
Suport SQL	da	da	cu SQL-Link
Protecție prin parolă	da	da	da
Integrare de imagini	da	nu	nu
Suport Windows 3.0	nu	nu	nu
Manager fișiere	da	nu	nu
Tehnica ferestrelor	da	nu	nu
Tehnici speciale de indexare	da	da	nu

Recunoașterea scrisului

Rețele neuronale

Partea a IV-a

În ultimii ani, recunoașterea scrisului prin calculator s-a perfecționat mereu. Recunoașterea de caractere prin intermediul rețelelor neuronale este un pas într-o direcție promițătoare.

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

La prima vedere pare ciudat ca recunoașterea de caractere să reprezinte pentru calculator o problemă foarte dificilă, când omul nu are nici un fel de dificultăți și doar și calculatorul lucrează numai cu cifre și litere. Dacă acestea sînt introduse prin intermediul tastaturii, aceasta desigur nu este o problemă pentru mașină, deoarece recunoașterea unei litere este făcută de om, iar prin tastare caracterul este transformat într-un cod care poate fi prelucrat de calculator (cod ASCII).

Dacă însă procesul de recunoaștere care are loc în timpul citirii trebuie executat de mașină, atunci în primul rînd trebuie "transformat" în informație numerică suportul inițial - această pagină de revistă de exemplu. Acest lucru se face prin intermediul unei camere video sau cu un scanner, care parcurg rînd cu rînd suportul de pornire, preluînd informații cu ajutorul unor

fotodiode. La echipamente ce lucrează alb-negru, diferențele luminoase - întunecat sînt transformate pentru reprezentare pe ecranul calculatorului în succesiuni de biți: un bit setat corespunde unui punct întunecos, un bit nesetat unuia luminos. Deoarece aceleași litere, ca succesiuni de biți, apar diferit funcție de formă și mărime, numai un algoritm performant este capabil să realizeze interpretarea.

Procesul de recunoaștere la om se bazează pe următoarele etape: prin acomodarea diafragmei, literele apar într-un soi de "formă normalizată", într-o diferențiere cît mai netă față de fundal. Apoi, omului îi reușește prin asociere să completeze litere incomplete și să citească și versiuni deformate. Această capacitate este aplicată și la cuvinte întregi. Deoarece omul folosește întotdeauna și contextul cuvintelor în propoziții sau al literelor în cuvinte, caractere asemănătoare, cum ar fi "1" și "l", se deosebesc prin faptul că cifra "1" nu apare niciodată în interiorul unui cuvînt în texte obișnuite, iar litera "l" nu apare izolat sau în cadrul unui număr.

Recunoașterea corectă a caracterelor

Sarcina programelor OCR (Optical Character Recognition) este de a simula cît mai bine aceste caracteristici umane. În multe programe își găsesc aplicație mai ales două metode: suprapunerea de modele (pattern matching) și identificarea de caracteristici (feature recognition). La suprapunerea de modele (compararea de forme - pattern matching) succesiunile de biți generate de scanner sînt comparate, pixel cu pixel, cu litere "ideale", memorate în calculator. Deoarece caracterele scanate sînt însă foarte diferite în mărime sau lățime, ele



Peter Schmitz a studiat matematica și biologia; în paralel, s-a ocupat cu probleme de evoluție relative la autoorganizarea proceselor vii. Lucrarea sa de diplomă (1988) se plasează undeva între geometria algebrică și teoria numerelor. A realizat un soft de simulare al modului de lucru al rețelelor neuronale și a fondat în 1989 o întreprindere care se ocupă de aplicarea metodelor inteligenței artificiale în aplicații industriale.

trebuie mai întîi "normalizate" la dimensiuni unitare. Chiar și în acest caz, imagini binare ale unui același caracter pot fi foarte diferite (vezi fig. 1). Soluționarea acestei probleme se încearcă prin memorarea mai multor modele pentru fiecare literă și admiterea unui domeniu de toleranță. O toleranță prea mare duce însă la identificări eronate. Dacă se memorează prea multe modele, devine necesară multă memorie și crește timpul de calcul.

În cadrul procedurii de identificare a caracteristicilor, literele sînt caracterizate prin proprietăți topologice. De exemplu "C" este determinat printr-o linie, arcuită convex spre stînga. Apar însă și senzori multiple: atît cifra "0" cît și litera "O"

sînt caracterizate printr-un drum închis. Înainte de a putea identifica literele prin caracteristici, secvențele de biți trebuie să "skeletonizate". (Adică, de exemplu, dintr-o linie de 3 pixeli se obține una de un singur pixel).

În ciuda anunțurilor de reușită a diverși producători, suporturile copiate le creează dificultăți majore

puțini pixeli. Caracterele memorate se folosesc ca mostră de antrenament pentru o rețea neuronală. Cum s-a mai spus, fiecărei mostre i se atașează un model de ieșire (respectiv, apartenența la o celulă de ieșire). Deci de exemplu mostra de pixeli pentru "A" activează celula de ieșire 1, mostra "B" celula de ieșire 2 ș.a.m.d.

explicită a algoritmului, astfel că această configurație se produce independent, în cursul fazei de învățare. Proprietățile ei sînt dirijate prin intermediul mostrelor de antrenament și al funcțiilor obiectiv asociate.

Avînd odată o rețea "antrenată", datorită asociativității rețelelor neuronale, imagini binare

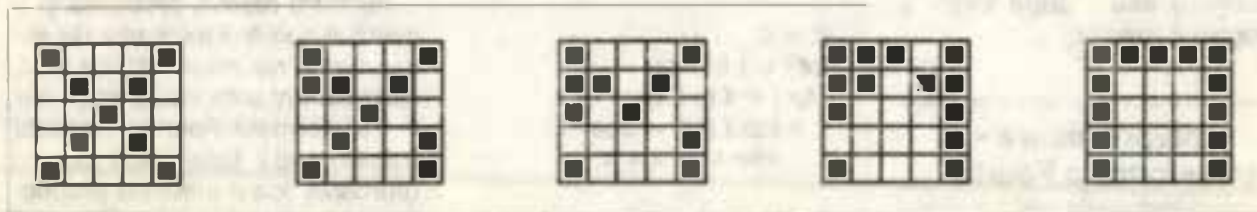


Figura 1 arată dificultățile interpretării mostrelor de pixeli. Secvența reprezintă transformarea unui X într-un U. Dacă al doilea model din stînga încă poate fi recunoscut ca "X" și al doilea model din dreapta încă poate fi interpretat ca "U", la modelul din centru sînt posibile noi interpretări (de exemplu, K sau W).

mai tuturor programelor OCR. La digitalizare sau sînt "lipite" prea multe litere, astfel încît nu sînt interpretate ca litere singulare, sau sînt atît de "sparte", încît dintr-un caracter apar mai multe. Chiar și litere izolate corect sînt deseori percepute atît de deformat încît devin ilizibile. În timp ce segmentarea literelor, deci fragmentarea unui cuvînt în componentele sale individuale, depinde în primul rînd de performanțele altor procedee, rețelele neuronale își dovedesc eficiența mai ales la identificarea caracterelor deformat sau "sparte".

Împreună cu programul de simulare NBP se livrează un exemplu de recunoaștere a scrisului cu care se pot face experimentări. Intrarea rețelei constă din 18x18 pixeli iar ieșirea din 99 de celule, pentru recunoașterea a 99 de caractere distincte. În exemplu nu se tratează însă "decuparea" unor caractere dintr-un text întreg, ci doar identificarea de mostre de pixeli, pentru a clasifica corect literele.

Fișiere de pixeli scannate în format IMG pot fi reprezentate pe ecran cu ajutorul unui program specific; acesta decupează litere dintr-un text și le memorează într-un fișier propriu. Este posibilă și micșorarea imaginii, pentru a reduce volumul de date. Funcție de rezoluția scannerului, pentru un caracter se obțin mai mulți sau mai

Aceste informații relative la scopul învățării sînt memorate odată cu mostrele de pixeli. Pot fi imaginate desiguri și alte codări ca funcții obiectiv ale rețelei neuronale - de exemplu, direct valoarea ASCII corespunzătoare diverselor caractere. În acest caz ar fi necesare numai 3 celule de ieșire pentru reprezentarea zecimală, respectiv 8 pentru reprezentarea binară. Cercetări proprii au relevat însă faptul că acest mod de reprezentare a informației constituie o îngreunare serioasă a procesului de convergență în faza de învățare, mai ales datorită faptului că prin folosirea funcției sigmoide (vezi partea a II a acestei serii) stabilități se formează numai în jurul valorilor 0 și 1. În plus, litere total distincte au coduri binare ASCII asemănătoare, ceea ce se răfrînge negativ asupra asociativității necesare.

Sarcina fazei de învățare este de a modifica ponderile din rețea în așa fel încît să apară o configurație care satisface caracteristicile cerute ale fișierului de antrenament. Ceea ce deosebește rețelele neuronale de alte tehnici este faptul că nu mai este necesară o descriere

asemănătoare ale unor litere sînt clasificate corect. Ceea ce, în cadrul codării menționate inițial, înseamnă că se activează o singură celulă de ieșire și se dezactivează toate celelalte. Prin "mărirea" activității există o măsură suplimentară a apartenenței la o clasă. Ea poate fi utilizată ca definiție a domeniului de toleranță: o mostră aparține unei clase dacă valoarea de ieșire corespunzătoare este peste un anumit prag, iar toate valorile celelalte se găsesc sub un anumit prag. Dificil este însă a-ți face o imagine clară despre noțiunea pomenită mai sus de "asemănare". Cît de mult trebuie să fie corelate două imagini binare, pentru a fi considerate "asemănătoare"?

Se pot cu ușurință imagina realizări foarte diferite ale unei aceleași litere, care diferă însă foarte mult prin amplasarea pixelilor. Pentru a nu fi nevoiți, ca la "pattern matching", de a prelua toate combinațiile posibile în fișierul de antrenament, se recomandă și aici tehnici de anteprelucrare corespunzătoare. În programul de simulare NBP sînt realizate 3 tipuri.

În primul rând, există posibilitatea translatații: imaginile de pixeli sînt duse în colțul din stînga-sus, iar coloanele și rîndurile vide sînt pur și simplu eliminate.

În al doilea rînd, în imaginile micșorate - și micșorarea este de fapt o anteprelucrare - se poate varia pragul pentru amplasarea unui nou pixel. Astfel se realizează o subțiere sau - după caz - o îngroșare a literelor.

Anteprelucrare - transformata Fourier

Ca o a treia tehnică este oferită o transformată Fourier normată bi-dimensională. La intrări pătratice, ea are proprietatea deosebită a invarianței la translație. Ceea ce înseamnă că indiferent unde se găsește o imagine binară în cadrul unui cîmp de intrare, ea este proiectată totdeauna la fel.

Totuși, chiar și transformata Fourier rapidă (FFT) necesită calcule laborioase. (Ea se folosește totuși, în mod standard, în electrotehnica datorită proprietăților deosebite în analiza funcțiilor periodice - e drept, în acest domeniu, transformata Fourier unidimensională).

Ca suport pentru recunoașterea scrisului, datorită consumului de timp, ea se recomandă numai cu un hardware specializat. Se poate pune întrebarea la ce mai e pomenită atunci, în acest context. Deoarece în această serie de articole nu poate fi vorba de prezentarea unor soluții complete pentru probleme speciale, ci doar de introducerea în tehnici noi de prelucrare a informațiilor, cititorului i se oferă un prim contact care să-i permită manipularea corectă a acestora, cîteva puncte de plecare posibile pornind de la exemple.

Cazul combinației transformată Fourier/propagare înapoi este discutat pe exemplul aflării numărului de pixeli setați într-o suprafață pre-

determinată. Fie dată o suprafață de 5x5 pixeli. În această suprafață de 25 pixeli se admit pînă la 5 pixeli "setați" - avînd valoare logică "true". Rețelei neuronale i se cere să "numere" corect pixelii setați.

Este evident că pe alte căi soluția se obține mult mai ușor și mai repede. O buclă simplă de contorizare ar putea fi:

```
k = 0;
for i = 1 to 5 do
  for j = 1 to 5 do
    if input (i, j) = true
      then k: = k + 1;
```

Aici însă, algoritmul este dat explicit. Spre deosebire de aceasta, rețeaua trebuie să "găsească" singură algoritmul, pornind de la cîteva (puține) mostre de antrenament!

Cu NBP se produce o rețea de 5X5 celule de intrare și 5 celule de ieșire, pentru clasele "un pixel", "doi pixeli", pînă la "5 pixeli".

Problema din ultimul articol, de

combinație: pentru a plasa un pixel într-o suprafață de 25 de cîmpuri diferite, există 25 de posibilități. Pentru 2 pixeli, există $25 \times 24 / 2 = 300$ de posibilități, deoarece nu importă ordinea plasării. Corespunzător, pentru 3 pixeli există $25 \times 24 \times 23 / 3 / 2 = 2300$, pentru 4 pixeli 12.650 și pentru 5 pixeli chiar 53.130 de combinații.

Pentru a rezolva problema și a menține numărul mostrelor de antrenament necesare cît mai mic, mostrele sînt anteprelucrate printr-o transformată Fourier. Această transformată furnizează pentru (aproape) toate mostrele proprietatea invarianței la translație și o invarianță la oglindire axială pe axa x. După transformare Fourier, mostrele din figura 3 sînt identice.

Spre deosebire de translația pomenită mai sus, la care numai pixelii nesetați din stînga și de sus sînt eliminați, valorile cîmpurilor nu mai constau în continuare din valori de "0" și de "1", respectiv stările "adevărat și fals", ci din transformate Fourier, deci numere în virgulă

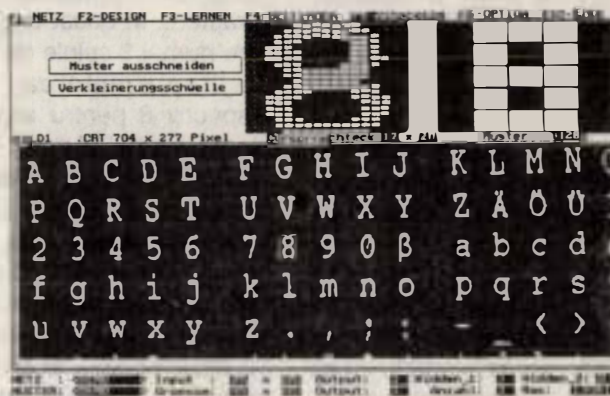


Figura 2 arată ecranul vizibil din NBP la prelucrarea de imagini scannate. Cu dreptunghiul cursor reprezentat mărit în partea superioară, se poate decupa o mostră. Aceasta este apoi micșorată la dimensiunea cerută la intrare, folosind un prag selectabil. Rezultatul micșorării este afișat pe ecran în colțul din dreapta sus.

a minimiza numărul mostrelor de antrenament, este reluată odată cu acest exemplu.

Mai întii, cîteva considerații preliminare relativ la posibilitățile de

mobilită. Avantajul metodei este determinat de toleranța mai mare la erori - cîte un pixel rătăcit, nedorit, nu are efecte așa de mari ca și la translație. Dezavantajos este calculul laborios ce trebuie efectuat.

Probleme cu posibilități foarte mari de combinare se abordează cel mai bine în felul următor: după selecția unei anteprelucrări potrivite, din fiecare clasă (2,3,4,5 pixeli) se memorează câteva mostre și se antrenează rețeaua cu acestea. Din pixelii singulari nu există mostre diferite, deoarece transformata Fourier este totdeauna aceeași.

rețeaua se comportă așa cum pare "de bun simț": activează celula care se apropie cel mai mult de moștra dată, și anume celula 5.

Fără combinarea cu transformata Fourier, efortul de calcul în faza de învățare ar fi disproporționat de mare, deoarece ar fi fost de învățat și verificat posibilități de amplasare

frunzulițele din el. Ca utilizator de unelte pentru rezolvarea unor probleme reale ale industriei este însă mai interesantă întrebarea - cu ce este mai performantă această tehnică decât altele, uzuale?

Răspunsul la aceasta depășește cadrul acestei serii, în care pe baza unor noțiuni de bază

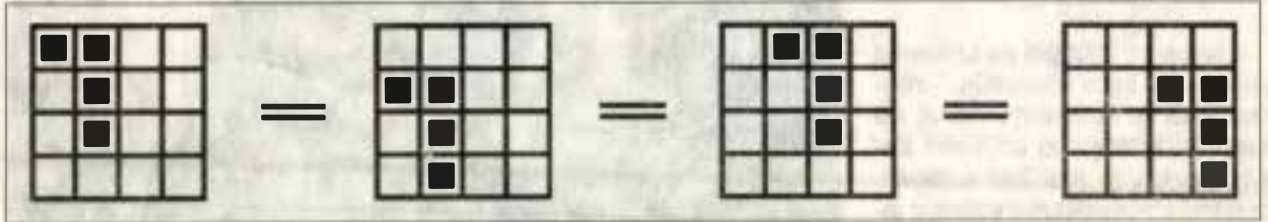


Figura 3. Transformata Fourier proiectează identic modele translatate. Rezultatul transformării se folosește ca intrare într-o rețea neuronală. În acest fel se obține un fel de preclasificare.

După terminarea cu succes a învățării, se testează comportarea rețelei cu mostre care îi sînt necunoscute. Dacă acestea sînt recunoscute corect, atunci ea au fost deja asimilate de rețea: rețeaua "generalizează" deci datele de antrenament cunoscute. Dacă noile mostre sînt identificate eronat, ele se adaugă la fișierul de antrenament, reluîndu-se faza de "învățare" a rețelei. Continuînd în felul acesta, se pot acoperi toate combinațiile posibile ale problemei. În exemplul de mai sus, sînt necesare doar 22 (!) de mostre de antrenament pentru a rezolva corect toate cele $25 + 300 + 2300 + 12650 + 53130 = 68.405$ combinații posibile.

a pixelilor din cele mai diverse. Sînt rețelele neuronale drumul spre "tehnica inteligentă"? Înainte de a răspunde la această întrebare, ar trebui clar definit ce se înțelege prin inteligență. Există oameni care susțin cu toată seriozitatea că o sită este inteligentă: deoarece știe să deosebească ceaiul de

sumare au fost prezentate câteva exemple de aplicare. Autorul are experiența unor produse finalizate în aceste domenii și a realizat câteva aplicații; cei interesați pot obține desigur informații suplimentare direct de la el.

Peter Schmitz

Reducerea timpului de calcul

După cum arată deja exemplul de mai sus, printr-o combinare adecvată cu alte tehnici, (spre deosebire de exemplul din ultimul articol), se pot obține reduceri substanțiale de date în fișierul de antrenament și respectiv timp de învățare mult reduși.

În afară de aceasta, în situații neprevăzute - ca de exemplu 6 sau 7 pixeli în câmpul de intrare -



Micro ATCI S.R.L.
4300 Tîrgu Mureș
C.P. 64, O.P.1
tel. 954/31660

oferă

celor interesați în **rezolvarea**, pe calculatoarele personale compatibile IBM-PC, a

sistemelor mari de ecuații lineare

SISREZ v1.0

un pachet de programe cu performanțe ridicate privind viteza și precizia de execuție.

Încă o dată de la început ?

OOP - Partea a patra

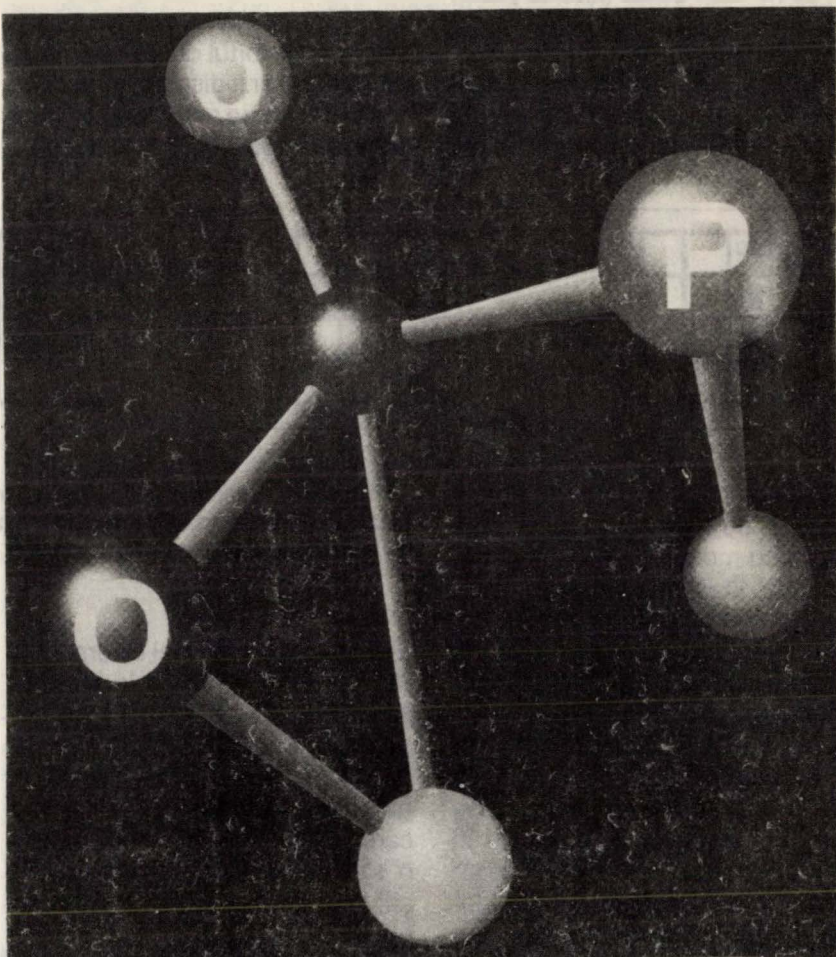
Cu BS_Tools va fi prezentată o mică ierarhie de obiecte care îmbunătățește confortul la introducerea și extragerea datelor.

Adesea se întâmplă ca utilizarea rutinelor de intrare/ieșire în Turbo-Pascal să fie incomodă. Astfel, de exemplu, la afișarea unui text trebuie poziționat mai întâi cursorul, de asemenea editarea valorilor introduse este incomodă. Programatorii se văd mereu puși în fața situației de a "scoate" mai mult din rutinele de bază. Ar fi foarte bine deci dacă la nevoie ar putea fi accesat un Unit care să ofere confortul necesar. Ar fi de imaginat rutine care să poată scrie și citi de pe ecran șiruri de caractere sau de cifre. În același timp se dorește ca problema extensiilor să rămână deschisă. Ulterior urmează ca modul de prezentare al aplicației să fie îmbunătățit prin introducerea ferestrelor și a boxelor de dialog.

Dar cine nu cunoaște deja problema pe care o implică astfel de extensii? Se modifică ceva într-un anumit loc și dintr-o dată nu mai funcționează nici un program care utiliza acel Unit.

Cu ajutorul obiectelor astfel de probleme pot fi evitate. Singura condiție este ca obiectele deja existente să nu fie modificate funcțional. Modificările trebuie să se axeze deci pe "ce" face un obiect. "Cum" face o metodă ceva, trebuie păstrat. Ceea ce este funcțional nou se definește oricum în noile tipuri de obiecte. Atunci când este necesar, aceste însușiri pot fi preluate de la obiectele disponibile.

Dacă se dorește programarea orientată obiect a tuturor uneltelor care îmbunătățesc confortul utilizării intrărilor și ieșirilor, atunci trebuie găsite similitudinile și diferențele. Caracteristicile identi-



ce pot fi definite în obiectul mamă, tipurile speciale de obiecte moștenind aceste caracteristici. Similitudinile nu trebuie căutate îndelung. Toate obiectele necesită metode care să le permită prezentarea pe ecran și îndepărtarea lor de acolo.

Dar este evident că un șir de caractere va necesita o altă rutină pentru reprezentare decât o fereastră. Comun este faptul că toate obiectele au nevoie de o metodă pentru a putea fi prezentate. Modul în care vor trebui implementate aceste metode va fi diferit.

Unit-ul care va prelua aceste tipuri de obiecte se va denumi **BS_Tools**. O primă prezentare a lui este prezentă în "Listing 1" și reprezintă fundamentul pentru o ierarhie de obiecte.

Comun tuturor obiectelor din Unit este faptul că ele trebuie să poată fi prezentate în poziția dorită de pe ecran. Obiectul mamă în acest caz ar putea fi **BS_Loc**. El definește două tipuri de date **X** și **Y**. Ele sînt identice cu poziția ulterioară de intrare sau de ieșire. Metoda **Init** este utilizată pentru stabilirea poziției. Cu **GetX** și **GetY** poate fi interogată poziția momentană a unui obiect pe ecran. Pot fi imaginate și alte funcții - cum ar fi de exemplu poziționarea cursorului. Din motive de spațiu ne vom limita însă la funcțiile anterior enumerate.

S-ar putea crede că metoda **Init** este inutilă, ea nu face altceva decât să transmită date, în rest nu se întâmplă nimic. Și totuși nu este așa. Definiția tipului **Obiect** (partea

întîi a acestui curs) exprimă clar faptul că un obiect este o reuniune a datelor și a metodelor de prelucrare *permise* asupra lor. "Dezavantajul" faptului că datele trebuie transmise mai întîi obiectului este compensat de faptul că datele interne nu pot fi influențate decît de metodele cărora li se permite acest lucru. Acest fenomen - denumit încapsulare a datelor - este tipic pentru programarea orientată obiect. Încapsularea poate fi întărită dacă definițiile obiectelor sînt date în partea Interface, iar partea de cod a metodelor aparținătoare în partea Implementation. Un utilizator care are la dispoziție doar partea "publică" a Unit-ului, trebuie să-și transmită datele spre obiect prin metodele `Init` disponibile. Altă soluție nu există. În acest mod datele interne sînt protejate împotriva unor modificări nepermise.

O a doua caracteristică foarte importantă a obiectelor este posibilitatea moștenirii caracteristicilor. Se definește deci un nou tip de obiect, care "se trage" dintr-altul. Cu aceasta noul obiect preia toate însușirile deja definite în obiectul mamă. Tipul de obiect `BS_Semn` este derivat din `BS_Loc`, vizibil prin numele obiectului mamă scris între paranteze. Un obiect de tipul `BS_Semn` poate deci dispune de datele și metodele obiectului `BS_Loc`.

Economie de text sursă

Deja în partea Implementation din Unit este utilizat acest lucru. Metoda `Init` a obiectului `BS_Semn` utilizează metoda `Init` a obiectului mamă.

Și metoda `Move`, pe lîngă două metode proprii, folosește și metoda `Init` a lui `BS_Loc`.

Cu ajutorul unui mic program de test se poate stabili imediat că și sarcinile lui `GetX` și `GetY` sînt înțelese de un obiect de tipul `BS_Semn`. Pe lîngă efectul protejării datelor împotriva unor accese nedorite prin încapsulare, se economisește

și cod, deoarece metodele comune nu trebuie redefinite.

Acest prim exemplu (Listing 1) care se sprijină puternic pe primele părți ale acestui curs este însă foarte subțire încă. Cu excepția scrierii, ștergerii și mutării unui caracter altceva nu se poate face.

Din acest motiv Unit-ul `BS_Tools` este extins în "Listing 2". Nou este aici obiectul `BS_String`. Metodele `Init`, `Draw` și `Delete` sînt definite din nou sau altfel exprimate: ele au fost adaptate pentru lucrul cu șiruri. Cu `Get_Str` poate fi stabilită valoarea parametrului intern `Str`. În maniera obișnuită, au fost utilizate metodele obiectului mamă pentru a se economisi cod. Metoda `Move` n-a fost declarată încă o dată pentru `BS_String` deoarece codul ei a rămas același. Trebuie acceptat deci faptul că metodele obiectului mamă pot fi preluate pe nevăzute.

Cu un mic program de test se poate observa că rutinele `BS_String.Draw` și `BS_String.Delete` își realizează în totalitate funcțiile. Numai mutarea textelor nu se face tocmai corect. Pe noua poziție de pe ecran va apare numai primul caracter din șir. Ce este greșit?

Sursa program propriu-zisă este corectă. O instanțiere a lui `BS_String` apelează rutina `Move`, în care prima instrucțiune este `Delete`. Dar despre care metodă de ștergere este vorba? Instanțierea ar dori să lanseze în execuție metoda `BS_String.Delete`. În loc de aceasta se execută însă cea din `BS_Semn`. Același lucru se întîmplă și în următoarea instrucțiune a programului. Din nou este apelată o metodă care nu se potrivește instanțierii. Acest lucru poate fi explicat astfel: `Move` este o metodă a lui `BS_Semn`, deci compilatorul stabilește faptul că `Draw` și `Delete` sînt metode care aparțin lui `BS_Semn`. Urmarea este că numai un caracter al șirului va fi mutat.

Trebuie deci programată o metodă proprie pentru `BS_String`, cu toate că nu se modifică nimic în textul sursă? Normal că nu trebuie. Cu un mecanism pentru metode

virtuale această problemă poate fi rezolvată. Acest mecanism nu va face altceva decît va lăsa liber în momentul compilării care metodă `Delete` și care metodă `Draw` urmează a fi utilizată. Aceasta va fi stabilit doar atunci cînd tipul instanțierii apelante este cunoscut.

O metodă pentru toți

Acest mecanism nu funcționează însă de la sine. El trebuie inițializat. Dacă se uită acest lucru, în locul saltului la metoda dorită se va afla altceva. Urmarea poate fi o cădere a sistemului. Compilatorul nu poate stabili însă dacă o instanțiere cu metode virtuale a fost inițializată. Răspunderea, ca mecanismul metodelor virtuale să fie inițializat pentru fiecare instanțiere apelantă, cade pe umerii utilizatorului.

Inițializarea are loc printr-un apel special de procedură. În antetul de procedură cuvîntul `Procedure` este înlocuit cu cuvîntul `Constructor`. Este avantajos să se modifice funcționarea unei proceduri deja existente cu constructor, care să fie apelată în orice caz înaintea unei metode virtuale. În Unit-ul `BS_Tools` acestea sînt rutinele `Init` ale obiectelor `BS_Semn` și `BS_String`. În ambele obiecte, metodele `Draw` și `Delete` trebuie să fie virtuale. Este suficientă adăugarea cuvîntului `Virtual` în declarație.

Dacă se compară "Listingul 3" cu "Listingul 2" se observă imediat unde au fost operate modificări. Un mic program de test va funcționa mulțumitor în acest caz. Unit-ul `BS_Tools` din "Listingul 2" poate fi abandonat.

Fără să fi scos în evidență pînă acum, a fost utilizată o altă caracteristică importantă a programării orientate obiect: ambele obiecte capabile de reprezentare de pînă acum folosesc identificări identice pentru a rezolva lucruri diferite.

Metoda `Draw` a obiectului `BS_Semn` face cu totul altceva decît metoda cu același nume a obiectului fiu `BS_String`. Cu `Delete` se întîmplă același lucru.

Această posibilitate, de a denumi acțiuni diferite cu același nume, este denumită polimorfism.

Pentru utilizarea metodelor virtuale mai există însă o restricție. Nu numai numele parametrilor trebuie să corespundă ci și tipul și numărul acestora. Deoarece **Draw** și **Delete** nu transmit parametri, în acest caz nu trebuie să ținem cont de această restricție.

Virtual și dinamic

În partea a treia a acestui curs ne-am referit la metodele virtuale. Dorim să subliniem încă o dată că metodele virtuale cuplate cu programarea dinamică oferă posibilitatea extinderii programării orientate obiect.

În "Listing 4" este prezentată în acest scop ultima versiune a Unitului **BS_Tools**. În partea Interface sînt prezenți, în fața definițiilor propriu-zise ale obiectelor, pointerii necesari.

Un obiect dinamic are avantajul că este creat doar atunci cînd este necesar. Mai ales atunci cînd sînt deschise simultan mai multe instanțieri ale unui tip apar avantajele gestionării dinamice a memoriei. La pornirea programului înafara celor 4 byte pentru fiecare pointer nu se rezervă altă memorie. Programatorul este cel care trebuie să gestioneze heap-ul (memoria disponibilă).

Dacă sînt necesare instanțieri atunci ele pot fi create și eliminate din memorie cu ajutorul procedurilor deja cunoscute **New** și **Dispose**. Metoda **Done**, inserată, servește la curățarea corectă a heap-ului. Metoda **Done** este declarată cu un nou cuvînt cheie: **Destructor**. Sarcina propriu-zisă de eliberare a memoriei, în cazul lui **Destructor**, se execută în fundal. Chiar dacă metoda nu conține nici o instrucțiune, ea își îndeplinește menirea. Un **Destructor** poate fi imaginat ca un **Unit**, care nu exportă nimic, dar care prelucrează o parte de inițializare.

Simultan în **BS_Tools** a fost definit un nou obiect: **BS_Intro**. El

exportă trei noi date (**I_String**, **I_Attr** și **I_Length**). Nucleul îl constituie funcția **ReadString**. Cu ajutorul parametrilor transmiși prin intermediul constructorului **InIt** este posibilă afișarea unui prompter într-o anumită poziție pe ecran și așteptarea introducerii unui șir de caractere în dreapta acestuia. **NRMax** stabilește lungimea maximă a șirului de editat. Este clară, și în acest caz, utilizarea diferitelor metode moștenite.

Totuși din motive de spațiu vă prezentăm doar trunchiul unei rutine de introducere confortabile. Oricine poate reproduce ceea ce se întîmplă în aceste rutine, n-ar trebui să aibă probleme în a-i face adăugirile necesare.

Obiectul **BS_Intr** poate constitui punctul de plecare pentru o multitudine de obiecte speciale. Sînt de imaginat astfel obiecte care să permită introducerea și editarea de numere de tipul **Real** sau **LongInt**. Obiectul ar putea prelua intern convertirea în formatul dorit. Imaginabilă este și stabilirea unui domeniu pentru valorile de intrare. Astfel ar putea fi introduse numere și s-ar putea simultan valida încadrarea lor în domeniul impus.

Obiectele mamă nu trebuie modificate

BS_Intr nu este însă un obiect tocmai curat programat. Cîteva metode moștenite nu funcționează corect. Nefiind utilizabile trebuie definite din nou. Dar și metoda **ReadString** trebuie modificată în cîteva locuri pentru ca datele interne să fie corect poziționate.

Pentru cititorii care doresc să încerce cîteva din modificările respective, cîteva sfaturi vor fi binevenite: pentru ca metoda **Move** să poată fi folosită în anumite condiții, parametrii interni **X** și **Y** trebuie să rămînă nemodificați după apelul lui **ReadString**. Deoarece obiectul nu poate fi mutat în timpul editării este suficient dacă șirul prompter și șirul de intrare sînt aranjate pentru mutare. Ele trebuie colectate deci împreună în parametrul intern **Str**.

Faptul că **Move** nu funcționează corect din cauza lui **ReadString** se poate observa cu ajutorul scurtului exemplu din "Listing 5". În modul "Trace Into" al debugger-ului se pot depista și corecta ușor anomaliile: deoarece **InIt** a fost declarat constructor, în metoda **Move** a lui **BS_Semn** (moștenire după două generații!) sînt utilizate deja metodele corecte **Draw** și **Delete**. Problema ține deci de utilizarea corectă a datelor interne.

Datorită exemplului din "Listing 5" a devenit evident cît de mult a crescut confortul cu această rutină. Cu puține linii program se poate introduce un șir de caractere în orice poziție dorită de pe ecran. Dacă trebuie introduse mai multe date pot fi deschise simultan mai multe obiecte. Atît timp cît acestea există în heap, șirul de intrare trebuie să fie oricînd disponibil. El poate fi afișat deci încă o dată la nevoie. De remarcat sînt și procedurile extinse **New** și **Dispose**. Primul parametru din paranteză specifică instanțierea pentru care trebuie rezervat sau eliberat spațiu. Al doilea parametru specifică rutina cu care va fi executat procesul.

Obiectele odată definitivate mai aduc un avantaj programatorului. El nu mai trebuie să știe decît ce fac obiectele. Cum funcționează obiectele intern rămîne necunoscut utilizatorului și nici nu-l interesează. Este suficient, de exemplu, să se știe că obiectul **BS_String** afișează un șir în poziția dorită de pe ecran. Dacă **BS_String** rezolvă acest lucru prin utilizarea procedurilor standard **Read** și **Write** sau - așa cum este cazul lui **BS_Tools** - prin scriere directă în memoria ecran, este neesențial.

Este evident faptul că abia utilizarea complexă a tuturor tehnicilor programării orientate obiect aduce avantaje efective. Pentru un începător este indicată familiarizarea pas cu pas cu particularitățile OOP. Utilizarea debugger-ului integrat este cea mai edificatoare asupra proceselor interne.

(R.M.)

```

Unit Bs_Tools;
Interface

Const
SegBWS = $B800; {Color}
{SegBWS = $B000; Mono}
OfsBWS : Word = $0000;

Type
BS_Loc = Object
  X,Y : Integer;
  Procedure Init (xPos, yPos : Integer);
  Function GetX : Integer;
  Function GetY : Integer;
End;
BS_Semn = Object (BS_Loc)
  Semn : Char;
  Attr : Byte;
  Procedure Init (xPos, yPos : Integer;
    Ch : Char; Co : Byte);
  Procedure Draw;
  Procedure Delete;
  Procedure Move (xNou, yNou : Integer);
End;

Implementation
Procedure BS_Loc.Init
  (xPos, yPos : Integer);
Begin
  X := xPos;
  Y := yPos;
End;
Function BS_Loc.GetX : Integer;
Begin
  GetX := X;
End;
Function BS_Loc.GetY : Integer;
Begin
  GetY := Y;
End;
Procedure BS_Semn.Init
  (xPos, yPos : Integer;
  Ch : Char; Co : Byte);
Begin
  BS_Loc.Init (xPos,yPos);
  Semn := Ch;
  Attr := Co;
End;

Procedure BS_Semn.Draw;
Begin
  OfsBWS := 2*(X-1) + 160*(Y-1);
  Mem[SegBWS : OfsBWS + 1] := Attr;
  Mem[SegBWS : OfsBWS] := Ord(Semn);
End;

Procedure BS_Semn.Delete;
Begin
  OfsBWS := 2*(X-1) + 160*(Y-1);
  Mem[SegBWS : OfsBWS] := 32;
End;

Procedure BS_Semn.Move
  (xNou, yNou : Integer);
Begin
  Delete;
  BS_Loc.Init (xNou,yNou);
  Draw;
End;

End.

{Listing 1. BS_Tools - o prima prezentare}

```

```

Unit Bs_Tools;

Interface

Const
SegBWS = $B800; {Color}
{SegBWS = $B000; Mono}
OfsBWS : Word = $0000;

Type
BS_Loc = Object
  X,Y : Integer;
  Procedure Init (xPos, yPos : Integer);
  Function GetX : Integer;
  Function GetY : Integer;
End;
BS_Semn = Object (BS_Loc)
  Semn : Char;
  Attr : Byte;
  Procedure Init (xPos, yPos : Integer;
    Ch : Char; Co : Byte);
  Procedure Draw;
  Procedure Delete;
  Procedure Move (xNou, yNou : Integer);
End;

BS_String = Object (BS_Semn)
  Str : String;
  Procedure Init (xPos, yPos : Integer;
    S : String; Co : Byte);
  Procedure Draw;
  Function Read_BS : String;
  Procedure Delete;
End;

Implementation
Procedure BS_Loc.Init
  (xPos, yPos : Integer);
Begin
  X := xPos;
  Y := yPos;
End;

Function BS_Loc.GetX : Integer;
Begin
  GetX := X;
End;

Function BS_Loc.GetY : Integer;
Begin
  GetY := Y;
End;

Procedure BS_Semn.Init
  (xPos, yPos : Integer;
  Ch : Char; Co : Byte);
Begin
  BS_Loc.Init (xPos,yPos);
  Semn := Ch;
  Attr := Co;
End;

Procedure BS_Semn.Delete;
Begin
  OfsBWS := 2*(X-1) + 160*(Y-1);
  Mem[SegBWS : OfsBWS] := 32;
End;

Procedure BS_Semn.Move
  (xNou, yNou : Integer);
Begin
  Delete;
  BS_Loc.Init (xNou,yNou);
  Draw;
End;

Procedure BS_String.Init
  (xPos, yPos : Integer;
  S : String; Co : Byte);
Begin
  BS_Loc.Init (xPos,yPos);
  Str := S;
  Attr := Co;
End;

Procedure BS_String.Draw;
Var i,s : Integer;
  Ch : Char;
Begin
  s := X;
  For i := 1 To Length (Str) Do Begin
    BS_Semn.Init ((i-1) + s,Y,Str[i],Attr);
    BS_Semn.Draw;
  End;
  X := s;
  Semn := Str[i];
End;

Function BS_String.Read_BS : String;
Begin
  Read_BS := Str;
End;

Procedure BS_String.Delete;
Const SfirsitSemn : Set Of Char = [Chr(0),
  Chr(9),Chr(10),Chr(12),Chr(13)];
Var ij : Integer;
Begin
  i := X;
  j := Y;
  Repeat
    OfsBWS := 2*(i-1) + 160*(j-1);
    If Chr (Mem[SegBWS : OfsBWS]) in
      SfirsitSemn Then Exit;
    BS_Semn.Delete;
    Inc(i);
  Until i = 80;
End;

End.

{Listing 2. BS_Tools -intr-o versiune}
{ imbunatatita, dar mutarea de}
{ siruri de caractere inca nu functioneaza}

```

```

Unit Bs_Tools;

Interface

Uses Crt;

Const
  SegBWS = $B800; {Color}
  {SegBWS = $B000; Mono }
  OfsBWS : Word = $0000;

Type

  BS_Loc = Object
    X,Y : Integer;
    Procedure Init (xPos, yPos : Integer);
    Function GetX : Integer;
    Function GetY : Integer;
  End;

  BS_Semn = Object (BS_Loc)
    Semn : Char;
    Attr : Byte;
    Constructor Init (xPos, yPos : Integer,
      Ch : Char);
    Procedure Draw; Virtual;
    Procedure Delete; Virtual;
    Procedure Move (xNou, yNou : Integer);
  End;

  BS_String = Object (BS_Semn)
    Str : String;
    Constructor Init (xPos, yPos : Integer,
      S : String);
    Procedure Draw; Virtual;
    Function GetStr : String;
    Procedure Delete; Virtual;
  End;

Implementation

Procedure BS_Loc.Init (xPos, yPos : Integer);
Begin
  X := xPos;
  Y := yPos;
End;

Function BS_Loc.GetX : Integer;
Begin
  GetX := X;
End;

Function BS_Loc.GetY : Integer;
Begin
  GetY := Y;
End;

Constructor BS_Semn.Init
  (xPos, yPos : Integer;
  Ch : Char);
Begin
  BS_Loc.Init (xPos, yPos);
  Semn := Ch;
End;

Procedure BS_Semn.Draw;
Begin
  OfsBWS := 2*(X-1) + 160*(Y-1);
  Mem[SegBWS : OfsBWS + 1] := TextAttr;
  Mem[SegBWS : OfsBWS] := Ord(Semn);
End;

```

```

Procedure BS_Semn.Delete;
Begin
  OfsBWS := 2*(X-1) + 160*(Y-1);
  Mem[SegBWS : OfsBWS] := 32;
End;

Procedure BS_Semn.Move
  (xNou, yNou : Integer);
Begin
  Delete;
  BS_Loc.Init (xNou, yNou);
  Draw;
End;

Constructor BS_String.Init
  (xPos, yPos : Integer;
  S : String);
Begin
  BS_Loc.Init (xPos, yPos);
  Str := S;
End;

Procedure BS_String.Draw;
Var i : Integer;
  Ch : Char;
Begin
  s := X;
  For i := 1 To Length (Str) Do Begin
    BS_Semn.Init ((i-1) + s, Y, Str[i]);
    BS_Semn.Draw;
  End;
  X := s;
  Semn := Str[i];
End;

Function BS_String.GetStr : String;
Begin
  GetStr := Str;
End;

Procedure BS_String.Delete;
Const Terminator : Set Of Char = [Chr(0),
  Chr(9), Chr(10), Chr(12), Chr(13)];
Var i : Integer;
Begin
  i := X;
  Repeat
    OfsBWS := 2*(X-1) + 160*(Y-1);
    If Chr (Mem[SegBWS : OfsBWS]) in
      Terminator
    Then Begin
      X := i;
      Exit;
    End;
    BS_Loc.Init(X, Y);
    BS_Semn.Delete;
    Inc(X);
  Until X = 80;
  X := i;
End;

End.

{Listing 3. Scheletul de baza al unei }
{ ierarhii de obiecte }

```

```

Unit Bs_Tools;

Interface

Uses Crt;

Const
  SegBWS = $B800; {Color}
  {SegBWS = $B000; Mono }
  OfsBWS : Word = $0000;

Type

  BS_LocPtr = ^BS_Loc;
  BS_SemnPtr = ^BS_Semn;
  BS_StringPtr = ^BS_String;
  BS_IntrPtr = ^BS_Intr;

  BS_Loc = Object
    X,Y : Integer;
    Procedure Init (xPos, yPos : Integer);
    Function GetX : Integer;
    Function GetY : Integer;
  End;

  BS_Semn = Object (BS_Loc)
    Semn : Char;
    Constructor Init (xPos, yPos : Integer;
      Ch : Char);
    Procedure Draw; Virtual;
    Procedure Delete; Virtual;
    Procedure Move (xNou, yNou : Integer);
    Destructor Done; Virtual;
  End;

  BS_String = Object (BS_Semn)
    Str : String;
    Constructor Init (xPos, yPos : Integer;
      S : String);
    Procedure Draw; Virtual;
    Function GetStr : String;
    Procedure Delete; Virtual;
  End;

  BS_Intr = Object (BS_String)
    I_String : String;
    I_Attr : Byte;
    I_Length : Byte;
    Constructor Init (xPos, yPos : Integer;
      Prompt : String;
      NrMax, Attr : Byte);
    Function ReadString : String;
  End;

Implementation

Procedure BS_Loc.Init (xPos, yPos : Integer);
Begin
  X := xPos;
  Y := yPos;
End;

Function BS_Loc.GetX : Integer;
Begin
  GetX := X;
End;

Function BS_Loc.GetY : Integer;
Begin
  GetY := Y;
End;

```

```
Constructor BS_Semn.Init (xPos, yPos : Integer;
    Ch : Char);
```

```
Begin
    BS_Loc.Init (xPos,yPos);
    Semn := Ch;
```

```
End;
Procedure BS_Semn.Draw;
```

```
Begin
    OfsBWS := 2*(X-1) + 160*(Y-1);
    Mem[SegBWS : OfsBWS + 1] := TextAttr;
    Mem[SegBWS : OfsBWS] := Ord(Semn);
```

```
End;
Procedure BS_Semn.Delete;
```

```
Begin
    OfsBWS := 2*(X-1) + 160*(Y-1);
    Mem[SegBWS : OfsBWS] := 32;
```

```
End;
Procedure BS_Semn.Move (xNou, yNou : Integer);
```

```
Begin
    Delete;
    BS_Loc.Init (xNou,yNou);
    Draw;
```

```
End;
Destructor BS_Semn.Done;
```

```
Begin
End;
Constructor BS_String.Init (xPos, yPos : Integer;
    S : String);
```

```
Begin
    BS_Loc.Init (xPos,yPos);
    Str := S;
```

```
End;
Procedure BS_String.Draw;
```

```
Var i,s : Integer;
    Ch : Char;
Begin
    s := X;
    For i := 1 To Length (Str) Do Begin
        BS_Semn.Init ((i-1)*s, Y, Str[i]);
        BS_Semn.Draw;
```

```
End;
X := s;
Semn := Str[1];
```

```
End;
Function BS_String.GetStr : String;
```

```
Begin
    GetStr := Str;
```

```
End;
Procedure BS_String.Delete;
Const
    Terminator : Set Of Char = [Chr(0),Chr(9),
        Chr(10),Chr(12),Chr(13)];
```

```
Var i : Integer;
Begin
    i := X;
    Repeat
        OfsBWS := 2*(X-1) + 160*(Y-1);
        If Chr (Mem[SegBWS : OfsBWS]) in Terminator
            Then Begin
                X := i;
                Exit;
```

```
End;
Bs_Loc.Init(X,Y);
BS_Semn.Delete;
```

```
Inc(X);
Until X = 80;
```

```
X := i;
End;
Constructor BS_Intr.Init (xPos, yPos : Integer;
    Prompt : String;
    NrMax, Attr : Byte);
```

```
Begin
```

```
    BS_String.Init (xPos,yPos,Prompt);
    I_Attr := Attr;
    I_Length := NrMax;
End;
```

```
Function BS_Intr.ReadString : String;
Const
```

```
    Terminator : Set Of Char =
    [Chr(0),Chr(9),Chr(10),Chr(12),Chr(13)];
    Set1 : Set Of Char = ['.', '~'];
```

```
Var Tasta : Char;
    PosAct, Inceput : Integer;
    EdString : String;
    a : Byte;
```

```
Begin
    BS_String.Draw;
    PosAct := X + Length(Str);
    Inceput := PosAct;
```

```
X := Inceput;
ReadString := "";
EdString := "";
GotoXY (PosAct,Y);
a := TextAttr;
TextAttr := I_Attr;
```

```
Repeat
    Tasta := ReadKey;
    If Tasta = #0 Then Begin
```

```
        Tasta := Readkey;
        Case Tasta Of
            'K':
                If PosAct > Inceput Then
                    PosAct := PosAct - 1;
            'M':
                If PosAct < Inceput + I_Length Then
                    PosAct := PosAct + 1;
```

```
        End;
    Else
        If Tasta In Set1 Then
            If PosAct < Inceput + I_Length Then Begin
                EdString := Copy (EdString,1,PosAct-Inceput) +
                    Tasta +
                    Copy (EdString,PosAct-Inceput + 1,
                        Length(EdString));
                PosAct := PosAct + 1;
```

```
            End;
            GotoXY (PosAct,Y);
            Str := EdString;
            BS_String.Draw;
            Until Tasta In Terminator;
            ReadString := EdString;
            TextAttr := a;
```

```
End;
End.
{Listing 4. BS_Tools - ultima varianta }
```

```
Program List16;
Uses Crt,BS_Tools;
```

```
Var C : BS_IntrPtr;
    Attr : Byte;
    S : String;
```

```
Begin
    C := Crt;
    New (C,Init(12,12,'Introduceti, va rog : ',12,111));
    S := C^.ReadString;
    C^.Move(20,20);
    Dispose (C,Done);
End.
```

```
{Listing 5. Cu BS_Tools introducerea de siruri devine simpla}
```

Protecție prin parolă

O parolă eficientă ar trebui să conțină mai multe caractere, să nu fie prea simplă și să fie schimbată la intervale regulate. Nume, date de naștere sau combinații "fără sens" de regulă nu ridică probleme deosebite în fața unor "hackeri" împătimiți. Acronimele sînt recomandabile, ale unui vers sau cîntec binecunoscut. De exemplu:

u(n')	m(ielule)
t(e)	?
d(uci)	l(a)
t(u)	p(ășune)
	d(omnule)

În listingul Unlock.asm, această secvență e folosită pentru a "dezăvorî" tastatura. Secvența e prelucrată, rezultatul prelucrării fiind un cod ce se memorează în parolă. Dacă ați greșit cumva vreun caracter la tastarea parolei, puteți oricînd reinițializa tastînd <ESC>. Din valoarea compilată, este foarte greu de a deduce o secvență care să o genereze (inclusă în listing numai din comoditate). Unlock.asm e folosit ca program pereche cu driverul rezident Lockdrv.asm. Acesta e cel care "zăvorește" tastatura, nepermițînd nici un acces la ea pînă ce nu se rulează Unlock. Cum cea mai frumoasă parolă nu face doi bani dacă fișierul inițial de comenzi, Autoexec.bat, poate fi întrerupt cu <Ctrl-C> sau <Ctrl-Break>, zăvorîrea tastaturii din Config.sys, face imposibilă intruziunea cuiva care nu știe parola. Desigur, Unlock ar trebui inclus în Autoexec.bat - altfel, nici Dvs. nu mai aveți acces la tastatură...Așa cum vă e prezentat, Lockdrv.sys poate fi folosit de fapt ca un cadru pentru a transforma programe rezidente în memorie și care preiau anumiți vectori de întrerupere, în device-driver. Avantajul metodei: prin modul în care un device-driver se integrează în sistem, el ocupă puțin loc. Dezavantajul: spațiul ocupat nu mai poate fi eliberat. Compilarea se poate face de ex. cu comenziile

```
tasm lockdrv
tasm unlock
tlink /t /x lockdrv, lockdrv.sys
tlink /t /x unlock
```

(I.F.)

```
1 title #unlock.asm
2 ; inceput program
3 code segment word
4 assume cs:code, ds:code
5 org 100h
6 start: push bx
7 push cx
8 push dx
9 push ds
10 push es
11 ; deschide canal pentru #Lock#
12 mov ah,3dh
13 mov al,0
14 mov dx,offset DEVname
```

```
15 int 21h
16 mov DEVhan, ax
17 ; citește 8 bytes din #Lock#
18 mov ah,3fh
19 mov bx,DEVhan
20 mov cx,8
21 mov dx,offset INT1buf
22 int 21h
23 push ds
24 mov ax,2510h
25 lds dx,INT1buf
26 int 21h
27 pop ds
28 push ds
29 mov ax,2516h
30 lds dx,INT2buf
31 int 21h
32 pop ds
33 mov ah,3eh
34 mov bx,DEVhan
35 int 21h
36 mov dx,offset message
37 mov ah,9h
38 int 21h
39 ; initializare
40 init: mov dx,0
41 ; numarul de taste
42 mov cx,count
43 ; introducere scancod-uri
44 next: mov ah,0h
45 int 16h
46 cmp ah,1
47 jz init
48 add dx,ax
49 ror dx,1
50 loop next
51 cmp dx,pass
52 jnz init
53 mov ax,3h
54 int 10h
55 pop es
56 pop ds
57 pop dx
58 pop cx
59 pop bx
60 mov ah,4ch
61 int 21h
62 ; zona de date
63 INT1buf label dword
64 INT1bufo dw ?
65 INT1bufs dw ?
66 INT2buf label dword
67 INT2bufo dw ?
68 INT2bufs dw ?
69 DEVhan dw ?
70 DEVname db "#Lock#",0
71 count dw 11 ; numarul de taste
72 ;parola:"utdt,m?lp,d"
73 pass dw 7938h
74 message db "keyboard is locked - please unlock it ...$"
75 code ends
76 end start
77
```

UNLOCK.ASM

```

1 title lockdrv.asm
2 ;request-header-structure
3 ; pointer to request header
4 RQptr equ ds:[bx]
5
6 RQhdr struc
7 db ?
8 db ?
9 ; command code
10 cmd db ?
11 ; status
12 stat dw ?
13 db 8 dup(?)
14 db ?
15 ; adresa de transfer
16 adr equ this dword
17 adro dw ?
18 adrs dw - ?
19 ; contor
20 cnt dw ?
21 RQhdr ends
22
23 ; start program
24 code segment word
25 assume cs:code
26 org 0h
27 ; device header
28 ; pointer to next driver
29 dd -1
30 ; attribute for char-driver
31 dw 8000h
32 ; strategy routine
33 dw strat
34 dw inter
35 db "#LOCK# "
36
37 ; data
38 RQadr label dword
39 RQadro dw ?
40 RQadrs dw ?
41 INTptr dw ?
42 INT1bufo dw ?
43 INT1bufs dw ?
44 INT2bufo dw ?
45 INT2bufs dw ?
46 count dw ?
47 ; new interrupt routine
48 newint: iret
49 strat proc far
50 mov RQadro, bx
51 mov RQadrs, es
52 ret
53 strat endp
54
55 ; rutina de intrerupere rezolva de fapt sarcinile
56 driver-ului
57 inter proc far
58 push ax
59 push bx
60 push cx
61 push dx
62 push ds
63 push es
64 push si
65 push di
66 push bp
67 lds bx, RQadr
68 mov al, RQptr.cmd
69 cmp al, 0
70 je init
71 cmp al, 4
72 je read
73 ; eroare: comanda inacceptabila

```

```

73 mov ax, 8103h
74 mov RQptr.stat, ax
75 exit: pop bp
76 pop di
77 pop si
78 pop es
79 pop ds
80 pop dx
81 pop cx
82 pop bx
83 pop ax
84 ret
85 inter endp
86 ; citirea
87 read: les di, RQptr.adr
88 mov cx, RQptr.cnt
89 jcxz exit
90 les di, RQptr.adr
91 mov si, INTptr
92 mov al, cs:[si]
93 stosb
94 inc INTptr
95 inc count
96 cmp count, 8
97 jne no
98 mov count, 0
99 mov INTptr, offset INT1bufo
100 no: mov ah, 1
101 mov RQptr.stat, ax
102 jmp exit
103 ; initialize driver
104 init: mov ax, cs ; permisia o singura data
105 mov ds, ax
106 assume ds: code
107 mov word ptr [black], 9090h
108 mov ax, 3510h
109 int 21h
110 mov INT1bufo, bx
111 mov INT1bufs, es
112 mov ax, 3516h
113 int 21h
114 mov INT2bufo, bx
115 mov INT2bufs, es
116 mov count, 0
117 mov INTptr, offset INT1bufo
118 mov dx, offset newint
119 mov ax, 2510h
120 int 21h
121 mov dx, offset newint
122 mov ax, 2516h
123 int 21h
124 ; sfirsit parte rezidenta #LOCK#
125 mov dx, offset init + 0fh
126 mov cl, 4
127 shr dx, cl
128 mov ax, cs
129 add dx, ax
130 lds bx, RQadr
131 mov RQptr.adro, 0
132 mov RQptr.adrs, dx
133 mov ah, 1
134 mov RQptr.stat, ax
135 jmp exit
136 ; sfirsit program
137 code ends
138 end

```

LOCKDRV.ASM

```

1 unit comuzica; {$D-,L-,S-}
2 interface uses dos,crt;
3 type tiptabnote = array[0..32759,0..1] of byte;
4 var nrnote:word; tabnote: ^ tiptabnote;
5 procedure cintec;
6
7 implementation
8 var salv:pointer;indice,contor,indnot:word;
9
10 const valnote:array[1..84] of word =
11   ( 65, 69, 73, 78, 82, 87, 92, 98, 104, 110, 116, 123,
12     131, 139, 147, 156, 165, 175, 185, 196,
13     208, 220, 233, 247,
14     262, 278, 294, 312, 330, 350, 370, 392,
15     416, 440, 466, 494,
16     524, 556, 588, 624, 660, 700, 740, 784,
17     832, 880, 932, 988,
18     1048, 1112, 1176, 1248, 1320, 1400, 1480, 1568,
19     1664, 1760, 1864, 1976,
20     2096, 2224, 2352, 2496, 2640, 2800, 2960, 3136,
21     3328, 3520, 3728, 3952,
22     4192, 4448, 4704, 4992, 5280, 5600, 5920, 6272,
23     6656, 7040, 7456, 7904);
24
25 procedure tratimp; interrupt;
26 begin
27   if contor = 0 then
28     begin
29       nosound;
30       indice := (indice + 1) mod nrnote;
31       indnot := tabnote ^ [indice, 0];
32       if (indnot0) and (indnotU) then
33         sound(valnote[indnot]);
34       contor := tabnote ^ [indice, 1] and 127
35     end;
36   dec(contor)
37 end;
38
39 procedure cintec;
40 begin
41   if nrnote > 0
42   then
43     begin
44       getintvec(28, salv); indice := nrnote - 1;
45       contor := 0; setintvec(28, @tratimp)
46     end
47   else
48     begin
49       nosound;
50       setintvec(28, salv)
51     end
52   end;
53 end.

```

MUZICA.PAS

```

1 program mtest; uses muzica;
2 const note:array[0..29,0..1] of byte =
3   ((3,8),(3,8),(4,8),(5,8),(5,8),(4,8),(3,8),(2,8),
4     (1,8),(1,8),(2,8),(3,8),(3,12),(2,4),(2,16),
5     (3,8),(3,8),(4,8),(5,8),(5,8),(4,8),(3,8),(2,8),
6     (1,8),(1,8),(2,8),(3,8),(2,12),(1,4),(1,16));
7 begin
8   nrnote := 30; tabnote := @note; cintec;
9   (* orice secventa de instructiuni *)
10  write('Tastati Enter pentru terminare');
11  readln; nrnote := 0; cintec
12 end.

```

MTEST.PAS

Vă place muzica ?

Prezentăm în continuare un exemplu de utilizare în Turbo Pascal a întreruperii de timp 28 (1Ch). Acest exemplu ilustrează posibilitatea de a executa în paralel două acțiuni: - prima, funcțiile propriu-zise ale programului; - a doua, derularea unei melodii în paralel cu execuția programului (ceea ce există la majoritatea programelor de joc).

În tabloul VALNOTE (programul MUZICA.PAS) sînt date frecvențele pentru următoarele note: DO RE MI FA SOL LA SI DO.

În sursa program MTEST.PAS:

În tabelul NOTE este dată partitura. Fiecare notă este precizată prin două numere: primul este indicele în tabelul de frecvențe, al doilea este durata notei în unități de 55 ms.

Bibliografie: Sistemul de operare DOS. Ghidul programatorului - editat de asociația Micro-Informatica Cluj-Napoca.

mat. Ioan Cozac

Ștergere rapidă

Deseori, se dorește ștergerea completă a unei dischete, fără însă a o reformată. Pentru aceasta, trebuie șterse toate fișierele din toate subdirectoarele - ba probabil printre ele mai sînt și fișiere protejate la scriere, care trebuiesc întii "deprotejate" pentru a putea fi șterse. Cu programul QFORMAT, o dischetă deja formatată poate fi rapid "reformatată". Programul șterge pur și simplu cele două tabele FAT precum și directorul rădăcină, după care suprascrie toate sectoarele dischetei. După ce s-a operat o astfel de ștergere, tot ce a fost pe dischetă este definitiv și iremediabil dispărut. La startare, programul întreabă în ce unitate se va lucra (A sau B). Alte unități sînt blocate; o ștergere accidentală a harddisk-ului este exclusă. Funcțional, programul este echivalent cu WipeDisk din setul de utilitare Norton.

(I.F.)

```

1
2   Const Maxsec = 124;
3   Var Buf:Array[1..512*MaxSec] of Byte;
4   Sector, Drive, Count, i : Word;
5   Boots, Fats, FatSecs, DirSecs, Rest: Word;
6   ch: Char;
7
8   Procedure ReadSectors;
9   Function RSector: Boolean;
10  begin
11    InLine ($55/$a1/Drive/$8b/$0e/Count/
12      $8b/$16/Sector/$bb/Buf/$cd/
13      $25/$72/$04/$b0/$01/$eb/$02/
14      $b0/$00/$9d/$5d/$88/$46/$ff);
15  end;

```

```

16 begin
17 if Rsector = False then begin
18     WriteLn ('Eroare la citire !!!');
19     Halt (255);
20 end;
21 end;
22
23 Procedure WriteSectors;
24 Function Wsector: Boolean;
25 begin
26     InLine ($55/$a1/Drive/$8b/$0e/Count/
27             $8b/$16/Sector/$bb/Buf/$cd/
28             $26/$72/$04/$b0/$01/$eb/$02/
29             $b0/$00/$9d/$5d/$88/$46/$f);
30 end;
31 begin
32 if WSector = False then begin
33     WriteLn ('Eroare la scriere !!!');
34     Halt (255);
35 end;
36 end;
37
38 begin
39 WriteLn (' *** Stergere discheta ***');
40 repeat
41     Write ('Unitatea (0 ptr. A, 1 ptr. B): ');
42     ReadLn (Drive);
43 until (Drive = 0) or (Drive = 1);
44 Write ('Sterg intr-adevar ? (d/n): ');
45 ReadLn (ch);
46 if UpCase (ch) = 'D' then begin
47     Sector := 0;
48     Count := 1;
49     ReadSectors;
50     Boots := Buf[15] + Buf[16]*256;
51     Fats := Buf[17];
52     FatSecs := Buf[23] + Buf[24]*256;
53     DirSecs := (Buf[18] + Buf[19]*256) div 16;
54     Rest := Buf[20] + Buf[21]*256;
55     Sector := Boots;
56     Count := FatSecs;
57     ReadSectors;
58 for i := 4 to FatSecs*512 do Buf[i] := 0;
59 for i := 1 to Fats do begin
60     WriteSectors;
61     Sector := Sector + FatSecs;
62 end;
63     Count := DirSecs;
64 for i := 1 to DirSecs*512 do Buf[i] := 0;
65 WriteSectors;
66     Sector := Sector + DirSecs;
67 for i := 1 to 512*MaxSec do Buf[i] := 241;
68     Rest := Rest-Sector;
69 while Rest > 0 do begin
70     if Rest > MaxSec then Count := Rest
71     else Count := Maxsec;
72     WriteSectors;
73     Sector := Sector + Count;
74     Rest := Rest-Count;
75 end;
76 WriteLn ('S-a sters totul de pe discheta !');
77 end;
78 end.
79
80

```

QFORMAT.PAS

ChDir confortabil

Mai ales la structuri de directoare stufoase, trecerea dintr-un director într-altul este deseori o operație care consumă timp și efort din partea utilizatorului. Programul CHDIR.PAS ușurează această operație. La lansare, i se dă ca parametru o parte a numelui din calea la care ar trebui să se ajungă. În continuare, programul parcurge toate subdirectoarele de pe unitatea curentă care conțin în nume șirul specificat ca parametru. Dacă se găsește un astfel de subdirector, numele complet al căii respective este afișat pe ecran. Dacă acesta este subdirectorul căutat, se apasă <RETURN> și schimbarea de cale se efectuează. La acționarea oricărei alte taste, se caută alte subdirectoare care se conformează criteriului specificat.

(I.F.)

```

1     {$M 64000, 0, 0}
2     uses Dos, Crt;
3     Var Ds: DirStr;
4         Ns: NameStr;
5         Es: ExtStr;
6     procedure ChangeDir (path: string);
7     var S: SearchRec;
8     begin
9         FindFirst (path + '*.*', $12, S);
10        while DosError = 0 do begin
11            if (S.Attr and $10 = $10) and
12                (S.Name[1] <> '.')
13            then begin
14                if Pos(Ns + Es, S.Name) > 0 then begin
15                    GotoXY (1, WhereY); ClrEol;
16                    Write ('Move to: ', path + S.Name);
17                    if ReadKey = #13 then begin
18                        Chdir (path + S.Name);
19                        Halt(0);
20                    end;
21                end;
22                ChangeDir (path + S.Name + '\');
23            end;
24            FindNext (S);
25        end;
26    end;
27
28    begin
29        Fsplit (FExpand (ParamStr (1)), Ds, Ns, Es);
30        Ds := Ds[1] + '\';
31        ChangeDir (Ds);
32        GotoXY (1, WhereY); ClrEol;
33        Write ('No directory change!');
34    end.
35

```

CCD.PAS

Poșta redacției

"... noi avem nevoie de foarte multă literatură de specialitate. Citind o carte de teoria proiectării sistemelor informaționale (apărută la o editură românească, bineînțeles) am ales o listă de denumiri de cărți de care aș avea nevoie în primul rând. Lista o anexează, subliniind poate cele mai necesare cărți."

Ovidiu Clubotă, str. Florilor 6/2, apt. 35,
277068 Chișinău, Republica Moldova,
URSS

1. Ionescu C., Sabău Gh., Cojocaru A., Avram V. - Baze de date relaționale, Lito A.S.E., 1985
2. Drăghici M. - Îndrumar metodologic pentru proiectarea schemei conceptuale a bazei de date, ICSIT-TCI, 1986
3. Bara I., Răureanu M. - Experiment privind arhitectura unui sistem inteligent de gestiune a bazelor de date relaționale, ICSIT-TCI, 1986
4. Gh. I. Pisău - Elaborarea și implementarea sistemelor informatice
5. V. Penescu ș.a. Fișiere, baze și bănci de date
6. Pescaru V., Catona I., Duță I., Popescu C., Stratan I. - Fișiere, baze, bănci de date, Ed. tehnică 1976
7. Dumitrache V., Unghianu M. - Băncile de date, Ed. științifică și enciclopedică, 1982
8. V. Baltac, A. Davidoviciu, C. Mașec ș.a. - Sisteme interactive și limbaje conversaționale
9. P. Constantinescu ș.a. - Sisteme informatice, modele ale conducerii

Ne-ar părea tare rău ca această scrisoare să treacă prin conștiința noastră ca un simplu fapt divers. Este - și trebuie să fie - mult mai mult. Îi rugăm pe toți cititorii noștri să recepționeze acest apel - chiar dacă scrisoarea a venit pe adresa noastră. Și nu numai de la cititorul nostru (care se întâmplă să fie primul nostru corespondent de dincolo de Prut). Ci de la toți cei ca el. Faptul că speranțele cititorului nostru au găsit, prin intermediul revistei, o adresă concretă, nu ne scoate din anonimat. Poate că cineva se va apuca să facă o treabă organizată. Dar să nu așteptăm - sîntem convinși că vor fi bine utilizate

În numărul viitor:

- 10 programe antivirus
- DTP și fonturi
- DOS extindere

toate cărțile ce vor fi trimise. Direct la Chișinău (preferabil!), sau înfii nouă.. Cei care consideră că pot trimite cărți de specialitate, sînt rugați să o facă. Mulți dintre noi au cărți ce pot desigur prezenta interes - și de care ne putem dispensa.. Sigur că unele sînt învechite și nu mai au valoare - dar există și multe cărți care își păstrează valoarea în timp. Nu cred că există multe biblioteci de întreprindere care să nu aibă cărți pe care s-au așternut multe degete de prof. Și care ar putea prezenta o reală valoare pentru colegii noștri de dincolo de Prut, prin simplul fapt că tratează niște probleme de specialitate în limba română. Să fie chiar imposibil ...? Puțin sceptici - sincer, nu credem că va fi extrem de multă muncă de prestat - ne oferim și noi ca 'mediatori'. Cărțile ce se vor aduna ca donații la noi promitem să le ducem într-un fel sau altul la cei cărora le sînt destinate. Și nu vedem nici un motiv ca printre cei care fac donații să nu se afle și edituri, instituții, întreprinderi.

Apropo de instituții - o trecere în revistă a fișierului nostru de abonați relevă un fapt puțin surprinzător: marea majoritate a celor care au făcut abonamente la revista noastră sînt persoane particulare. Cîteva firme particulare, cîteva (puține) Centre de calcul, cîteva instituții de stat - și alții! Dragi cititori,, știm că uneori este incomod, și că uneori biblioteca întreprinderii la care lucrați funcționează prost sau nu funcționează de loc. Dar mai știm și că sînt foarte puțini directori sau contabili șefi care să citească revista noastră. Neavînd solicitări exprese în acest sens de la subalterni, nici unul dintre aceștia nu va face abonamente la ilustra necunoscută revistă! Iar bibliotecara - dacă există - este, probabil, cel puțin încurcată dacă trebuie să decidă dacă merită sau nu să facă abonament. Sigur că revista e scumpă - sîntem conștienți de acest lucru, singura alternativă ar fi să renunțăm. Și nu vrem să ne dăm bătăuți. O variantă posibilă ar fi creșterea tirajului - dar pentru aceasta, difuzarea ar trebui să se întîmple cu totul altfel decît se întîmplă de fapt. Nu putem ieși din cercul vicios în care ne învîrtim - decît cu ajutorul D-voastră!

Revista e scumpă pentru că tirajul e mic; tirajul e mic pentru că riscul nedifuzării corecte, laolaltă cu prețul ridicat al hîrtiei și tipăririi, nu ne permit pur și simplu să ieșim într-un tiraj mai mare. Abonamentele sînt o certitudine - cu ele în spate, tirajul poate crește. Și dacă abonamentul este scump - poate că pentru întreprinderea sau biroul în

care lucrați nu este totuși un efort semnificativ!

Nu vrem acte de caritate - dacă nu merită să supraviețuim, foarte bine, vom dispărea, ca alții alții. Poate am calculat greșit, poate că ne-am întins mai mult decît ne este plapuma, poate că pur și simplu nu ne pricepem.

Dar dacă revista vă place - cît de puțin - atunci vă rugăm să faceți acel mic efort de a ne face cunoscute și altora. Noi facem revista - dar am vrea ca ea să fie un mic și a Dvs. De fapt, fără Dvs., ea nu ar mai avea sens.

AJ Dvs.,
ing. Iosif Fettich

Wanted !! Reward!!

Am primit bani pentru abonamente de la următoarele persoane sau adrese, care au uitat să ne comunice adresa, respectiv numele:

- MX Consulting Group București
- Gliga Jenica, Cluj
- Nemeș Corina
- Miron Cornelia, Arad
- Sibiu, str. Bîrsei nr. 10, bl.39, ap.17

Cine îi cunoaște, e rugat să îi determine să ia legătura cu redacția, pentru a completa datele lipsă.

Mica publicitate

Vînd HC-85, șah-computer Mephisto, joystick-PC, unități floppy 3,5", dischete 3,5", socluri Tulipe, diferite componente. Negrea Dan, str. 9 mai, nr. 3, sc. A, apt. 13, 5500 Bacău, tel. 931/44450.

Ofer: jocuri, utilitare și documentații pt. calculatoare Spectrum. Szentes Attila, str. Pescarilor, bl. 6, sc. A, apt. 9, 4400 Bistrița.

Cumpăr microdrives cart-ridges (cartușe cu bandă) pentru calculatorul Sinclair QL sau Spectrum și software pentru QL. Tel. 90/754782.

Important !

Doriți să aveți garanția că nu veți pierde nici un număr din "if" ?
Încheiați un abonament folosindu-vă de talonul alăturat, sau
adresați-vă difuzorilor noștri autorizați:

Baia Mare

Omni Computer
(Cudalbu Octavian)
tel. 36433

Brăila

Teodorescu Constantin
tel. 47203

București

Dracea Tudor
tel. 210340

Cluj Napoca

- AbMod
- Magazin Hardware-Software
b-dul 22 Decembrie nr. 135
 - Sediul central
tel. 111090

Craiova

Stătescu Paul
tel. 48117

Deva

Toma Tudor
tel. 10144, 10234

Onești

Brumă Cornel
tel. 24222-236

Oradea

AbMod
tel. 60278, 44476

Rădăuți

Burciu Florin
tel. 63483

Sighișoara

Beer Mihai
tel. 73638

Timișoara

Tomoroga Mircea
tel. 77422

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. 172, O.P. 1

RO - 4300 Tîrgu Mureș

**Lista faxurilor autorizate de Ministerul Comunicațiilor
pentru a fi conectate în rețeaua națională de telecomunicații**

Listă aprobată la 5 iunie 1991

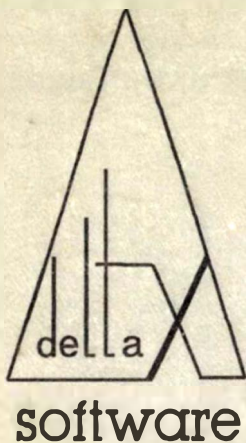
ADLER ROYAL FX 835	INFOTEC 6112	PANAFAX UF270M	SEL-ITT 3532
AEG OLYMPIA OLIFAX 330	INFOTEC 6500	PANAFAX UF400	SEL-ITT 3534
AGORIS 200 EGT	INFOTEC 6510	PANAFAX 420	SHARP FO 50
ALCATEL 3602	INFRAFAX 9600SF	PANASONIC KX F50	SHARP FO 215
ALCATEL CT200	ITEC FN 10	PANASONIC KX F80	SHARP FO 220
ALTECHAF 2500	ITI IF 510	PANASONIC KX F90	SHARP UX 50
AMEFON AF 1000	ITT 3531	PANASONIC KX F90 B	SHARP UX 101
AMCO TFX 200	KONIKA 120	PANASONIC KX F100	SHARP UX 110
AMSTRAD FX 5000	KONIKA 150	PANASONIC KX F120B	SHARP UX 160
AMSTRAD FX 9600 T	KONIKA UBIX FT500	PANASONIC KX F3550BS	SHARP UX 191
ARFAX 2000	LANIER 115 AD	PHILIPS 3050	SHARP 1800
ASCOM FAX 150	LANIER 2225	PHILIPS 3060	SIEMENS HF 2021
ATFAX 9600 FAX CARD	MERCEDES X9	PHILIPS 3090 FAX LINE	SIEMENS HF 2301
AUDIOVOX AFX 1000	META QX 101	PHILIPS 3100	SIEMENS HF 2302
AUDIOVOX AFX 2000	MINOLTA 250	PITNEY-BOWES 7100	SIEMENS HF 2303
AUDIOVOX AFX 3000	MINOLTA 160/161	PITNEY-BOWES 7800	STABOFAX 10
AVATEX EFAX 88	MINOLTA 360/361	PITNEY-BOWES 8050	STEMARK STF 23
AVATEX EFAX 110	MITA TC30	PITNEY-BOWES 8000	TANDYFAX 1000
BOS BTT 1010	MITA TC40	POLY FOCUS PF 2200	TELECOPYMAT 200
BROTHER 210	MITSHUBISHI FA60Z	PRIORITY FAX 2000	TENOFAX 3 TELENORMA
BROTHER 305	MURATA F 1	REGMA TC 20	THOMFAX 3605 ALCATEL
CANON FAXPHONE 8	MURATA F 30	REX ROTARY 6003 TC	TOSHIBA 3400
CANON FAXPHONE 15	MURATA M 900	RICOH FAX 07	TOSHIBA BD 7610
CANON 80	MURATA M 1100	RICOH FAX 08	TOSHIBA TF111
CANON 110	MURATA M1-1600	RICOH FAX 20	TOSHIBA TF112
CANON 120	MURATA M 1800	RICOH FAX 25	TOSHIBA TF152
CANON 230	NASHUA F810	RICOH FAX 60	TOSHIBA TF211
CANON 260	NASHUA INFOTEC 6500	RICOH RF 860	TOSHIBA TF251
CANON 270	NEFAX 2 NEC	ROADSTAR SF2010	TRIUMPH ADLER FX 905
CANON 270S	NEFAX 10	SAGEM SAFAX LAUREATT	TRIUMPH ADLER TA FX 910
CANON 350	NEFAX 11	SAMSUNG SF1000	UNDERWOOD UDF 71
CANON 410	NEFAX 18	SAMSUNG FX1010	UNIFAX 1412 NEC
CANON 450	NEFAX 310	SAMSUNG 2010	UTAX F130
CANON 510	NEFAX 4100F	SAMSUNG SF2020	VALIDIAN PF 505
CANON 520	NEFAX 4500	SANFAX 3	XEROX 295
CANON 710	NIPPON NP 90	SANFAX 225	XEROX 7007
CANON 730	NISSEI FAX 53	SANFAX 330	XEROX 7009
CANON 750	NISSEI FAX 303	SANYO SANFAX 100	XEROX 7010
CITIZEN JFX 1961	NISSEI FAX 305	SCHNEIDER FAX SPF-100	XEROX 7020
DATA FAX 300 SERIES	NISSEI FAX 320	SCHNEIDER FAX SPF-101	ZICOM F900
DEFAX 2 DEVELOP	NITSUKO FXE500	SEL FAX 10	ZX 1896 MODEM FAX
DIAFAX 001 SF 1000	OLIVETTI OFX 325	SEL FAX 20	
EFAX EF 110	OLIVETTI OFX 530		
EPSON Priority FAX 1000	OKIFAX OF7		
FAX LINE	OKIFAX OF8		
FAXLINE 3090 PHILIPS	OKIFAX OF8 M		
FUJITSU DEX 80	OKIFAX OF17 H		
FUJITSU DEX-TEN 3125	OKIFAX OF17 L		
FUNAI FA-X2000	OKIFAX OF38		
GECO UA 2100	OKIFAX OF110		
GOLDSTAR LF110	PACKARD BELL PB FAX 100		
GOLDSTAR LF 330	PACKARD BELL FAX 400		
GOLDSTAR LF 510	PANAFAX 135		
HARRIS 3M 111	PANAFAX UF120		
HARRIS 3M 115	PANAFAX UF130		
HARRIS 3M 2110	PANAFAX UF150		
HIFAX 17E	PANAFAX UF160		
INFOTEC 6110	PANAFAX UF250		

"Echipamente INFIA S.R.L."

realizează

**Dispozitiv pentru scriere / ștergere
memorii EPROM**

Comenziile se primesc pe adresa:
Cod 76.600, O.P. București 75, C.P. 75-37
Relații la telefon: 90-86.45.53 după ora 19,00



Societatea de Informatică **DELTA-X S.R.L.**
Oradea, P-ța Independenței Nr.47 A3/23
Telefon: 991-34257, 51804

Vă oferă la prețuri avantajoase următoarele produse program generalizabile pe calculatoare compatibile IBM PC XT/AT:

- 1. Bal-Con** - evidența contabilă generală
- 2. Mi-Fi** - evidența și calculul amortizării mijloacelor fixe
- 3. Ge-Ma** - gestiunea și evidența contabilă a valorilor materiale (materiale, obiecte de inventar, produse finite).

Produsele enumerate, sînt puternic parametrizate avînd un înalt grad de generalizare și au strînse legături între ele, utilizînd colecții comune de date, transferînd informații între ele. Evidența contabilă Bal-Con, reprezintă "vîrfurile piramidei" care pune la dispoziția celorlalte produse planul de conturi contabile, respectiv primește date (note contabile generate automat) din partea acestora. Sistemul este deschis, nefiind obligatorie utilizarea produselor program enumerate împreună, respectiv este facilitată conectarea altor aplicații la evidența contabilă (salarii, desfacere, etc.). Utilizarea și operarea este facilă prin meniurile, mesajele explicative concrete apelabile în toate fazele de prelucrare, și autodocumentare.

Prin programele de instalare livrate, utilizatorul poate configura produsele la caracteristicile echipamentului utilizat.

Versiunile finalizate ale celor trei produse menționate reflectă legislația economică și reglementările metodologice în vigoare și vor fi aliniate și în viitor la eventualele schimbări legislative sau metodologice care vor apare.

4. dAccess - accesare multiuser a bazelor de date compatibile dBASE.

Permite culegerea, prelucrarea și vizualizarea datelor în regim multiuser, de la 1 la 8 terminale, conectate pe căile seriale COM1 -COMB și de la tastatura/monitorul calculatorului.

Oferă în regim multiuser, prin taskuri independente, majoritatea prelucrărilor permise de sistemele tip dBASE, definirea acestor taskuri făcîndu-se într-un mod simplu, familiar utilizatorilor.

Avînd în vedere, că majoritatea utilizatorilor de tehnică de calcul din economia noastră, dispun de terminale simple în bună stare de funcționare (VDT-52S, etc.), prin utilizarea acestui produs achiziționarea unor calculatoare IBM-PC compatibile în configurația obișnuită (2 căi seriale) nu exclude utilizarea eficientă a acestor terminale devenite disponibile.