if

CALCULATOARE PERSONALE

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

6 / 91

(9)

57 Lei

Staţiile de lucru Next

Memento: *Specificaţiile XMS*

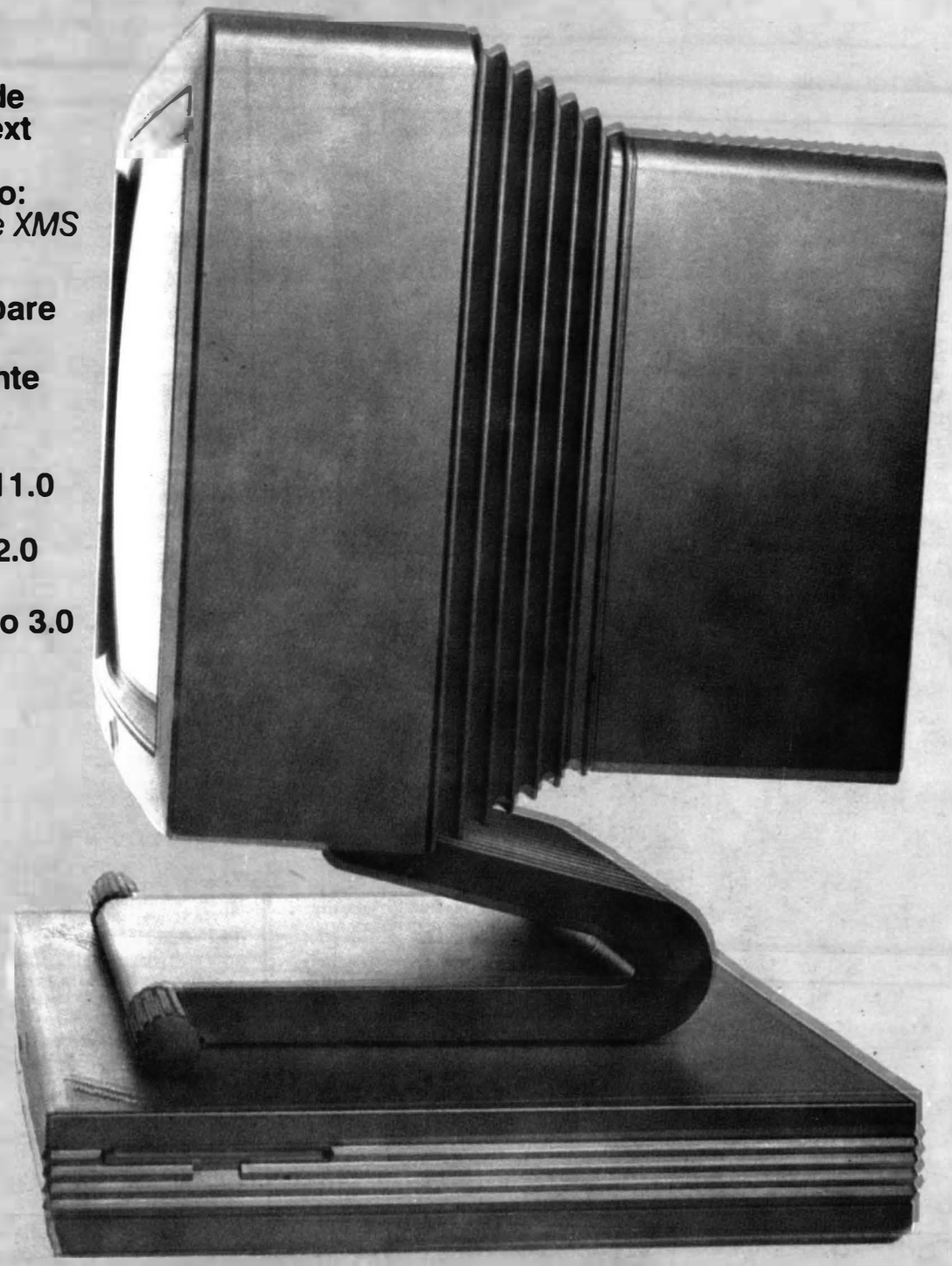
Cod de bare

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

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. Motilor 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

if

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

Director: ing. Dumitru Dunca

Redacţia:

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

Colaboratori externi:

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

Tipariul: tipografia Tîrgu Mureş
Tipografi:
Halaţiu Mihai,
Cormoş Mircea

Revista apare lunar.

Preţ: 57 lei

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

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

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

Adresa şi telefonul redacţiei:
Micro ATCI,
RO-4300 Tîrgu Mureş,
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.

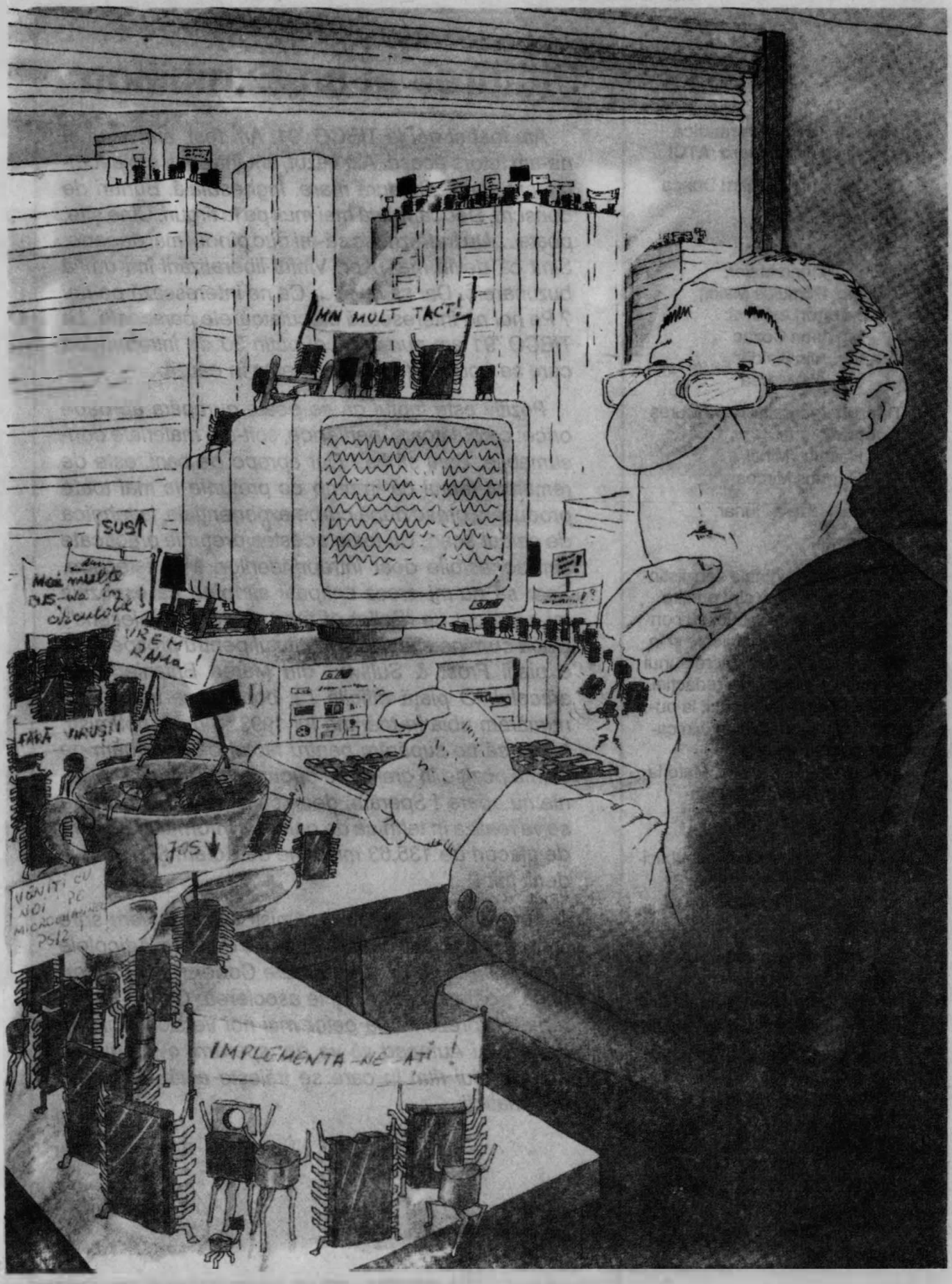
Pictures at an Exhibition

Am fost şi noi la TIBCO '91. Am fost, am văzut şi ne-am întors acasă. Am văzut, am întrebat, am ascultat, am simţit. Căldură mare. Înghesuială. Bunuri de consum. Deocamdată mai mult pe la tîrguri. Cine ştie, poate ... Nu îndrăznesc să-mi duc gîndul mai departe. Simt că devin mai uşor. 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 puţin 30 de întreprinderi care se ocupau (şi) 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 preţurile la mai toate produsele cresc după curbe exponenţiale, la tehnica de calcul scad. Cu toate acestea preţurile practicate sînt accesibile doar întreprinderilor. Mai aşteptăm. Sper să nu-mi creez antipatii afirmînd că expoziţia mi-a evocat un "Ballet of the unhatched chickens". După cum ne sfătuieşte institutul pentru prospectare a pieţii Frost & Sullivan din Marea Britanie, (vezi articolul "O piaţă dificilă !"), bobocii va trebui să-i numărăm abia în toamna lui 1993. Sîntem optimişti. În loc să ne supărăm pentru faptul că ne situăm pe ultima poziţie în grafic, ne bucurăm că figurăm. Albania nu apare ! Sperăm, deci, că în 1993 în România se va realiza în tehnica de calcul şi informatică o cifră de afaceri de 135,63 milioane de dolari. Şi mai mult decît atît !

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

Romulus Maier


MAN MULT TACT!
SUS↑
JOS↓
IMPLEMENTA-NE ATI!

Cuprins

PC Tools al VII-lea

Versiunea 7.0 a lui PC Tools a început o nouă rundă în confruntarea pentru titlul de cel mai bun program utilitar. Noile atu-uri: suport reţea şi suport Windows.

Săptămînile trecute, Central Point Software, din SUA, a anunţat a şaptea versiune a programului utilitar PC Tools. Noua versiune conţine, după declaraţiile lui Corey Smith, preşedintele CPS, 80 de detalii îmbunătăţite şi 10 module noi, care, ca de obicei, pot fi apelate de sub o suprafaţă utilizator unitară. PC Tools devine primul program utilitar care poate fi rulat atît sub MS-DOS cît şi sub Windows şi a cărui mod de operare este identic în ambele medii.

Printre cele mai importante noutăţi se numără un program de transfer de date, asemănător cu Laplink, cu ajutorul căruia, prin intermediul unui modem, a unei reţele, sau a unui cablu, poate fi comandat un calculator de la distanţă. Transferul de date între cele două calculatoare este posibil cu ajutorul programului Commute. Posibilitatea lucrului în reţea joacă un rol important atît în cazul lui Commute cît şi în cazul celorlalte module. Astfel "Undelete" poate recupera acum şi fişiere aflate pe un server Novell, iar Filefind utilizează atributele extinse de reţea. Printre celelalte noutăţi se numără un program antivirus şi alte utilitare cunoscute deja din programele concurenţei, cum ar fi "Wipe" care şterge un fişier astfel încît să nu mai poată fi recuperat, sau "FileFix" care poate recupera fişiere dBase sau Lotus distruse.

PC Tools costă 179$ în SUA şi trebuia să fie livrat începînd din luna mai.

Basic sub Windows

Visual Basic, produs de Microsoft, permite elaborarea unor aplicaţii orientate obiect sub suprafaţa utilizator grafică Windows.

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

Analiştii pieţii din SUA estimează că această prezentare este o mişcare strategică importantă pentru Microsoft. David Buyer, analist la Montgomery Securities: "Visual Basic va determina o şi mai mare răspîndire a lui Windows 3.0 în întreprinderi şi va duce la o creştere a cifrei de afaceri Microsoft. Dezvoltarea de aplicaţii Windows este simplificată semnificativ. Visual Basic va deveni fundamentul multor programe Windows". Visual Basic constă, în mare, din două componente centrale: "Forms", pentru cuplarea suprafeţei grafice cu cunoscutele obiecte Windows, şi "Program Units" care conţin procedurile limbajului.

Rick Sherlund, analist la Goldman Sachs, comentează: "Momentan, cu doar 90 milioane dolari, limbajele de programare reprezintă doar o mică parte din cifra de afaceri Microsoft. După lansarea pe piaţă a lui Visual Basic se aşteaptă rate de creştere de 30 pînă 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 pentru MS-DOS 5.0, "Upper Memory Board", care poate fi montată în toate PC-urile, de la XT şi pînă la 486. Cea mai nouă versiune a sistemului de operare MS-DOS recunoaşte, în gestionarea memoriei, aşa-numitele Upper Memory Blocks (UPB). Cu ajutorul lor driver-ele şi programele rezidente pot fi încărcate în memoria superioară (Upper Memory). În acest mod utilizatorului îi rămîn disponibili pînă la 600 KByte din memoria de bază.

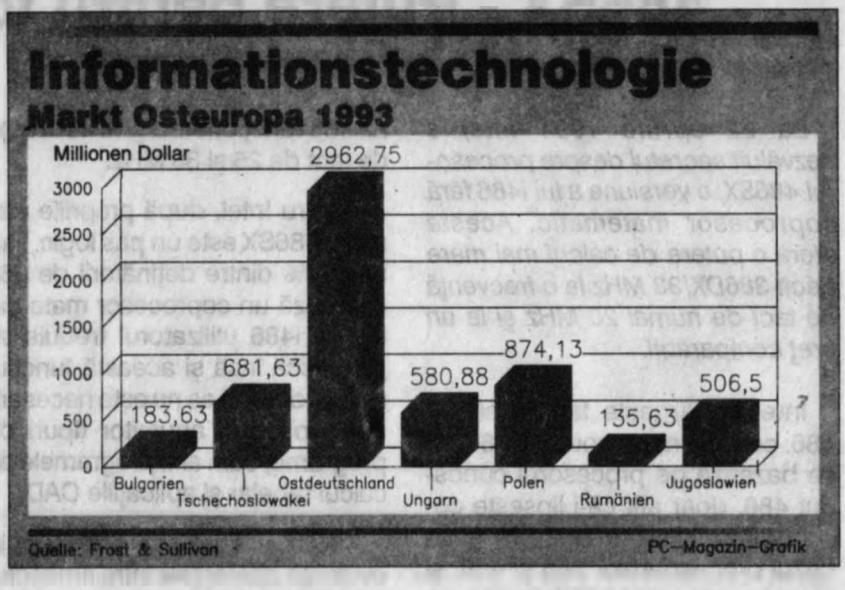
Cartela Upper Memory pune la dispoziţia MS-DOS 5.0 spaţiul fizic RAM necesar. În plus, spaţiul disponibil în extensia de memorie poate fi utilizat, la nevoie, pentru extinderea memoriei de bază. Memoria disponibilă pe placa de extensie, poate fi împărţită în ferestre de cîte 16 KByte care pot fi inserate, la dorinţă, în interiorul spaţiului de adresare cuprins între 512 KByte şi 1 MByte.

Programul de instalare stabileşte domeniile RAM libere din sistem şi preia, orientat dialog, întreaga configurare. Reglarea prin comutatoare DIP sau jumperi a fost înlăturată în întregime. În configuraţia de bază, placa este echipată cu 256 KByte şi 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 piaţă dificilă !

Producătorii de calculatoare care doresc să fie prezenţi în Europa de Est trebuie să aibă nervii tari. Ei trebuie să investească încă de pe acum, dar nu se pot aştepta la cîştiguri decît după 1993, afirmă institutul de prospectare a pieţii Frost & Sullivan într-un studiu.



Febra aurului şi speranţa într-un cîştig rapid, după introducerea economiei de piaţă în ţările din estul Europei, sînt deplasate, este de părere institutul de prospectare a pieţii Frost & Sullivan, în urma unui studiu despre posibilităţile pieţelor esteuropene. "Această piaţă nu este pentru cei cu nervii slabi", se afirmă în studiu, deoarece şanse au doar aceia care investesc acum pentru a-şi asigura imediat prezenţa pe piaţă. Cîştigul rezultat din aceste investiţii 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 şi informatică în ţările 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 şi 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 piaţă esteuropeană se va situa în următorii ani pe teritoriul fostului RDG. Aceasta se explică şi 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 pieţii se va face mai degrabă lent: "La început cea mai mare cerere va exista pentru calculatoare personale. Ulterior se vor vinde destul de bine şi echipamente Unix şi calculatoare de marcă pentru întreprinderi de dimensiune medie. Numai în landurile fostului RDG va exista încă de la început şi o piaţă semnificativă pentru calculatoare mari."

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

(R.M.)

Oglinda preţurilor (mai 1991)

Sisteme	Preţ 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 decît 386DX/33 MHz la o frecvenţă de tact de numai 20 MHz şi la un preţ comparabil.

Intel îşi lărgeşte familia sa de 486: cea mai mare noutate, 486SX, se bazează pe procesorul cunoscut 486, doar atît că-i lipseşte coprocesorul matematic. Ca şi în cazul procesoarelor 386 şi 486, şi în acest caz este vorba despre un procesor pur de 32 de biţi, compatibil integral soft cu predecesorii săi din familia Intel x86.

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

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

Pentru 486SX, Intel oferă şi coprocesorul matematic 487SX, care face calculatoarele SX comparabile cu versiunea completă 486. Combinaţia 486SX - 487SX costă totuşi mai mult decît un 486. O altă limitare este faptul că 486SX se produce deocamdată numai în varianta 20 MHz, în timp ce procesorul 486 funcţionează şi la frecvenţe de tact de 25 şi 33 MHz.

Pentru Intel, după propriile afirmaţii, 486SX este un pas logic. Numai 30% dintre deţinătorii de 386 utilizează un coprocesor matematic. La i486 utilizatorul trebuie să plătească însă şi această funcţiune, cu toate că ea nu este necesară decît în cazul anumitor tipuri de programe, cum ar fi programele de calcul tabelar şi aplicaţiile CAD.

486SX scoate mai pregnant în evidenţă diferenţele între arhitecturile lui 386 şi 486, deoarece i486 este mai mult decît "doar" un procesor 386 cu cache şi coprocesor. Ultimele modele ale familiei x86 au fost îmbunătăţite ca design faţă de 386, ceea ce explică şi avantajul vitezei sporite în cazul lui 486SX.

După datele furnizate de Intel, arhitectura 486, la acelaşi tact, permite o putere de două pînă la trei ori mai mare decît cea a procesoarelor 386. Cu toate că familia x86 aparţine familiei procesoarelor CISC (Complete Instruction Set Computer), 486 şi 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 cuarţului, în timp ce 386 necesită cuarţuri de două ori mai rapide pentru a putea atinge aceeaşi rată de tact efectivă. În toate procesoarele 386 la 33 MHz bate deci inima unui cuarţ de 66 MHz, ceea ce implică măsuri speciale de ecranare. Din această cauză crearea unor procesoare 386 cu o frecvenţă de tact mai ridicată devine o problemă, deoarece un cuarţ de 80 sau de 100 MHz devine tot mai greu de ecranat. Arhitectura 486 foloseşte mai economicos ciclurile de tact şi din acest motiv poate lucra mai uşor la un tact de 50 MHz sau chiar de 100 MHz. Procesorul 486 la 50 MHz nu a fost încă anunţat oficial, dar diverşi producători de calculatoare, printre care şi IBM, vorbesc deja despre el, şi îşi 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 decît o placă de bază 386DX. Viteza procesorului 486SX este influenţată mai ales de cache-ul integrat, care lipseşte la 386.

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

Specialiştii apreciază că vînzările de 386 vor scădea sensibil, dacă producătorii vor transfera avantajul preţului mai scăzut şi asupra produselor finale. Dacă însă vor pretinde un preţ suplimentar pentru puterea sporită, atunci 386 mai are încă şanse bune de a se menţine pe piaţă. Cît de rapid va reuşi deci 486SX să-l înlocuiască pe 386 pe piaţă rămîne de văzut.

Politica de preţuri agresivă, practicată de Intel, lasă să se întrevadă faptul că în Santa Clara schimbarea este dorită mai degrabă astăzi decît 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 anunţarea oficială a procesorului Intel 486SX, IBM a şi dat o soluţie pentru modelele 90 şi 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 şi-a prezentat noul procesor 486SX IBM şi-a prezentat modelele PS/2 90 şi 95 dotate cu acest procesor.

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

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

Cele 2 variante noi reprezintă configuraţiile iniţiale ale modelelor PS/2 90 şi 95, pe care IBM le-a oferit pînă în prezent doar cu versiunea completă a procesorului 486.

Vedere de ansamblu

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

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) şi o placă grafică modernă XGA, care oferă o rezoluţie de 1034x768 pixeli la 256 de culori, sau de 640x400 pixeli la 300.000 culori (16 biţi).

Modelul 95 se aseamănă în mare măsură cu modelul 90 în ceea ce priveşte datele tehnice, doar că dispune de o carcasă tower mult mai mare şi 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 şi de controller-ul de harddisk) şi dispune de spaţiu 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 operaţiuni, 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 şi 95. Peter Richard, susţine că ar fi o demonstraţie de tehnologie şi nu de un produs terminat. Placa de 50 MHz ar putea fi în scurt timp o extensie a modelelor 90 şi 95.

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

(I.M.)

Familia x86 dintr-o privire

Date comparative

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

Next please

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

Ele sînt domnitoarele reţelelor, maestrele simulărilor complexe, stăpînele graficii color - cu staţiile de lucru începe lumea de dincolo de PC-uri.

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

Această limită a preţurilor 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 piaţă trei noi staţii de lucru la preţuri tentante: "Nextstation" (cca. 10.800 DM), "Nextstation Color" (cca. 17.300 DM) şi "Nextcube" (cca. 17.300 DM).

Pentru utilizatorii obişnuiţi aceasta este ca şi cum le-ai oferi o maşină Ferrari la jumătate de preţ: senzaţional de ieftin, dar cu toate acestea de neplătit.

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

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

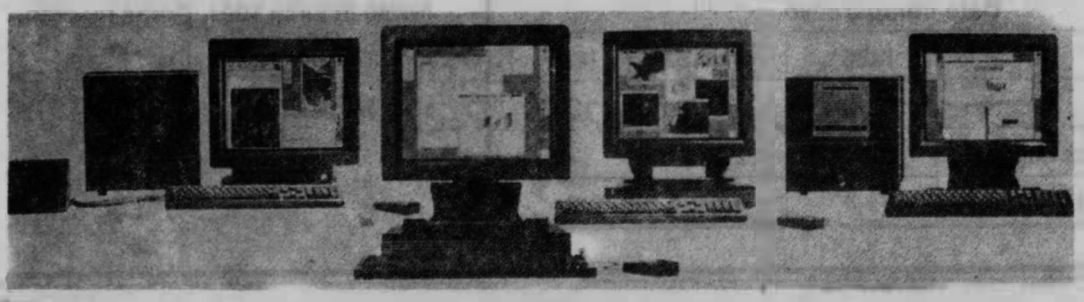
Fiecare Next dispune de un lector de dischete de 3,5" care lucrează cu "Extended Density" pe care se pot stoca de două ori mai multe date decît pe dischetele "High Density" ale calculatoarelor compatibile IBM; dischetele au o capacitate de 2,44 MByte şi costă 8 DM/bucată. Lectorul recunoaşte şi dischetele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

Versiunile de bază "Next Station" şi "Next Station Color" sînt livrate cu harddisk-uri avînd capacitatea de 105 MByte (timp de acces 17 ms, rată de transfer 4 MByte/sec). Opţional poate fi livrat şi un harddisk de 340 Mbyte (timp de acces 15 ms, rată de transfer 8 MByte/sec). "Nextcube" poate fi echipat chiar şi cu un harddisk de 1,44 GByte (timp de acces 13 ms).

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

Dacă totuşi nu este suficient, atunci cu "Nextcube" se pot obţine imagini şi mai împestriţate: 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 biţi, 24 de biţi permit generarea a 16,7 milioane de culori, iar cei 8 biţi rămaşi permit reprezentarea a 256 de nivele de transparenţă), rezoluţia rămînînd aceeaşi.

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



Familia Next.

poate înzestra documentele şi fişierele şi cu comentarii verbale.

Sisemul de operare utilizat de Next este Unix, sistem de operare pe care îl folosesc de altfel şi majoritatea celorlalte tipuri de staţii de lucru. Unix este recomandabil în special pentru lucrul în reţea şi pentru a permite rularea simultană a mai multor programe pe acelaşi calculator (multitasking). Dezavantajul: Unix este destul de incomod în operare. Pentru a contracara acest minus Next utilizează suprafaţa utilizator grafică "Nextstep" asemănătoare suprafeţei 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 manipulînd simboluri cu ajutorul mouse-lui.

Next produce şi o imprimantă laser (8 pag./min.), care permite o rezoluţie selectabilă de 300 sau 400 dpi (dot per inch). Cu toate că imprimanta laser nu este o imprimantă Postscript (aşa se explică şi preţul 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 şi pe hîrtie (WYSIWYG = what you see is what you get).

Dar nici cel mai bun calculator şi nici cea mai bună suprafaţă 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 reproş, pe harddisk-ul fiecărui Next livrîndu-se în mod standard un număr de pachete soft.

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

Angajamentul unor firme de renume oferă garanţia faptului că nici în viitor nu va lipsi softul pentru Next: Wordperfect, Lotus, Ahton Tate şi Oracle se ocupă de dezvoltarea de programe pentru Next.

Model: Next-Station
Producător: Next Computers
Preţ: 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", rezoluţie 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.

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

(R.M.)

Software pentru Next

Next livrează fiecare calculator cu o selecţie 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 reţea. Cu ajutorul unui modem-telefax acesta îşi poate utiliza Next-ul şi ca telefax.
- **WriteNow**, un procesor de texte (fără funcţie de scriere de scrisori în serie).
- **Draw**, un program de desenare.
- **Digital Webster**, un dicţionar englez, şi Webster's Thesaurus.
- **Date**, o agendă de lucru.

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

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

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

A calcula şi 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 piaţă şi deja a fost prezentată noua versiune Quattro Pro 3.0.

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

Pe lîngă suprafaţa orientată caracter este disponibil acum şi un mod de lucru WYSIWYG care permite virtualizarea tuturor fonturilor, a atributelor de scriere, a liniilor şi a umbririlor, aşa 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 inscripţionărilor liniilor şi coloanelor, butoanele meniurile şi 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% şi 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 WYSIWG Quattro Pro atinge o viteză acceptabilă. Defilarea pe ecran şi funcţiile la nivel de bloc au devenit mai lente, din cauza modului grafic, dar aceste dezavantaje se păstrează în limite acceptabile.

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

Această metodă ajută puţin, totuşi, 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 sfîrşit (perforată) se poate obţine o copie 1:1 a tabelei.

Şi în editorul grafic au fost incluse noutăţi. Ca şi în cazul programelor de grafică profesionale desenul poate fi construit avînd la bază un rastru, care poate fi folosit pentru poziţionarea şi pentru dimensionarea obiectelor. Simultan poate fi activată şi funcţia "snap-to" caz în care Quattro Pro va începe unui obiect grafic întotdeauna exact într-un punct al rastrului şi îl va sfîrşi într-un alt punct al acestuia.

Ca şi pînă acum cîmpurile de text pot fi definite ca butoane şi la selectarea lor pot apela o altă diagramă sau un macrou. Nou este aşa- 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 afişată următoarea imagine.

Modul de prezentare al unei expuneri (slide-show) poate fi îmbunătăţit 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 atenţia spectatorului. Extrem de utilă este şi funcţia "autosave" în cazul graficelor: cu ajutorul ei putem fi siguri că orice modificare efectuată într-o diagramă va fi salvată pe harddisk.

Pe lîngă noutăţile enumerate, Quattro Pro 3.0 dispune şi de cîteva mici funcţii care în versiunea beta nu erau încă definitivate. În versiunea finală va fi posibilă atît citirea cît şi scrierea de fişiere 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 operaţie, toate fişierele deschise la un moment dat. Cîteva comenzi noi pentru construirea de macrouri permit însoţirea prezentărilor cu mici bucăţi muzicale.

Quattro Pro dovedeşte cu versiunea 3.0 că este un concurent serios în competiţia pe care o desfăşoară împreună cu Lotus 1-2-3 şi Excel, pentru titlul de cel mai bun program de calcul tabelar. Cerinţele hard (AT, 512 KByte RAM, harddisk) nu vor creşte nici în noua versiune.

(R.M.)

Tabel comparativ

	Quattro Pro 3.0	Excel 3.0	Wingz
Preţ	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
Fişiere deschise simultan	32	funcţie de mem.	funcţie de mem.
Foi de calcul tridimensionale	nu	nu	nu
Lungime cîmp	255 caractere	255 caractere	255 caractere
Coprocesor	da	da	da
Spaţiu ocupat pe harddisk	4,5 MByte	5 MByte	3 MByte
Caracteristici de putere: Calcul			
Funcţii autodefinite	nu	da	da
Căutare pt. obţinerea unei valori finale impuse	da	da	nu
Solver	da	da	nu
Tabele ce-se-întîmplă-dacă cu X variabile	2	2	2
Consolidare inteligentă	nu	da	nu
Cuplare cu fişiere de pe harddisk	da	da	nu
Caracteristici de putere: Publicistică			
Combinare de tabele şi diagrame	da	da	da
Suport Postscript	da	da	da
Import formate grafice	CLP, CGM	clippboard	clippboard
Funcţii 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
Macrouri	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 planşetei de desen sînt numărate. Pentru efectuarea de desene tehnice PC-urile cu soft corespunzător au nevoie de mai puţin loc, risipesc mai puţină hîrtie şi oferă utilizatorului mai multă putere şi confort decît creionul.

Prescurtarea CAD este folosită pentru Computer Aided Drafting, şi respectiv pentru Computer Aided Design. A doua variantă a acestei noţiuni este mai cuprinzătoare şi mai uzuală. În timp ce Drafting se referă la desenarea cu ajutorul calculatorului, Design depăşeşte cu mult cadrul lucrărilor grafice efectuate cu ajutorul calculatorului. Din acest motiv ar fi greşit să traducem această noţiune doar prin construcţie: ea cuprinde mult mai mult şi anume întregul spectru al activităţilor care trebuiesc depuse pentru crearea şi întreţinerea unui produs.

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

Un element esenţial care trebuie să ghideze decizia achiziţiei unui sistem CAD este şi compatibilitatea formatului datelor cu a altor aplicaţii. 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 şi de accesibile ca preţ 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: construcţii de maşini, arhitectură, electrotehnică şi mai nou din ce în ce mai mult arhitecura de interior. Producerea de softuri pentru crearea de construcţii 3D şi umbriri şi posibilităţile care decurg din acestea, de a crea şi calcula imagini spaţiale expresive de produse, clădiri sau peisaje, au propulsat CAD-ul în vîrful aplicaţiilor profesionale de grafică.

Diferenţa dintre un program CAD şi 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 şi dimensională exactă şi îi ataşează diferite atribute, cum ar fi de pildă starea, culoarea, tipul, nivelul (layer), care îi stabilesc apartenenţa la o anumită grupă de obiecte. În plus mai pot fi memorate date despre materiale, firmele furnizoare, indicaţii 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 şi multe altele.

Programele de desenare nu răspund tuturor acestor necesităţi. În cazul lor imaginea este memorată într-un fişier ca o imagine rastru, 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
- ieşire şi export

Masca ecran

Monitorul (ecranul) este împărţit 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 şi la marginile sale laterale.

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

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

Din punct de vedere hardware pretenţiile referitoare la monitor şi la adaptorul grafic sînt ridicate. Deoarece în urma ocupării unei părţi de ecran cu cîmpurile de control rămîne relativ puţin spaţiu pentru desenul prpriu-zis, diagonala ecran trebuie să fie cît mai mare. Dimensiunea ei este limitată însă din considerente financiare şi tehnice. Standard sînt utilizate monitoare de 19" cu o rezoluţie de 1024x768 puncte. În acest caz se poate lucra profesional. Costurile se păstrează şi 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

mulţumitor şi 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 posibilităţile sînt limitate. Cea mai simplă, dar şi 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 soluţie 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 măşti 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 dispoziţie comenzi de desenare elementare care să permită desenarea de linii, curbe, cercuri, poligoane, corpuri 3D elementare sau texte. În plus mai trebuie să existe comenzi de uşurare a lucrului; cum ar fi: "capturarea" cursorului într-un anumit punct, la intersecţia a două linii. Doar în acest mod pot fi executate desene exacte şi la scară.

Mai sînt şi alte comenzi care pot uşura lucrul: elementele desenate trebuie să poată fi deplasate, copiate, oglindite, întinse sau scurtate. 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, şi, şi...

Această înşiruire ar putea continua la nesfîrşit, dar se poate observa deja cît de complex este un soft pentru CAD şi cîte exerciţii sînt necesare pînă să se înveţe modul de operare.

Baza de date CAD

Toate elementele şi respectiv obiectele unui desen sînt memorate şi prelucrate de către programul CAD într-un anumit format. Printre aceste prelucrări se numără, în primul rînd, regăsirea unei construcţii deja desenate. Pentru ca acest lucru să fie posibil trebuie respectate anumite convenţii la creare şi memorare (stocare): la stocarea unui desen pe lîngă numele acestuia mai trebuie stabilite şi alte elemente de bază, cum ar fi formatul hîrtiei, unităţile de măsură, nivelul, tipul liniilor, directorul, ş.a.m.d.

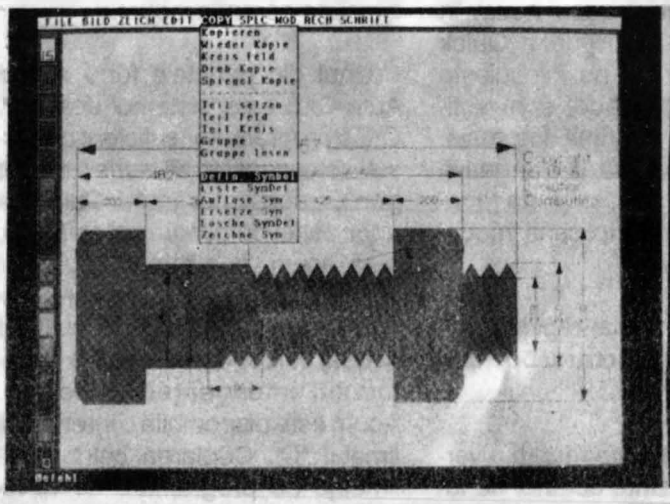
Ieşire şi export

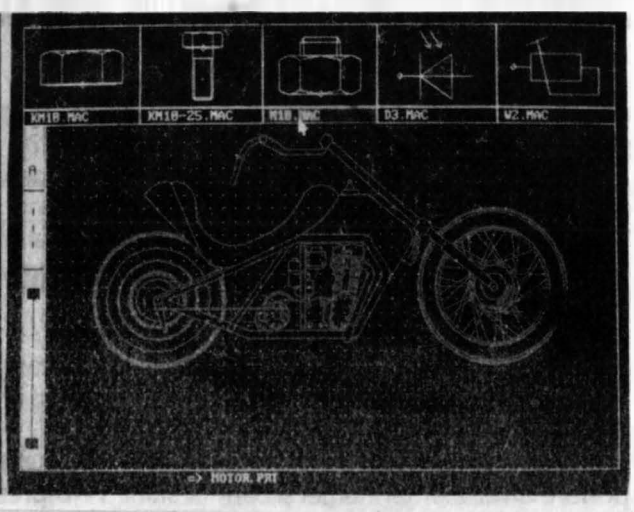
Rezultatul unui proces de desenare sau de construcţie, în zilele noastre, mai trebuie încă să fie prezentat pe hîrtie. Pentru reprezentarea pe hîrtie sînt disponibile diferite variante: pentru utilizări profesionale şi industriale se utilizează un plotter, pentru amatori este suficientă o imprimantă.

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

Există o paletă foarte bogată de programe CAD, cu performanţe, dar şi la preţuri, 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), Autosketch 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.)

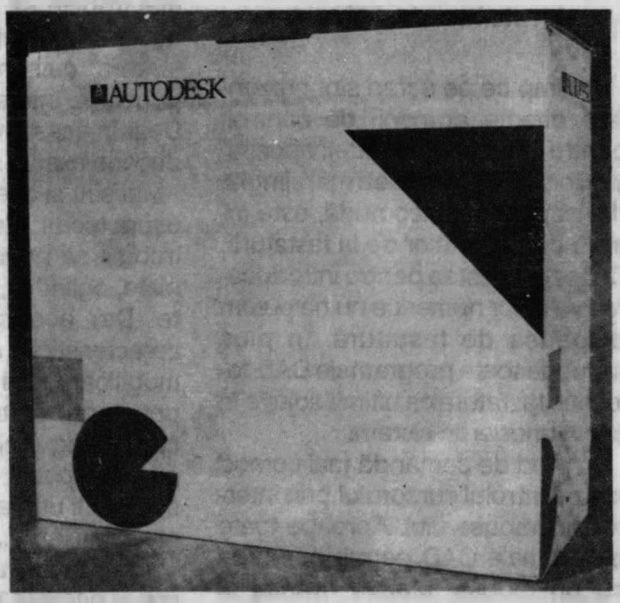




Lucrurile bune se obţin cu răbdare

Test: Auto - CAD 11.0

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



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

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

Ca întotdeauna, se pare că aşteptarea a meritat. Abundenţa noutăţilor versiunii Auto CAD 11.0 este de luat în seamă. Pe primul loc ar fi de semnalat facilităţile 3D. Cuvîntul magic pentru acestea este "AME" (Advanced Modeling Extension). În spatele acestei noţiuni se ascunde o nouă procedură de tratare a construcţiilor tridimensionale orientate pe volume, un progres semnificativ faţă de modelele anterioare orientate pe muchii şi suprafeţe.

Diferenţa între reprezentările 3D ale versiunilor 10 şi 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 construcţie din hîrtie. Din afară arăta ca un corp "adevărat" dar în interior nu se putea privi. Noua metodă - orientată pe volume - descrie corpurile şi în interior. Întregul mediu cu geometria sa, caracteristicile de material şi rezultînd de aici modul de comportare al rezistenţei inerţiei şi centrului de greutate, sînt descrise prin metoda AME.

Acum, cu ajutorul unei comenzi, pot fi afişate toate valorile relevante şi pot fi preluate pentru calcul în aplicaţiile învecinate. În afară de acestea se pot executa secţiuni la înălţimea dorită. Construcţiile complicate pot fi realizate prin îmbinarea unor corpuri simple şi la nevoie pot fi descompuse din nou.

Strîns legată de reprezentarea 3D este şi posibilitatea de umbrire a suprafeţelor. 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 funcţiunea "Quick Shade", care deşi nu înlocuieşte 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 lucru. Acum există posibilitatea definirii unor aşa-numite domenii model şi 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 situaţia în domeniul hîrtiei. Aici pot fi definite sau mascate diferite ferestre care să conţină numai anumite detalii ale modelului.

Ferestrele pot fi aduse împreună pe formatul de hîrtie dorit şi listate prin intermediul unui plotter sau a unei imprimante. În acest mod pot fi create cu uşurinţă desene de execuţie pentru producţie.

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 inteligenţei artificiale). Acum este disponibilă o interfaţă la limajul "C". Cuplarea celor două limbaje de programare se dove-

deşte 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 soluţii specifice branşei.

Auto-CAD 11.0 poate fi utilizat şi în reţea, funcţionînd pe toate sistemele de operare de reţea, cum ar fi Novell, 3Com sau NFS. Posibilitatea de a fi instalat şi 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 specificaţi. De asemenea pot fi blocate fişierele, ceea ce dă siguranţă că un anumit desen nu este prelucrat simultan de mai mulţi utilizatori. Extensiile pentru lucrul în reţea implică un efort de instalare mai mare, dar o parte semnificativă a acestui efort este preluată de procedura de instalare.

Ţinînd cont de multitudinea noutăţilor, nu este de mirare că Auto- CAD 11.0 nu poate fi rulat decît pe calculatoare cu 386 cu coprocesor şi respectiv pe calculatoare cu 486. Memoria minimă necesară este de 2 MByte. Dacă se doreşte utilizarea AME sau a funcţiunilor de umbrire atunci sînt necesari cel puţin 4 MByte. Auto-CAD 11.0 ştie să lucreze şi 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 recunoaşte standardul VCPI (Virtual Control Program Interface). Acest standard include convenţiile stabilite de mai mulţi producători pentru stabilirea modului în care sistemul de operare DOS trebuie să trateze memoria "extinsă" şi "expandată". Pentru aceasta sînt necesare drivere de memorie care să respecte standardul VCPI. Dintre acestea enumerăm: "Compaq CEMM 4.01", "Quarterdeck QEMM 4.01" şi "386 - MAX" al firmei Qualitas System.

Concluzionînd aşteptarea lui Auto - CAD 11.0 a meritat. Noile facilităţi nu sînt dedicatee în primul rînd utilizatorilor obişnuiţi, dar le sînt indispensabile utilizatorilor profesionali.

Preţ: 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

Fişier DXF Format fişier pentru schimbul de desene între diferite programe CAD

Grid Rastru. Pe monitor este afişat un rastru pentru orientare. Acesta nu va fi listat pe plotter sau imprimantă

Grupă Obiectele individuale pot fi incluse în grupe pentru a putea fi prelucrate împreună. Grupările pot fi descompuse din nou, la nevoie.

Hatch Haşură. Haşurarea trebuie să respecte normele DIN

HPGL Limbaj de comandă pentru controlul plotter-elor (Hewlett Packard Graphics Language).

Layer Nivel desen: asemănător modului de lucru cu folii pe un diaproiector. Un nivel poate fi mascat, aceasta înseamnă că nu va fi afişat pe ecran.

Listă de componente Lista simbolurilor care apar într-un desen şi 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 secţiunii vizibile din desen

Plotter Masă de desen automată. Aparat de ieşire cu ajutorul căruia desenul este reprezentat în tuş, sau carioca, pe hîrtie

Polyline Trasarea unei linii care constă din mai multe segmente de linii, cu o singură operaţie

Printplot Reprezentarea desenului pe o imprimantă de rezoluţie ridicată. Nu se atinge însă rezoluţia unui plotter.

Simbol (Macrou) Detaliile utilizate frecvent pot fi memorate ca macrouri şi ulterior pot fi inserate la nevoie şi în alte desene.

Snap Cursorul poate fi poziţionat numai în anumite puncte definite prin valoarea de prindere (snap). În acest mod se poate face poziţionarea exactă pe un nod (o intersecţie), fără a fi necesară specificarea manuală a coordonatelor punctului respectiv.

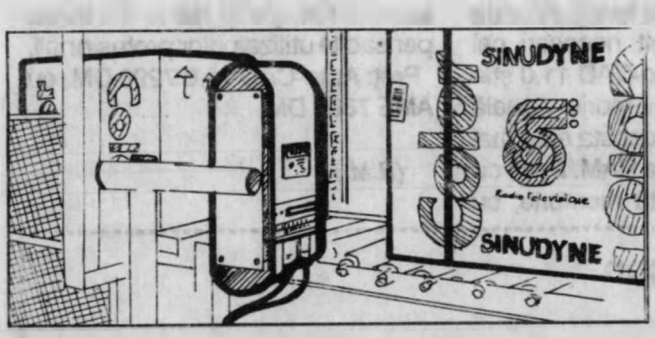
Varia Scara unui obiect poate fi corelată cu scara restului desenului. Această operaţie este necesară atunci cînd într-un desen se inserează macrouri (obiecte) predefinite.

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 şi vor fi afişate mai multe detalii, dacă este mai mic decît 1 desenul va fi micşorat şi se va obţine o vedere de ansamblu mai bună.

Cod de bare

Printre cele mai mari probleme ale întreprinderilor industriale şi comerciale moderne se numără şi felul în care funcţionează magaziile, felul în care se realizează fluxul tehnologic. Cît anume din timpul de realizare al unei maşini mai complexe este timp de lucru/asamblare efectiv şi cît anume este timp "de aşteptare" 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 şi în preţ. Cerinţele din ce în ce mai dure în această direcţie au determinat dezvoltări în două direcţii 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 şi transmisia lor, printr-o interfaţă 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 expediţie, fişe de magazie, bilete de avion, legitimaţii, etichete etc.

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

Preluarea automată a datelor tipărite oferă utilizatorului diverse posibilităţi:

- generarea tuturor documentelor ce privesc depozitarea şi expediţia (aviz de expediţie, fişă de urmărire, chitanţă, etc.)
- controlul automat al întregului sistem de transport şi depozitare
- inventar permanent la zi şi (prin aceasta) date exacte relativ la stocurile existente (de produse şi/sau materii prime); ca o bază pentru deciziile conducerii.

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

- în biblioteci (şi/sau videoteci): toate cărţile sînt identificate prin intermediul unui cod de bare. De asemenea, fiecare utilizator are o legitimaţie, 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 şi ieşirile de marfă se înregistrează prin intermediul calculatorului. Prelucrarea codului fiecărui articol se face prin intermediul codului de bare, aplicat pe ambalaje, şi al "creionului de citire".
- legitimaţiile sau abonamentele se recunosc folosind cititoare cu fantă; astfel, se poate da acces la calculatoare, intrare la biblioteci, schi-lift, domenii militare etc.
- pe benzi de fabricaţie, codurile de bare aplicate pe cartoane permit preluarea datelor - prin laser-scanner montat fix, într-o anumită poziţie, astfel încît citirea să

se facă de la o distanţă de 200 - 800 mm - şi transmiterea lor la nivelul superior.

Codarea

La concurenţă cu alte sisteme de codare care să permită identificarea automată sau semiautomată a diverselor entităţi ce trebuiesc urmărite, codul de bare are un avantaj indiscutabil: este relativ simplu de produs şi relativ simplu de recunoscut.

Sistemul OCR se bazează pe recunoaşterea caracterelor alfabetice cu care sîntem obişnuiţi, cu restricţia că forma caracterelor trebuie să fie standardizată pentru a permite citirea automată. Avantajul claritâţii - datele nu trebuiesc codate - este anulat de aparatele de citire şi recunoaştere complicate, precum şi de cerinţele ridicate vizavi de calitatea imprimării. La acesta se adaugă rate de erori mari: cercetări făcute în aplicaţiile tip supermarket au evidenţiat peste 30% citiri nereuşite din prima încercare şi peste 1% citiri eronate.

Sistemul magnetic cere şi el un efort tehnic şi financiar deosebit. Cînd codul trebuie produs în cantităţi numerice importante, limita economicităţii este foarte repede depăşită.

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

Spaţiu elementul deschis dintre două linii ale unui cod

Interspaţiu spaţiul dintre ultima linie a unui caracter şi prima linie a următorului caracter în cadrul unui cod de baze discret

Element expresie folosită pentru a descrie o linie sau un spaţiu component al unui cod

Modul elementul cel mai îngust dintr-un cod. Liniile sau spaţiile mai late se calculează ca multiplii ai modulului

Lărgimea modulului X: defineşte lărgimea celui mai îngust element (de ex. X=0,33 mm) Zona "de linişte" - zona de culoare deschisă dinainte de caracterul de start şi de după caracterul stop. Zona de linişte este necesară pentru a da cititorului informaţia de "început de cod". Minimul este de 10 ori lărgimea modulului, dar cel puţin 2,5 mm. La aplicaţii ce folosesc scannere cu un domeniu de contrast mai mare, zona de linişte trebuie aleasă mai mare (L = 15 x lărgimea modulului, dar cel puţin 6,5 mm).

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 şi încă o zonă deschisă. Dedesubt, un rînd cu textul în clar.

Caracter de start/stop - orice cod începe cu un caracter de start şi se termină cu un caracter de stop. Astfel citirea se poate face în ambele direcţii.

Cod cu autoverificare - cod de bare care permite verificarea fiecărui caracter, după un algoritm prestabilit. Erorile de substituţie sînt posibile

Cod discret cod de bare în care fiecare semn începe şi se termină cu o linie. Interspaţiul nu face parte din cod.

Cod continuu cod de bare la care interspaţiul face parte din cod

Cifră de control - unul sau mai multe semne suplimentare, care permit recunoaşterea erorilor

Eroare de substituţie - un semn e înlocuit de un altul. Numai prin utilizarea unei cifre de control, acest tip de eroare poate fi aproape exclus

Citire nereuşită - decodorul nu este în stare să decodifice informţia citită

Citire falsă informaţia decodată nu coincide cu cea din codul de bare (eroare de substituţie).

Rata de primă citire - numărul de citiri corecte la prima încercare, împărţit la numărul de încercări

Cod de fotoimprimare - 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/spaţii înguste sau largi. Secvenţa acestor linii respectiv spaţii determină un anumit conţinut numeric sau chiar alfanumeric.

Citirea se face optic. Prin reflexia diferenţiată a liniilor negre şi spaţiilor albe, în receptorul optic apare un tren de impulsuri care corespunde acestei secvenţe de linii şi spaţii.

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

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

Toţi aceşti factori au făcut ca în decursul timpului pentru diverse aplicaţii să apară o multitudine de coduri de bare, care toate îşi 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

Generalităţi - cod numeric; pot fi reprezentate cifrele 0-9. Acest cod e construit cu două linii late şi trei linii înguste. Raportul de imprimare linie îngustă:linie lată (R) este de 1:2 pînă la 1:3. Spaţiile nu conţin informaţii.

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

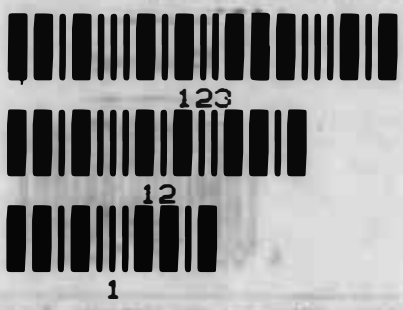
Dezavantaje - densitate mică a informaţiei (4,2 mm/cifră la o lăţime a modulului x = 0.3mm şi R = 1:3.)

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

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

Avantaje: densitate mare de informaţie. (2,7 mm/cifră la o lărgime a modulului de x 0,3 mm şi un raport de imprimare R = 1:3); autoverificabil

Dezavantaje: toate spaţiile poartă informaţie, de aceea toleranţa 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 / Spaţiul 1 - 5
1 = Linie/Spaţiu lat(ă)
0 = Linie/Spaţiu îngust(ă)

Exemplu:

Code 2/5 3 linii (Matrix)

Generalităţi - cod numeric; pot fi reprezentate cifrele 0-9. Fiecare caracter constă din 5 elemente, 3 linii şi două spaţii. Două elemente sînt late şi 3 sînt înguste. Deci atît liniile cît şi spaţiile pot fi late sau înguste. Interspaţiul nu poartă informaţie. Raportul de imprimare (R) linie îngustă: linie lată = 1:2,25 pînă la 1:3; spaţiu îngust: spaţiu lat = 1:2,25 pînă la 1:3. Linie deosebit de lată în caracterul de start şi cel de stop.

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

Dezavantaje: toleranţă <= 10% Caracter supralat în informaţia 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 = Spaţiul 1 - 2
1 = Linie/Spaţiu lat(ă)
0 = Linie/Spaţiu îngust(ă)
11 = 1,5 x Linie lată

Exemplu:

Cod 2/5 3 linii (Datalogic)

Generalităţi - cod numeric, pot fi reprezentate cifrele 0-9. Fiecare caracter constă din 5 elemente, 3 linii şi două spaţii. Două din elemente sînt late şi 3 sînt înguste. Deci atît liniile cît şi spaţiile pot fi late sau înguste. Interspaţiul nu poartă informaţie. Raportul de imprimare (R): linie îngustă: linie lată = 1:2,25 pînă la 1:3; spaţiu îngust: spaţiu lat = 1:2,25 pînă la 1:3

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

Dezavantaje - toleranţă 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 = Spaţiul 1 - 2
1 = Linie/Spaţiu lat(ă)
0 = Linie/Spaţiu îngust(ă)

Exemplu

Codabar

Generalităţi - 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 şi 3 spaţii. Se folosesc 2 sau 3 elemente late şi 5 sau 4 elemente înguste pentru reprezentarea codului. Interspaţiul nu poartă informaţie. Raportul de imprimare: (R) linie îngustă: linie lată (1:2) pînă la 1:3; spaţiu îngust: spaţiu 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. Interspaţiul nu poartă informaţie.

Dezavantaje: densitate de informaţie redusă. (5,5 mm/cifră la o lăţime a modulului de x = 0,3 mm şi R = 1:3.)

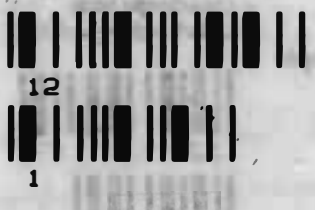
Tabela de coduri:

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

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 = Spaţiul 1 - 3
1 = Linie/Spaţiu lat(ă)
0 = Linie/Spaţiu îngust(ă)

Exemplu



Cod 39

Generalităţi - cod alfanumeric. Pot fi reprezentate cifrele 0-9, 26 de litere, 7 caractere speciale. Fiecare caracter constă din 9 elemente (5 linii şi 4 spaţii). 3 elemente sînt late şi 6 sînt înguste, cu excepţia caracterelor speciale. Interspaţiul nu poartă informaţie.

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

Avantaje: reprezentare alfanumerică

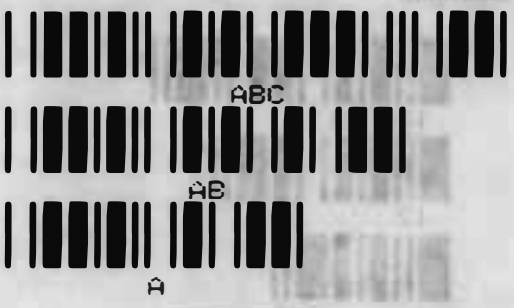
Dezavantaje: densitate mică de informaţie (4,8 mm/cifră la o lăţime a modulului de x = 0,3 mm şi R = 1:3.) toleranţă mică (10%).

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 = Spaţiul 1 - 4
1 = Linie/Spaţiu lat(ă)
0 = Linie/Spaţiu îngust(ă)

Exemplu:



Cod EAN

Generalităţi - cod numeric, pot fi reprezentate cifrele 0-9. Fiecare caracter constă din 11 elemente. Toate liniile şi spaţiile poartă informaţie. Pot fi reprezentate numai grupuri de 8 sau 13 caractere împreună.

Avantaje - densitate mare de informaţie în 10 dimensiuni diferite.

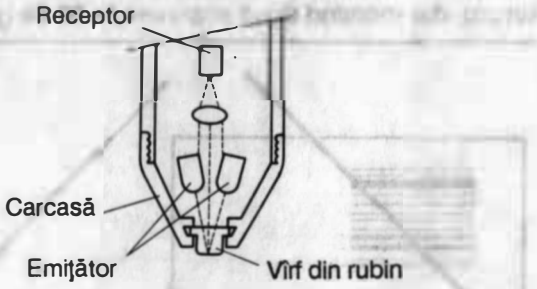
Dezavantaje - toleranţe foarte mici.

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 funcţionează pe baza efectului ilustrat în figură. Emiţătoarele emit lumină roşie (630 mm) sau infraroşie (950 mm); creionul este "dus" deasupra hîrtiei, atingînd-o în permanenţă. Suprafaţa ce conţine 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 şi spaţiile într-un tren de impulsuri. Acesta este disponibilizat în formă numerică la ieşire. Un decodor ataşat transformă trenul de impulsuri în date pe care le transmite serial (printr-o interfaţă RS232, RS422 sau 20mA)

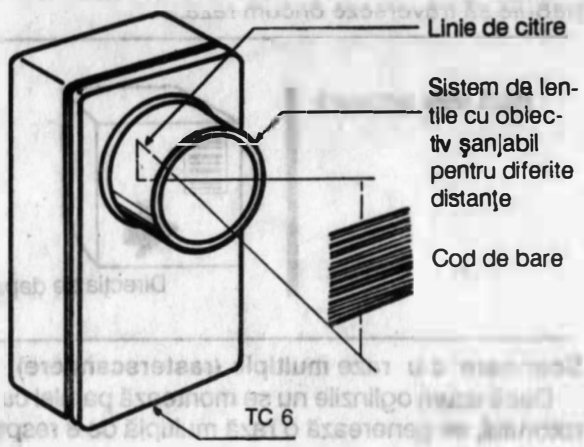


Cititoarele distanţoare folosesc acelaşi sitem, cu diferenţa că citirea se face la o distanţă de 10 mm +- 1 mm. Înălţimea minimă a codului de bare este de 3mm.

Cititoare dinamice

Sistemul cu cameră de luat vederi

Un rînd de fotodiode este dispus în spatele focarului, ca la o cameră suprafaţa filmului. Prin interogare internă rapidă a punctelor individuale, un cod de bare este transformat astfel într-un rastru de puncte. Cel puţin 4 pixeli trebuie să fie proiecţia unei linii înguste pentru ca aceasta să poată fi recunoscută. Utilizînd obiective variabile, pot fi recunoscute coduri de bară la pînă 4 m distanţă. Ca funcţie, 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 incidenţă.

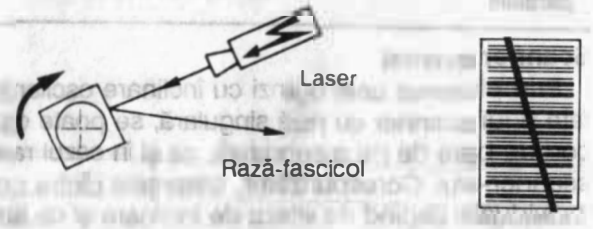
Dacă 4 sau mai multe oglinzi se montează pe un rotor şi se reflectă razele luminoase ale soarelui sau ale unei surse de lumină artificiale atunci pe o suprafaţă se proiectează un punct care se deplasează pe un traseu rectiliniu, repetitiv. Dacă rotorul se învîrte suficient de repede, atunci funcţie de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

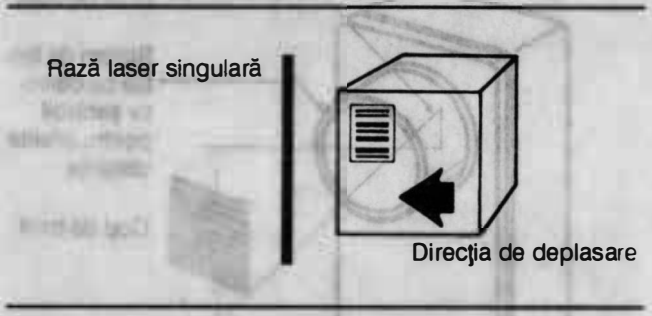
Scanner cu rază singulară

Folosind ca sursă de lumină un laser cu He - Ne, se ajunge la principiul de funcţionare 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 rotaţie a rotorului, se proiectează o rază singulară de lumină.



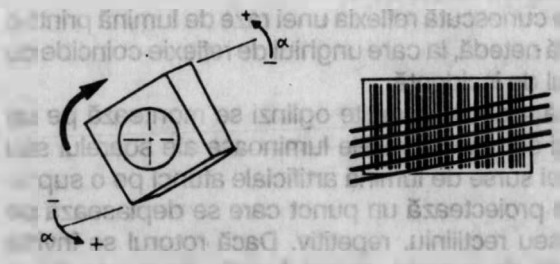
Acest scanner se foloseşte cînd pe o bandă transportoare liniile de cod sînt paralele cu direcţia deplasării iar direcţia citirii este în unghi drept faţă de aceasta. Liniile sînt orientate orizontal, iar raza laser vertical. Avantajul este că nu trebuie realizată o precizie prea mare a înălţimii la care se plasează codul (+- 600 mm).

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



Scannere cu raze multiple (rasterscannere)

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



Rasterscanner-ul se foloseşte atunci cînd liniile codului pot fi aplicate numai vertical faţă de direcţia de deplasare a încărcăturii (imprimanta poate tipări numai într-o direcţie, eticheta poate fi aplicată numai într-o anumită poziţie, etc.). Astfel se asigură o anumită toleranţă în ce priveşte înălţimea codurilor.

Scanner evantai

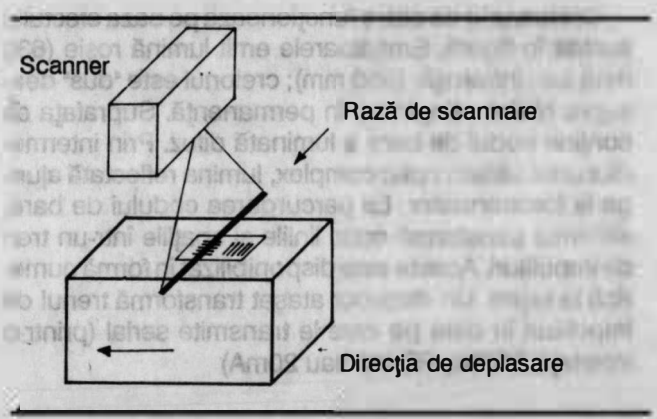
Prin plasarea unei oglinzi cu înclinare oscilantă în faţa unui scanner cu rază singulară, se poate citi un cod de bare de pe o suprafaţă, ca şi în cazul rasterscanner-ului. Corespunzător, distanţele dintre razele individuale depind de viteza de înclinare şi de amplitudinea oscilaţiilor

Citirea omnidirecţională

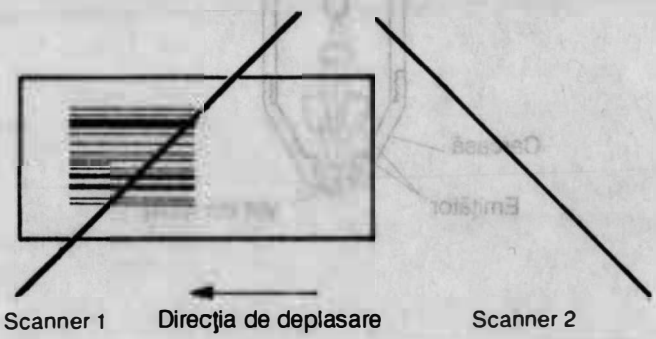
Dacă lucrurile ce se identifică nu au o orientare totdeauna aceeaşi, folosind un cod adecvat, acesta poate fi citit omnidirecţional.

Eticheta în T

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

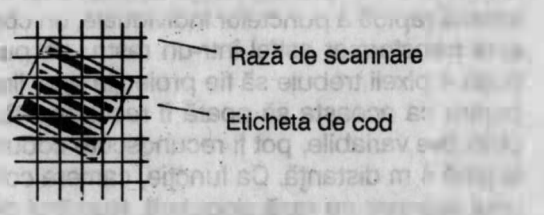


Citirea omnidirecţională cu două scannere cu rază individuală este o altă posibilitate, folosind un cod simplu, dar montînd două scannere la 90 de grade.



Scannere omnidirecţionale

Cu un scanner care nu emite numai unul sau mai multe raze paralele, ci o reţea completă, pot fi citite apoi toate codurile care au o construcţie cel puţin pătratică (lungimea liniilor = lăţimea lor), indiferent de poziţia codului faţă 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.

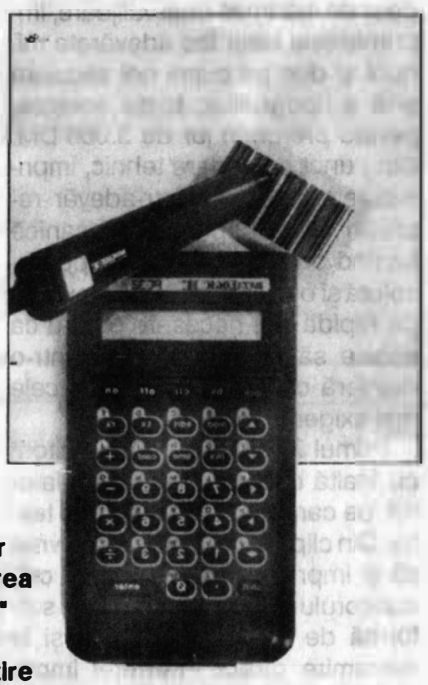


În final, vă prezentăm şi cîteva dispozitive, împreună cu aplicaţiile tipice. Ilustraţiile se bazează pe prospecte ale firmei Datalogic. Relaţii suplimentare relativ la gama de produse ofertate pot fi obţinute direct (Data-Logic International GmbH, Dreisteinstrasse 49 b, A - 2372 Giesshubel bei Wien, Austria). Altă firmă "cu nume" în domeniu: APOG, Paris, 298 av. G'ral de Gaulle, 92.140 Clamart, France.

Creion de citire

Aplicaţii:

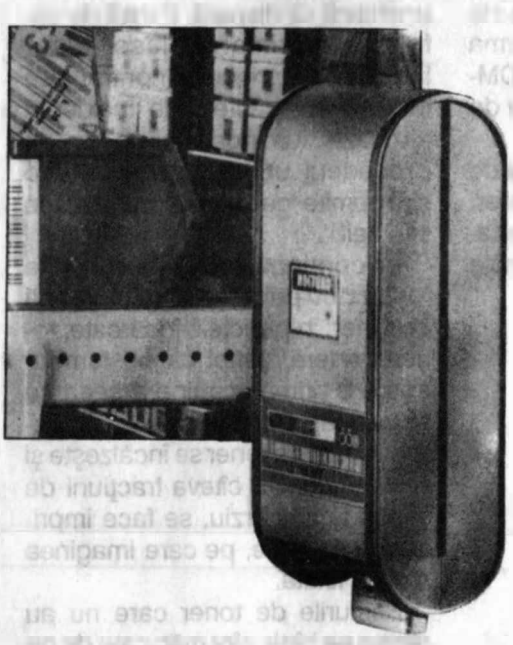
- case de marcaj în magazine cu autoservire
- inventarieri
- identificarea de pachete, paleţi, scrisori, acte, cărţi, fotografii, probe de date de producţie, administraţie, laboratoare



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

Aplicaţii:

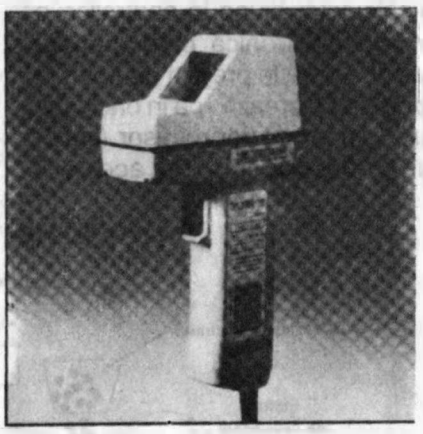
- preluarea şi memorarea datelor "pe teren" - inventare - consignaţie - preluare de comenzi - preluarea unor date de producţie - supravegerea producţiei - pontaj



Laser-scanner cu He-Ne

Aplicaţii:

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



Pistol de citire cod de bare

Aplicaţii

- citirea fără atingere a diverse coduri de bare, în toate domeniile tehnicii şi comer1ului
- preluarea de coduri aplicate pe hîrtie, carton, etichete, suport textil, în poziţii oarecari
- casă şi preluare de marfă
- identificarea unor părţi de montaj pe linii de asamblare

Cum funcţionează imprimantele laser?

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

Primul în şiragul de "muncitori" cu înaltă calificare este calculatorul, pe care utilizatorul îşi scrie textul. Din clipa în care utilizatorul vrea să-şi imprime textul, interfaţa calculatorului preia aceste date sub formă de semnale electrice şi le transmite direct interfeţei imprimantei. Important e desigur ca cele două interfeţe să se înţeleagă şi să transmită (respectiv să preia) datele fie în şir 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 şi în calculator, aşa-numita memorie nevolatilă (ROM - Read Only Memory). În ROM-uri sînt stocate programele care controlează funcţionarea 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-şi construiască întîi imaginea unei pagini întregi, înainte de a putea imprima chiar şi 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, deşi pe hîrtie constau deasemenea din puncte individuale, au nevoie de foarte puţină memorie. Căci forma lor e deja preprogramată în ROM-uri, astfel că la nevoie se preia de acolo.

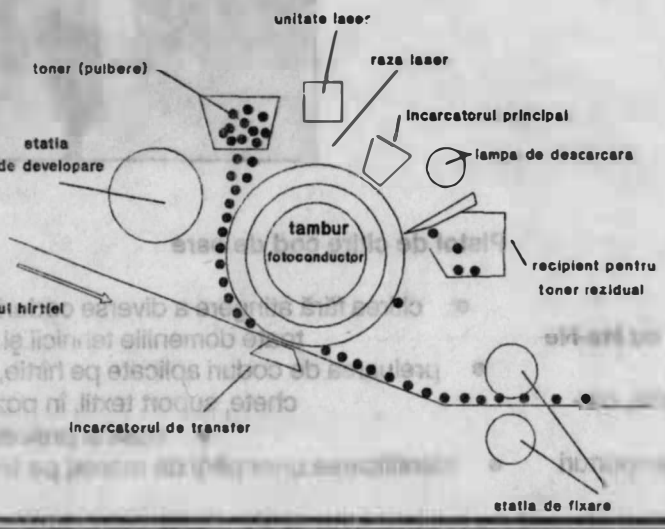
Din clipa în care toate datele de la calculator au ajuns la controller, acesta începe cu calculele necesare pentru a le transforma în puncte individuale. Căci pentru imprimanta laser, pagina imprimată nu constă din litere, ci din aproximativ 8.000.000 de puncte. Linie după linie de puncte este pregătită de controller şi trimisă unităţii de expunere. Nucleul acesteia este un tambur cu strat superficial fotosensibil, lat cît o pagină tipărită (de regulă A4). Acest tambur se roteşte şi este încărcat electric de încărcătorul principal. Rotaţia 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 aşezate pe o "stinghie" una lîngă alta şi 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 şi imprimante laser care lucrează pe principiul contrar - se încarcă în punctele de incidenţă ale razei luminoase. Funcţie de procedeul utilizat, imprimantele sînt numite "ce scriu negru" sau "ce scriu alb".

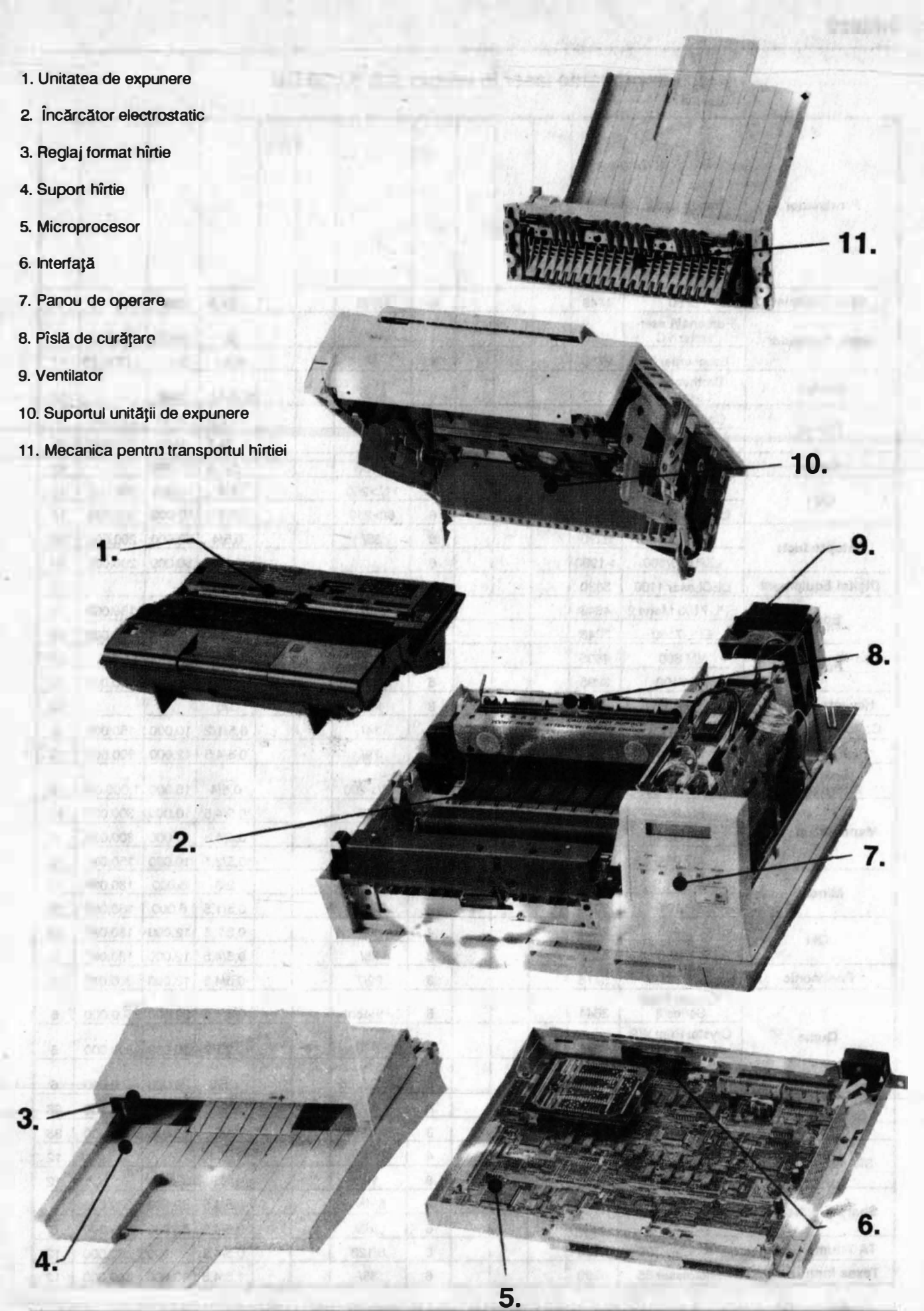
În continuare, tamburul este "prăfuit" cu pulberea extrem de fină de toner; în punctele încărcate, tonerul aderă, astfel că pe tambur apare o primă imagine a ceea ce se va tipări. Pentru aceasta, pulberea uscată de toner se încălzeşte şi se topeşte. Cu cîteva fracţiuni de secundă mai tîrziu, se face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

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

Principiul de funcţionare



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

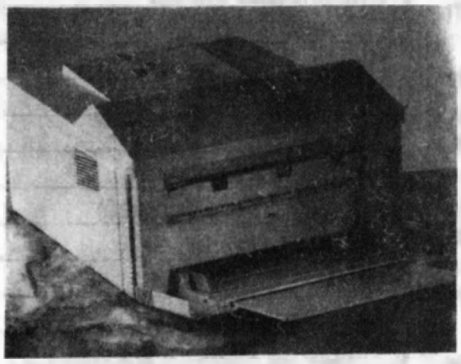


Imprimante laser la preţuri sub 5.000 DM

Producător	Imprimantă												
AEG Olympia	LP60	4446				6	49/10			0,5/4,5	6000	180.000	6
Apple Computer	Personal/Laser-writer 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 preţul avantajos al predecesoarelor sale.



Cu Laserjet IIISi, Hewlett-Packard şi-a delimitat poziţia în segmentul superior al seriei III. Cu noul Laserjet IIIP, producătorul îşi rotunjeşte seria şi în jos. La preţul de vînzare de 3500 DM, Hewlett-Packard doreşte să-şi accentueze poziţia sa de lider al pieţii şi în segmentul preţurilor coborîte.

Cu excepţia efigiei, noul Laserjet IIIP nu se deosebeşte cu nimic, ca aspect exterior, de predecesorul lui IIP. Toate detaliile carcasei au rămas aceleaşi. Şi panoul operator - care în practică s-a dovedit reuşit - a fost preluat întocmai.

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

În mod normal, alimentarea cu hîrtie se face din faţă, folosind o casetă de alimentare multifuncţională, ce are o capacitate de 70 de pagini şi se potriveşte şi pentru plicuri. Deşi Laserjet IIIP nu este concepută ca o imprimantă de mare performanţă, ci ca o imprimantă cu volum mediu de tipărire, rezerva de 70 de pagini este mai degrabă insuficientă. Caseta opţională de hîrtie, ce poate fi amplasată sub imprimantă, rezolvă două probleme deodată. Pe de o parte nu mai este necesară caseta obişnuită de alimentare cu hîrtie, suprafaţa ocupată reducîndu-se astfel la 34x40,5 cm. Pe de altă parte rezerva de hîrtie creşte la 250 de pagini. Mecanica pretenţioasă utilizată face ca această casetă opţională să fie suficient de scumpă: 400 DM.

Multe detalii constructive evidenţiază faptul că IIIP este un produs 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 dim imprimantă să mai antreneze în mişcare paginile deja tipărite.

Instalarea casetei de toner, care conţine toate materialele consumabile, se face prin faţă. Caseta este surprinzător de mică şi poate fi montată dintr-o singură mişcare. Ea este suficientă pentru cca. 3500 de pagini şi 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 preţuri din categoria imprimantelor ce nu sînt capabile de Postscript.

Legarea imprimantei se poate face fie serial, fie paralel. Ambele interfeţe fac parte din configuraţia standard. Tot în configuraţia standard - lucru neobişnuit la imprimante din această categorie de preţuri - 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 şi, 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 aplicaţiile), imprimanta poate fi folosită la parametrii maximi fără drivere suplimentare.

Cele mai importante noutăţi la PCL 5, faţă de PCL 4, sînt fonturile scalabile , grafica vectorială, direcţia de imprimare, modelul de imprimare şi 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 şi 999,75 puncte. Cînd un program solicită o anumită dimensiune, aceasta este calculată şi 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 şi ca plotter. Funcţiile de grafică vectorială pot fi combinate cu celelalte funcţii din PCL 5.

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

Dacă pînă acum totdeauna era problematic să reuşeşti scrieri în direcţii distincte pe aceeaşi pagină, acum acest lucru este uşor realizabil cu PCL 5. Orice text poate fi imprimat în patru direcţii, rotite cu cîte 90 grade.

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

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

La imagini scanate, de ex., îşi găseşte aplicaţie "îndesirea de rastru". Este vorba despre un procedeu de compresie care, aplicat fişierelor-imagini, economiseşte spaţiu 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 îmbunătăţită a imprimării. Un cip special, propriu HP, selectează şi prelucrează în timp real pachete de 6x2560 puncte, pentru a corecta, de ex., liniile oblice, curbele sau întretăierile de linii. Deşi rezoluţia la Laserjet IIIP este de doar 300 dpi (dot per inch), această prelucrare suplimentară duce la o imprimare vizibil mai bună, variind mărimea şi poziţia 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 obişnuitele îngroşări şi apariţia "bulgărilor" de toner.

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

Îmbunătăţirile se văd cel mai bine la compararea cu imprimarea fără RET. Avantajul este vizibil şi fără lupă şi "aruncă" Laserjet IIIP-ul mult în faţă în ceea ce priveşte calitatea imprimării, la nivelul celorlalte reprezentante ale seriei III.

În ceea ce priveşte viteza de imprimare, IIIP marchează progrese considerabile faţă de modelul II. Procesorul intern (un Motorola 68000) lucrează acum la 16 MHz (10 MHz la modelul II), iar viteza de transfer a crescut de 4 ori.

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

Fără îndoială, cu Laserjet IIIP, Hewlett-Packard confirmă poziţia sa de lider pe piaţa imprimantelor laser, echilibrînd acum şi sub aspectul preţului situaţia care avantaja, uneori, concurenţa.

(I.F.)

Caracteristici

Nume	**HP Laserjet IIIP**
Preţ (inclusiv MwSt.)	cca. 3500 DM
Carcasa	
Dimensiuni (L x î x l în cm)	34 x 21 x 63
Date tehnice	
Tehnica imprimării	Laser
Unitatea de imprimare	Canon
Rezoluţia (text şi 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 viaţă	nelimitată
Interfeţe	
Paralelă	da
Serială	da
Apple Talk	optional
Altele	RS242 optional
Firmware	
Familii de fonturi	Courier, Line Printer, Times, Univers
Număr de fonturi	67
Emulări	HP Laserjet III
Opţiuni hîrtie	
Formate	DIN A4, C5
Greutatea (g/m^2)	60 pînă la 105
Rezerva	70 / 320 pagini
Antrenare	manuală / automată
Ieşire (face-down / face-up)	70 / 20 pagini
Materiale consumabile	
Toner	-
Tambur	-
Unitatea de încălzire	-
Cartuş toner/tambur	3500 pag., cca. 240 DM
Costuri	
Preţ pe pagină (cu / fără preţul aparatului)	7,61 / 6,87 pfenning
Altele	
Garanţie	12 luni
Accesorii	Program hardcopy
Manuale	în engleză

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

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

Chiar dacă sînt prezente într-un fişier care se încarcă ca un "device driver" normal, funcţiile XMS nu sînt apelate prin metodele tipice driver-elor DOS, ci invocînd Int 2fH cu coduri de funcţie 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 părţi din memoria aflată deasupra primului megabyte, care teoretic ar trebui să fie inaccesibilă. După cum este ştiut, adresele utilizate de 8086 sînt codificate pe 20 biţi, fiind compuse dintr-un segment şi 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 referinţă la adresa 08759H. 80286 foloseşte adrese pe 24 biţi şi nu ignoră transportul obţinînd o referinţă la adresa 108759H, deasupra primului megabyte.

Pentru a obţine compatibilitatea cu programe care au fost scrise pentru 8086 (şi care în unele cazuri se bazau 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 totuşi 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.

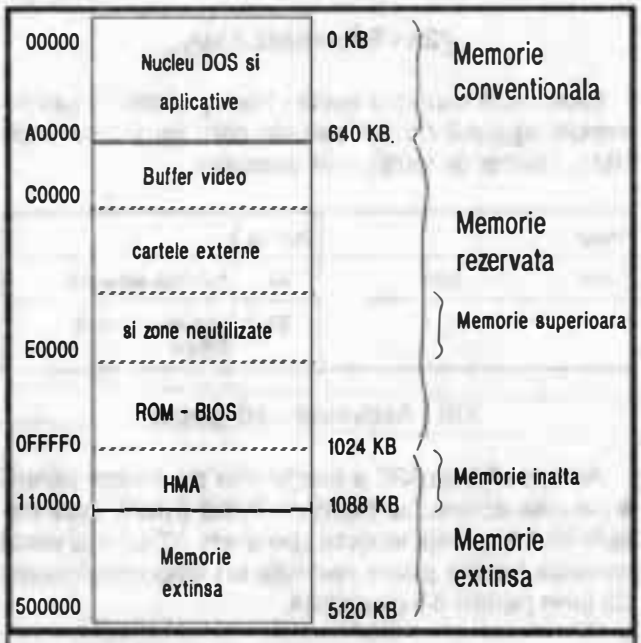
Termeni noi introduşi de specificaţiile XMS - L/I/M/A:

- **-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 şi a unei serii de adrese altfel neutlizate.
- **-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 funcţiile XMS şi dacă să permită sau nu utilizarea memoriei înalte;
- - utilizatorul poate asigura instalarea funcţiilor înaintea oricărui program care le-ar putea apela;
- - producătorii pot realiza "device driver"-e diferite.

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



Organizarea memoriei conform specificaţiilor XMS

Sumarul funcţiilor XMS

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

00h - Citeşte 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) şi o alocă dacă aceasta e disponibilă şi 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 - Elberează HMA

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

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 funcţie 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 readucînd accesul la modul 8086, în care adresa FFFFh:0010h coincide cu 0000h:0000h; trebuie să fie apelată de programe care au controlul HMA şi au activat linia A20, înainte de terminarea execuţiei; execuţia 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; execuţia 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, readucînd accesul la memorie la modul 8086, În care adresa FFFFh:0010h coincide cu 0000h:0000h; această funcţie trebuie să fie apelată de programe care să fi activat anterior linia A20 înainte de terminarea lor; execuţia 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 şi memoria extinsă liberă totală, excluzînd 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 şi 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 moderaţie.

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 funcţie trebuie apelată înainte de terminarea programului, pentru a evita ca blocul respectiv de memorie să rămînă permanent alocat şi inutilizabil de către alte aplicţii.

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 convenţională; 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 convenţională, fie sursă fie destinaţie se atribuie valoarea zero handle-lui 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 parametrii 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 destinaţie
DD	deplasament în interiorul blocului destinaţie, în bytes

0C h- Protejează un bloc de memorie extinsă

Împiedică deplasările în cadrul blocului şi returnează adresa liniară de 32 biţi a unui bloc anterior alocat; această funcţie este util 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 biţi
		BL	Cod eroare dacă AX< >1

0Dh-Deblochează un bloc de memorie extinsă

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

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

0Eh- Citeşte informaţii despre un handle XMS

Furnizează informaţii 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 biţi) 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 funcţie 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 prezenţei driver-ului XMS şi apelarea lui

Înainte ca o aplicaţie să apeleze funcţii XMS trebuie obligatoriu să verifice prezenţa driver-ului în memorie. Acesta se poate face conform următoarei scheme:

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

Sumarul codurilor de eroare

Încontinuare este prezentat tabelul tuturor codurilor de eroare XMS posibile, furnizate de funcţiile apelate totdeauna în registrul BL cînd AX = 0.

Cod	Descriere
00h	Nu este eroare ;
80h	Funcţie 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 destinaţie eronat
A6h	Deplasament destinaţie necorespunzător
A7h	Lungime incorectă
A8h	Deplasament cu depăşire
A9h	Eroare de paritate
AAh	Bloc neprotejat
ABh	Bloc protejat
ACh	Depăşire număr de blocuri permise
ADh	Protejare imposibilă
B0h	Memorie superioară insuficientă
B1h	Nu există memorie superioară disponibilă
B2h	Segment eronat

Consideraţii pentru dezvoltarea de aplicaţii

Consideraţiile următoare ar trebui urmate în proiectarea oricărei aplicaţii care face apel la interfaţa 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 direcţioneze în memoria înaltă, întrucît dacă linia A20 s-ar dezactiva ulterior, vectorii respectivi n-ar mai putea fi atinşi şi gestiunea întreruperilor s-ar pierde.

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

Biblioteca a fost testată cu Turbo Pascal 6.0 şi driver-ul XMS aparţinînd lui Microsoft - HIMEM.SYS ver 2.0 (care s-a dovedit a nu suporta toate funcţiile specificaţiei XMS: lipsesc funcţiile de acces la memoria superioară), pe următoarele configuraţii : 386 cu ISA şi 8 MByte RAM şi 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 low byte, 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
```

```
    _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;
    GetA20Status:   boolean;
                    { top of conventional memory: }
    HMATestArray:       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 totaly 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.';
```

```
        $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 succesfully 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
                        HMATestArray[i] : = i mod 255;
                    i : = 1;
                    while (HMATestArray[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;
    write('Allocating ',MaxFree,' kbytes of extended memory...');
    AllocExtMem(MaxFree, handle);
    if XMSError > 0 then Error
    else
    begin
        writeln('OK.');
        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;
        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
            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 puţin 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 experienţă în COBOL cunoaşte desigur punctele slabe care pot servi ca şi contraargumente pentru folosirea COBOL-ului - le vom trece pe scurt în revistă, încercînd să arătăm că mai puţin limbajul ca atare cît alte circumstanţe sînt cauze ale disputei şi că avantajele precumpănesc cu mult.

Să începem cu lucruri elementare: manualele de referinţă sînt toate, în limba engleză. Mai mult: autorii acestora ţin la o completitudine deseori exagerată şi informaţii 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 desăvîrşire.

Chiar dacă unele compilatoare COBOL sînt însoţite de complexe programe demonstrative - rămîne de văzut dacă găsiţi acele pasaje care sînt interesante pentru munca Dvs. Chiar dacă aţi studiat informatica, dacă e să vă obişnuiţi cu un nou compilator, ar trebui să prevedeţi o perioadă de experimentări destul de lungă şi mult timp "pierdut".

În afară de aceasta, deseori listinguri kilometrice cu "algoritmi" îmbîcsiţi şi total nestructuraţi după criteriile moderne, sperie cititorul. Critica aproape că nu are sens; pentru că acolo unde se programează strict în COBOL, generaţii elitiste ţin cu dinţii de tradiţii, iar cei care intră sînt nevoiţi să se supună cu stricteţe "standardelor" specifice întreprinderii. Pentru programe regîndite şi cu deschidere spre viitor - posibile şi în Cobol, în ciuda afirmaţiilor contrare! - pur şi simplu nu rămîne timp.

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

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

Dar să trecem şi la avantaje. Cînd este vorba de prelucrarea rapidă şi exactă pînă la cîteva poziţii după virgulă a unor fişiere extrem de mari, cu tabele multidimensionale şi indexate, Cobol-ul este una din puţinele unelte de programare ce garantează succesul. Un motiv important care a dus la apariţia Cobol-ului a fost cerinţa de a putea face cele mai variate selecţii în fişiere oarecare. De aceea, Cobol-ul dispune de un sofisticat mecanism de control pentru fişiere şi 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 şi constantele se divid în alfabetice, alfanumerice, numerice şi editate numeric şi se detaliază pînă la ultimul octet. Numai definirea 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 poziţii, corespunde nivelului 77, adică nici nu are alte date subordonate şi nici nu se subordonează vreunei alte grupe de date. La afişarea pe ecran sau la imprimare, zerourile premergătoare din poziţiile marcate prin "Z" vor fi înlocuite cu blancuri. LEI are două poziţii zecimale, e formatată în mii şi iniţializată cu valoarea 0. (Printr-o clauză în program, scrierea americană a numărului poate fi înlocuită cu scrierea europeană, cu virgulă pentru partea zecimală şi punct ca separator de mii).

Există şi variabile booleene. Ataşate nivelului "88", ele pot corespunde în tabele şi mai multor condiţii - deci unor mulţimi!

Cu puţine excepţii sintaxa instrucţiunilor 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  |  fd 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 s9v9(9)    value 3.141592654.
41  |  77  x           pic s9v9(10).
42  |  77  s           pic s9v9(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  |      perform Domeniu until x > 3.14.
64  |      open output File.
65  |      if IOResult not = zero
66  |          display "1. Eroare in fisier: ", IOResult
67  |          accept Tasta
68  |          move "1" to Proba
69  |      else
70  |          move zero to ZPos
71  |          perform Memorare until ZPos = 25.
72  |      close File.
73  |      if not Eroare
74  |          open input File.
75  |      if IOResult not = zero
76  |          display "2. Eroare in fisier: ", IOResult
77  |          accept Tasta
78  |      else
79  |          move zero to ZPos
80  |          perform Reciteste until ZPos = 25.
81  |      close File.
82  |  end.
83  |      stop run.
84  |
85  |  Domeniu.
86  |      move zero to i.
87  |      perform Calcul until i = 10.
88  |      compute SPos rounded = x / Pi * 39 + 40.
89  |      compute ZPos rounded = s * 11 + 13.
90  |      move "x" to Car (ZPos, SPos).
91  |      add 0.1 to x.
92  |      move 0 to s.
93  |      move 1 to f.
94  |  Calcul.
95  |      compute f = f * (i * 2 + 1).
96  |      compute s = s + ((-1) ** i) * (x ** (2 * i + 1))/
97  |      compute f = f * (i * 2 + 2).
98  |      add 1 to i.
99  |  Memorare.
100 |      add 1 to ZPos.
101 |      move ZPos to DataKey.
102 |      move zero to SPos.
103 |      perform Aduna until SPos = 79.
104 |      write DataRecord invalid key
105 |          display "Eroare de scriere: ", IOResult
106 |          accept Tasta.
107 |  Aduna.
108 |      add 1 to SPos.
109 |      move Car (ZPos, SPos) to DataChr (SPos).
110 |  Reciteste.
111 |      add 1 to ZPos.
112 |      move ZPos to DataKey.
113 |      read File invalid key
114 |          display "Eroare de citire: ", IOResult
115 |          accept Tasta.
116 |      move zero to SPos.
117 |      perform Afisare until SPos = 79.
118 |      display Matrice.
119 |  Afisare.
120 |      add 1 to SPos.
121 |      mov DataChr (SPos) to Rlesire (SPos).
122 |  *END OF BENCH.COB
123 |
```

Figura 1.

bazează pe vocabularul exact definit prin standardul ANSI-85, care este realmente foarte cuprinzător şi acoperă mulţumitor cele mai importante variante de programare.

Cu toate acestea, la o privire atentă portabilitatea se dovedeşte limitată. Puteţi desigur porta fără probleme un program Cobol din DOS spre Unix sau OS/2, şi după poziţionarea corespunzătoare a comutatorilor de compilare şi a altor opţiuni, obţineţi, într-un timp rezonabil, o variantă care să meargă - totuşi, probleme de adaptare apar întotdeauna.

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

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

Un exemplu posibil - un programator liber profesionist întreţine 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 învăţat, a învăţat utilizînd VS-Cobol de la IBM. Programatorul unei alte firme, care lucra în RM-Cobol, a renunţat la servici, drept pentru care probabil vor apare comenzi şi 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 comparaţie între diversele compilatoare, calculează sinusul folosind cele 4 operaţii aritmetice elementare, face apoi două ridicări la putere, scrie rezultatele sub formă de tabel într-un fişier, reciteşte fişierul şi reprezintă pe ecran o curbă stilizată. Nu e vorba desigur de un program deosebit de util, cît de a folosi instrucţiuni Cobol cît mai variate şi de a observa "reacţia" diferitelor compilatoare. Singurul aspect tehnic este

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

Dificultăţi fundamentale nu există.

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

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

La MS-Cobol 3.0, ecranul nu se mai şterge automat; programul funcţionează însă impecabil. La fel şi la Microfocus şi Realia Cobol. Acestea sînt, ambele, protejate prin "dongles", lucru mai puţin stînjenitor 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 interfaţă paralelă şi 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 şi Microfocus Cobol sînt aproape 100% compatibile. Însă în loc de IXSIO, Microfocus la linkeditare are nevoie de biblioteca EXTFX.

Acu-Cobol, exact ca şi 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 şters în mod automat.

Nici la MBP, ecranul nu e şters. Instrucţiunea ACCEPT produce automat un semnal sonor.

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

Probleme ridică, în general, instrucţiunile ACCEPT şi DISPLAY. Deoarece aceste instrucţiuni sînt prea puţin standardizate, fiecare producător "condimentează" după gustul propriu aceste instrucţiuni. 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 aplicaţiile scrise, ar trebui să renunţaţi din principiu la atributele ACCEPT şi DISPLAY specifice unui producător şi să utilizaţi în schimb interfaţa prin CALL. Prin aceasta, producătorii de compilatoare oferă posibilităţi de programare deosebit de individualizate - astfel, rutinele confortabile de tratare a ecranului rămîn locale şi se pot adapta independent de restul programului, "dintr-o singură mişcare".

(I.F.)

Cobol	Comenzile necesare lansării programului din fig.1	Timpul consumat (secunde)	Preţ (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 ixsio 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-şi aduce aminte de primul compilator Cobol pentru PC va rămîne perplex în faţa celei mai noi versiuni. Editoarele confortabile şi multele programe utile uşurează foarte mult munca programatorului profesional. În sfîrşit te poţi 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 direcţia corectă. Fiind echipată cu o multitudine de instrumente de programare utilitare, versiunea 4.0 îşi merită pe deplin numele de PDS: Professional Development System.

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

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

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

Dacă totuşi lipseşte din PWB o funcţie 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 fişierul TOOLS.INI - Dacă este necesar, atunci un macrou propriu poate recurge la funcţii deja existente în PWS sau la alte macrouri.

Programatorii ştiu să aprecieze proprietatea de memorare a compilatorului la terminarea lucrului. La reluarea unui proiect de program se poate continua cu aceleaşi condiţii.

Help-ul on line este exemplar, el înlocuieşte căutarea repetată în manualul de utilizare. Dacă cursorul este plasat pe un cuvînt cheie Cobol, la apărarea tastei F1 apare o nouă fereastră. Această fereastră conţine toate informaţiile importante cu explicaţii aferente funcţiunii şi o descriere detaliată a sintaxei.

În domeniul detectării erorilor, Cobol 4.0 oferă multe instrumente de ajutor. "Source Browser" vizualizează informaţii importante despre procedurile existente şi despre legăturile variabilelor introduse. Singura condiţie este folosirea comutatorului corespunzător care duce la crearea unui fişier 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 fişier conţine toate informaţiile 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 - Debbuger. El poate fi iniţiat direct de pe suprafaţa utilizator dar el nu poate fi mînuit prin mouse.

Nou în comparaţie cu versiunea anterioară este şi generatorul de măşti integrat (screen-uri). Astfel măştile pentru programare se pot crea mult mai uşor. Nici generatorul de măşti nu poate fi deservit cu mouse.

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

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

Avantajele sistemului OS/2 sînt clare. Linkeditarea dinamică face ca toate subprogramele linkeditate corespunzător să devină părţi ale DLL (Dynamic Link Library) a OS/2. Asfel fişierele executabile obţinute sînt mult mai compacte şi mai rapide, deoarece părţile necesare sînt activate doar în timpul execuţiei programului. Astfel se face şi o economie serioasă de memorie.

```
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 Sfirsit
```

Iniţializarea mouse-ului în Cobol

Pentru simplificarea programării de aplicaţii pentru Presentation Manager există programul suplimentar H2CPY. Acesta traduce fişierele de headere 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 şi posibilitatea lucrului cu SQL. Arhitectura client-server a fost îmbogăţită în noua variantă cu posibilitatea de comunicare cu eventuali serveri de baze de date SQL. Compilatorul permite integrarea instrucţiunilor SQL în programele Cobol cît şi interogarea directă a bazelor de date SQL.

Pentru orice proiect suprafaţa utilizator generează o bază de date proprie proiectului, iar Cobolul memorează şi administrează toate informaţiile 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 numeşte NMAKE. El recunoaşte imediat ce părţi de programe dintr-un anumit proiect au rămas nemodificate şi se ocupă doar de părţile care au suferit o modificare de la ultima prelucrare. O altă versiune specială a acestui program, denumită NMK reuşeşte să genereze şi 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 şi funcţiuni noi pentru programarea mouse-ului. Modul de iniţializare 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 şi 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 dorinţă neîndeplinită. În manuale există chiar şi 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 şi 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 şi texte noi.

(I.M.)

Încă o dată mai rapid

Foxpro 2.0

În cursa bazelor de date, Foxpro 2.0 a luat din nou un avans în faţa concurenţilor dBase IV şi Paradox 3.5.

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

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 şi 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 îmbunătăţi sensibil performanţele încă o dată. Cu aceasta Foxpro 2.0 nu numai că doboară două muşte cu o singură lovitură, dar foloseşte şi puternica arhitectură a magistralei de 32 de biţi.

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

Interogările Select se autooptimizează în funcţie de dimensiunea tabelelor, indexurile disponibile, mărimea memoriei de lucru şi a spaţiului disponibil pe harddisk etc.

Comanda poate crea indexuri temporare pentru a mări performanţele. Acest SGBD poate fi astfel recomandat şi pentru utilizare în cazul lucrului în reţea.

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

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

Foxpro 2.0 cunoaşte acum trei tipuri de indexuri: indexul standard (.IDX) al versiunilor Foxbare şi Foxpro 1.XX, noul "Compound Index" (.CDX) analog indexului "Multiple Index File" din dBase şi indexul compact. Fişierele de indexuri conţin mai mulţi pointeri. Suplimentar există un aşa-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 fişierele index la cca. o şesime din dimensiunea lor iniţială. Chiar şi acest lucru măreşte viteza de execuţie, deoarece fişierele care trebuiesc 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 invidioşi pe concurenţi ca dBase IV sau Paradox 3.5. Pentru acei care pun preţ 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 şi cu dBase III şi IV.

Comparativ cu standardul pieţii, versiunea beta a lui Foxpro 2.0 nu a strălucit la modul de comandă al imprimantelor. În meniul de imprimare există cîteva comenzi rudimentare pentru stabilirea marginilor, a orientării, etc., dar driver-ele speciale pentru imprimante matriciale şi laser lipsesc. Dacă se doreşte, de exemplu, listarea din Foxpro 2.0 pe o imprimantă HP-Laserjet, terminatorii de linie şi alte secvenţe de întreruperi trebuie realizate cu secvenţe de comenzi. Pentru profesionişti acest lucru nu constituie o problemă, dar începătorilor nu le este tocmai uşor.

După informaţiile de la Foxsoftware, pînă la lansarea pe piaţă, în această vară, vor mai fi aduse cîteva îmbunătăţiri la acest modul.

Generatorul de rapoarte, controlabil în întregime prin meniuri, poate lucra cu variabile, cu cîmpuri suplimentare de calcul (abatere, varianţă) şi poate repeta antetul de grup pe fiecare pagină. Salutară este şi opţiunea care permite restartarea numărului paginii la schimbarea grupului. Definiţia unui raport poate fi memorată.

Introducerea şi poziţionarea de cîmpuri şi de texte în interiorul unui raport se poate face comod cu ajutorul mouse-ului.

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

Pentru operaţiile obişnuite cu fişiere utilizatorul poate utiliza managerul fişierelor care permite copierea, ştergerea, mutarea, fişierelor sau a unor întregi directoare.

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

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

După cum se ştie, Foxpro nu este doar un SGBD foarte uşor de învăţat şi foarte rapid ci şi un instrument confortabil pentru proiectanţii de aplicaţii cu baze de date. Pentru aceştia din urmă Foxpro 2.0 are implementate cîteva noutăţi 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 şi funcţii pe care propriul limbaj de programare - un fel de dBase extins - nu le poate realiza. În plus pot fi utilizate acum şi biblioteci "externe" standardizate, ceea ce duce desigur la ridicarea productivităţii muncii. Avantajoasă este şi utilizarea generatoarelor de machete şi meniuri.

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

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

Foxpro 2.0 este un produs reuşit, noua versiune fiind recomandabilă nu numai ca o îmbunătăţire pentru fanii de Foxpro ci şi 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
Preţ (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 cîmpuri	255	255	255
Limbaj de programare	da	da	da
Suport SQL	da	da	cu SQL-Link
Protecţie prin parolă	da	da	da
Integrare de imagini	da	nu	nu
Suport Windows 3.0	nu	nu	nu
Manager fişiere	da	nu	nu
Tehnica ferestrelor	da	nu	nu
Tehnici speciale de indexare	da	da	nu

Recunoaşterea scrisului

Reţele neuronale

Partea a IV-a

În ultimii ani, recunoaşterea scrisului prin calculator s-a perfecţionat mereu. Recunoaşterea de caractere prin intermediul reţelelor neuronale este un pas într-o direcţie promiţătoare.

Cuprinsul cursului:

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

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

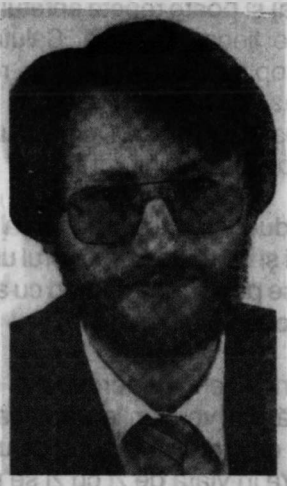
Dacă însă procesul de recunoaştere care are loc în timpul citirii trebuie executat de maşină, atunci în primul rînd trebuie "transformat" în informaţie numerică suportul iniţial - această pagină de revistă de exemplu. Acest lucru se face prin intermediul unei camere video sau cu un scanner, care parcurg rînd cu rînd suportul de pornire, preluînd informaţii cu ajutorul unor fotodiode. La echipamente ce lucrează alb-negru, diferenţele luminos - întunecat sînt transformate pentru reprezentare pe ecranul calculatorului în succesiuni de biţi: un bit setat corespunde unui punct întunecos, un bit nesetat unuia luminos. Deoarece aceleaşi litere, ca succesiuni de biţi, apar diferit funcţie de formă şi mărime, numai un algoritm performant este capabil să realizeze interpretarea.

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

Recunoaşterea corectă a caracterelor

Sarcina programelor OCR (Optical Character Recognition) este de a simula cît mai bine aceste caracteristici umane. În multe programe îşi găsesc aplicaţie mai ales două metode: suprapunerea de modele (pattern matching) şi identificarea de caracteristici (feature recognition). La suprapunerea de modele (compararea de forme - pattern matching) succesiunile de biţi generate de scanner sînt comparate, pixel cu pixul, cu litere "ideale", memorate în calculator. Deoarece caracterele scanate sînt însă foarte diferite în mărime sau lăţime, ele trebuiesc mai întîi "normalizate" la dimensiuni unitare. Chiar şi în acest caz, imagini binare ale unui aceluiaşi caracter pot fi foarte diferite (vezi fig. 1). Soluţionarea acestei probleme se încearcă prin memorarea mai multor modele pentru fiecare literă şi admiterea unui domeniu de toleranţă. O toleranţă prea mare duce însă la identificări eronate. Dacă se memorează prea multe modele, devine necesară multă memorie şi creşte timpul de calcul.

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



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

sînt caracterizate printr-un drum închis. Înainte de a putea identifica literele prin caracteristici, secvenţele de biţi trebuiesc "skeletonizate". (Adică, de exemplu, dintr-o linie de 3 pixeli se obţine una de un singur pixel).

În ciuda anunţurilor de reuşită a diverşi producători, suporturile copiate le creează dificultăţi majore 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 şi 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 performanţele altor procedee, reţelele neuronale îşi dovedesc eficienţa mai ales la identificarea caracterelor deformate sau "sparte".

Împreună cu programul de simulare NBP se livrează un exemplu de recunoaştere a scrisului cu care se pot face experimentări. Intrarea reţelei constă din 18x18 pixeli iar ieşirea din 99 de celule, pentru recunoaşterea 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.

Fişiere de pixeli scannate în format IMG pot fi reprezentate pe ecran cu ajutorul unui program specific; acesta decupează litere dintr-un text şi le memorează într-un fişier propriu. Este posibilă şi micşorarea imaginii, pentru a reduce volumul de date. Funcţie de rezoluţia scannerului, pentru un caracter se obţin mai mulţi sau mai puţini pixeli. Caracterele memorate se folosesc ca mostră de antrenament pentru o reţea neuronală. Cum s-a mai spus, fiecărei mostre i se ataşează un model de ieşire (respectiv, apartenenţa la o celulă de ieşire). Deci de exemplu mostra de pixeli pentru "A" activează celula de ieşire 1, mostra "B" celula de ieşire 2 ş.a.m.d.

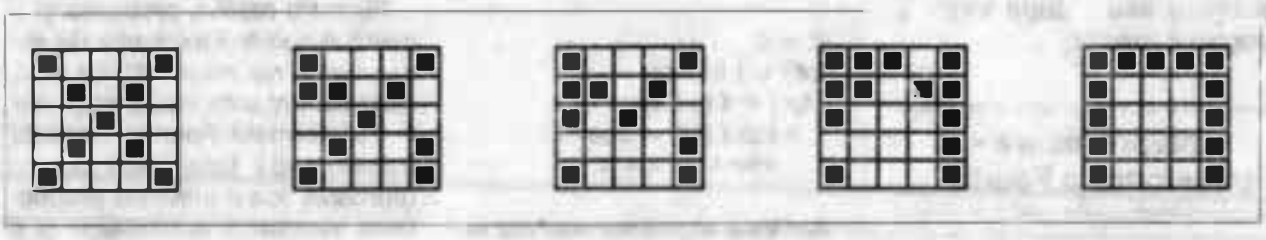


Figura 1 arată dificultăţile interpretării mostrelor de pixeli. Secvenţa reprezintă transformarea unui X într-un U. Dacă al doilea model din stînga încă poate fi recunoscut ca "X" şi al doilea model din dreapta încă poate fi interpretat ca "U", la modelul din centru sînt posibile noi interpretări (de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

Aceste informaţii relative la scopul învăţării sînt memorate odată cu mostrele de pixeli. Pot fi imaginate desiguri şi alte codări ca funcţii obiectiv ale reţelei neuronale - de exemplu, direct valoarea ASCII corespunzătoare diverselor caractere. În acest caz ar fi necesare numai 3 celule de ieşire pentru reprezentarea zecimală, respectiv 8 pentru reprezentarea binară. Cercetări proprii au relevat însă faptul că acest mod de reprezentare a informaţiei constituie o îngreunare serioasă a procesului de convergenţă în faza de învăţare, mai ales datorită faptului că prin folosirea funcţiei sigmoide (vezi partea a II a acestei serii) stabilităţi se formează numai în jurul valorilor 0 şi 1. În plus, litere total distincte au coduri binare ASCII asemănătoare, ceea ce se răsfrînge negativ asupra asociativităţii necesare.

Sarcina fazei de învăţare este de a modifica ponderile din reţea în aşa fel încît să apară o configuraţie care satisface caracteristicile cerute ale fişierului de antrenament. Ceea ce deosebeşte reţelele neuronale de alte tehnici este faptul că nu mai este necesară o descriere explicită a algoritmului, astfel că această configuraţie se produce independent, în cursul fazei de învăţare. Proprietăţile ei sînt dirijate prin intermediul mostrelor de antrenament şi al funcţiilor obiectiv asociate.

Avînd odată o reţea "antrenată", datorită asociativităţii reţelelor neuronale, imagini binare asemănătoare ale unor litere sînt clasificate corect. Ceea ce, în cadrul codării menţionate iniţial, înseamnă că se activează o singură celulă de ieşire şi se dezactivează toate celelalte. Prin "mărimea" activităţii există o măsură suplimentară a apartenenţei la o clasă. Ea poate fi utilizată ca definiţie a domeniului de toleranţă: o mostră aparţine unei clase dacă valoarea de ieşire corespunzătoare este peste un anumit prag, iar toate valorile celelalte se găsesc sub un anumit prag. Dificil este însă a-ţi face o imagine clară despre noţiunea pomenită mai sus de "asemănare". Cît de mult trebuie să fie corelate două imagini binare, pentru a fi considerate "asemănătoare"?

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

În primul rînd, există posibilitatea translatării: imaginile de pixeli sînt duse în colţul din stînga-sus, iar coloanele şi rîndurile vide sînt pur şi simplu eliminate.

În al doilea rînd, în imaginile micşorate - şi micşorarea este de fapt o anteprelucrare - se poate varia pragul pentru amplasarea unui nou pixel. Astfel se realizează o subţiere sau - după caz - o îngroşare a literelor.

Anteprelucrare - transformata Fourier

Ca o a treia tehnică este oferită o transformată Fourier normată bidimensională. La intrări pătratice, ea are proprietatea deosebită a invarianţei la translaţie. Ceea ce înseamnă că indiferent unde se găseşte o imagine binară în cadrul unui cîmp de intrare, ea este proiectată totdeauna la fel.

Totuşi, chiar şi transformata Fourier rapidă (FFT) necesită calcule laborioase. (Ea se foloseşte totuşi, în mod standard, în electrotehnică datorită propietăţilor deosebite în analiza funcţiilor periodice - e drept, în acest domeniu, transformata Fourier unidimensională).

Ca suport pentru recunoaşterea 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 soluţii complete pentru probleme speciale, ci doar de introducerea în tehnici noi de prelucrare a informaţiilor, 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 combinaţiei transformată Fourier/propagare înapoi este discutat pe exemplul aflării numărului de pixeli setaţi într-o suprafaţă predeterminată. Fie dată o suprafaţă de 5x5 pixeli. În această suprafaţă de 25 pixeli se admit pînă la 5 pixeli "setaţi" - avînd valoare logică "true". Reţelei neuronale i se cere să "numere" corect pixelii setaţi.

Este evident că pe alte căi soluţia se obţine mult mai uşor şi 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, reţeaua trebuie să "găsească" singură algoritmul, pornind de la cîteva (puţine) mostre de antrenament!

Cu NBP se produce o reţea de 5X5 celule de intrare şi 5 celule de ieşire, pentru clasele "un pixel", "doi pixeli", pînă la "5 pixeli".

Problema din ultimul articol, de a minimiza numărul mostrelor de antrenament, este reluată odată cu acest exemplu.

Mai întîi, cîteva consideraţii preliminare relativ la posibilităţile de combinare: pentru a plasa un pixel într-o suprafaţă de 25 de cîmpuri diferite, există 25 de posibilităţi. Pentru 2 pixeli, există 25x24/2 = 300 de posibilităţi, deoarece nu importă ordinea plasării. Corespunzător, pentru 3 pixeli există 25x24x23/3/2 = 2300, pentru 4 pixeli 12.650 şi pentru 5 pixeli chiar 53.130 de combinaţii.

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

Spre deosebire de translaţia pomenită mai sus, la care numai pixelii nesetaţi din stînga şi de sus sînt eliminaţi, valorile cîmpurilor nu mai constau în continuare din valori de "0" şi de "1", respectiv stările "adevărat şi fals", ci din transformate Fourier, deci numere în virgulă mobilă. Avantajul metodei este determinat de toleranţa mai mare la erori - cîte un pixel rătăcit, nedorit, nu are efecte aşa de mari ca şi la translaţie. Dezavantajos este calculul laborios ce trebuie efectuat.

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 micşorată la dimensiunea cerută la intrare, folosind un prag selectabil. Rezultatul micşorării este afişat pe ecran în colţul din dreapta sus.

Probleme cu posibilităţi foarte mari de combinare se abordează cel mai bine în felul următor: după selecţia unei anteprelucrări potrivite, din fiecare clasă (2,3,4,5 pixeli) se memorează cîteva mostre şi se antrenează reţeaua cu acestea. Din pixelii singulari nu există mostre diferite, deoarece transformata Fourier este totdeauna aceeaşi.

reţeaua se comportă aşa cum pare "de bun simţ": activează celula care se apropie cel mai mult de mostra dată, şi anume celula 5.

Fără combinarea cu transformata Fourier, efortul de calcul în faza de învăţare ar fi disproporţionat de mare, deoarece ar fi fost de învăţat şi verificat posibilităţi de amplasare

frunzuliţele 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 decît altele, uzuale?

Răspunsul la aceasta depăşeşte cadrul acestei serii, în care pe baza unor noţiuni de bază

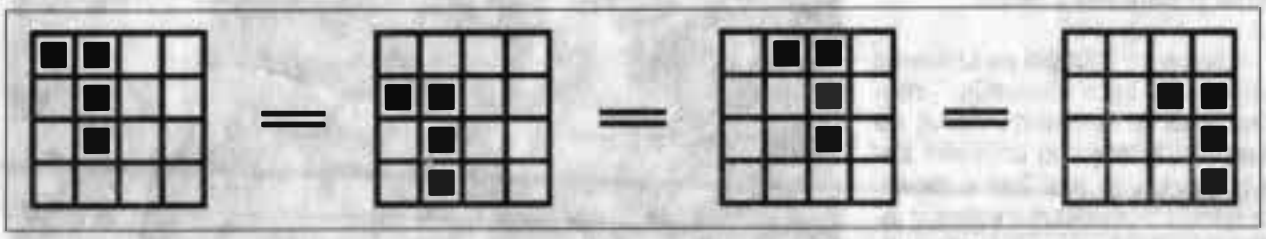


Figura 3. Transformata Fourier prolectează identic modele translatate. Rezultatul transformării se foloseşte ca intrare într-o reţea neuronală. În acest fel se obţine un fel de preclasificare.

După terminarea cu succes a învăţării, se testează comportarea reţelei cu mostre care îi sînt necunoscute. Dacă acestea sînt recunoscute corect, atunci ela au fost deja asimilate de reţea: reţeaua "generalizează" deci datele de antrenament cunoscute. Dacă noile mostre sînt identificate eronat, ele se adaugă la fişierul de antrenament, reluîndu-se faza de "învăţare" a reţelei. Continuînd în felul acesta, se pot acoperi toate combinaţiile 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 combinaţii posibile.

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 obţine reduceri substanţiale de date în fişierul de antrenament şi respectiv timpi de învăţare mult reduşi.

În afară de aceasta, în situaţii neprevăzute - ca de exemplu 6 sau 7 pixeli în cîmpul de intrare -

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

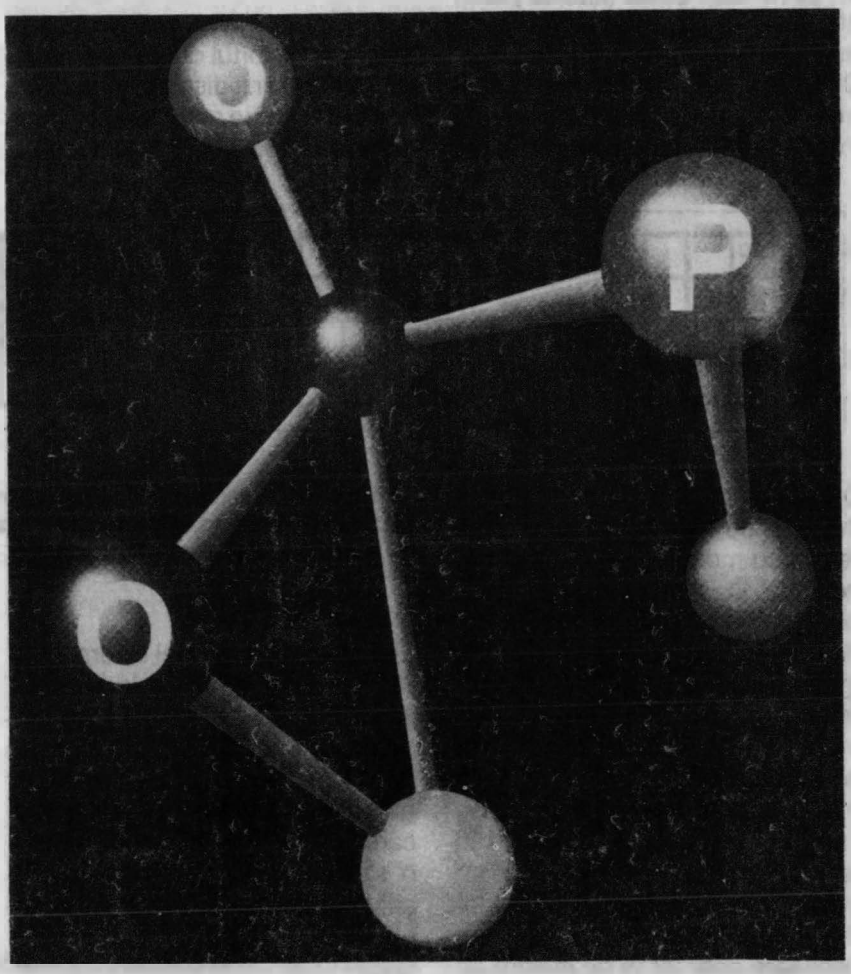
sumare au fost prezentate cîteva exemple de aplicare. Autorul are experienţa unor produse finalizate în aceste domenii şi a realizat cîteva aplicaţii; cei interesaţi pot obţine desigur informaţii suplimentare direct de la el.

Peter Schmitz

Încă o dată de la început ?

OOP - Partea a patra

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



Adesea se întîmplă ca utilizarea rutinelor de intrare/ieşire în Turbo-Pascal să fie incomodă. Astfel, de exemplu, la afişarea unui text trebuie poziţionat mai întîi cursorul, de asemenea editarea valorilor introduse este incomodă. Programatorii se văd mereu puşi în faţa situaţiei de a "scoate" mai mult din rutinele de bază. Ar fi foarte bine deci dacă la nevoie ar putea fi accedat un Unit care să ofere confortul necesar. Ar fi de imaginat rutine care să poată scrie şi citi de pe ecran şiruri de caractere sau de cifre. În acelaşi timp se doreşte ca problema extensiilor să rămînă deschisă. Ulterior urmează ca modul de prezentare al aplicaţiei să fie îmbunătăţit prin introducerea ferestrelor şi a boxelor de dialog.

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

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

Dacă se doreşte programarea orientată obiect a tuturor uneltelor care îmbunătăţesc confortul utilizării intrărilor şi ieşirilor, atunci trebuie găsite similitudinile şi diferenţele. Caracteristicile identice pot fi definite în obiectul mamă, tipurile speciale de obiecte moştenind aceste caracteristici. Similitudinile nu trebuie căutate îndelung. Toate obiectele necesită metode care să le permită prezentarea pe ecran şi îndepărtarea lor de acolo.

Dar este evident că un şir de caractere va necesita o altă rutină pentru reprezentare decît 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" şi reprezintă fundamentul pentru o ierarhie de obiecte.

Comun tuturor obiectelor din Unit este faptul că ele trebuie să poată fi prezentate în poziţia dorită de pe ecran. Obiectul mamă în acest caz ar putea fi **BS_Loc**. El defineşte două tipuri de date **X** şi **Y**. Ele sînt identice cu poziţia ulterioară de intrare sau de ieşire. Metoda **Init** este utilizată pentru stabilirea poziţiei. Cu **GetX** şi **GetY** poate fi interogată poziţia momentană a unui obiect pe ecran. Pot fi imaginate şi alte funcţii - cum ar fi de exemplu poziţionarea cursorului. Din motive de spaţiu ne vom limita însă la funcţiile anterior enumerate.

S-ar putea crede că metoda **Init** este inutilă, ea nu face altceva decît să transmită date, în rest nu se întîmplă nimic. Şi totuşi nu este aşa. Definiţia tipului **Obiect** (partea

întîi a acestui curs) exprimă clar faptul că un obiect este o reuniune a datelor şi 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 influenţate 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 definiţiile obiectelor sînt date în partea Interface, iar partea de cod a metodelor aparţinătoare în partea Implementation. Un utilizator care are la dispoziţie doar partea "publică" a Unit-ului, trebuie să-şi transmită datele spre obiect prin metodele Init disponibile. Altă soluţie nu există. În acest mod datele interne sînt protejate împotriva unor modificări nepermise.

O a doua caracteristică foarte importantă a obiectelor este posibilitatea moştenirii caracteristicilor. Se defineşte deci un nou tip de obiect, care "se trage" dintr-altul. Cu aceasta noul obiect preia toate însuşirile 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 şi 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.

Şi metoda **Move**, pe lîngă două metode proprii, foloseşte şi metoda Init a lui **BS_Loc**.

Cu ajutorul unui mic program de test se poate stabili imediat că şi sarcinile lui **GetX** şi **GetY** sînt înţelese de un obiect de tipul **BS_Semn**. Pe lîngă efectul protejării datelor împotriva unor accese nedorite prin încapsulare, se economiseşte şi cod, deoarece metodele comune nu trebuie redefinite.

Acest prim exemplu (Listing 1) care se sprijină puternic pe primele părţi ale acestui curs este însă foarte subţire încă. Cu excepţia scrierii, ştergerii şi 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** şi **Delete** sînt definite din nou sau altfel exprimate: ele au fost adaptate pentru lucrul cu şiruri. Cu **Get_Str** poate fi stabilită valoarea parametrului intern **Str**. În maniera obişnuită, 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 acelaşi. 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** şi **BS_String.Delete** îşi realizează în totalitate funcţiile. Numai mutarea textelor nu se face tocmai corect. Pe noua poziţie de pe ecran va apare numai primul caracter din şir. Ce este greşit?

Sursa program propriu-zisă este corectă. O instanţiere a lui **BS_String** apelează rutina **Move**, în care prima instrucţiune este **Delete**. Dar despre care metodă de ştergere este vorba? Instanţierea ar dori să lanseze în execuţie metoda **BS_String.Delete**. În loc de aceasta se execută însă cea din **BS_Semn**. Acelaşi lucru se întîmplă şi în următoarea instrucţiune a programului. Din nou este apelată o metodă care nu se potriveşte instanţierii. Acest lucru poate fi explicat astfel: **Move** este o metodă a lui **BS_Semn**, deci compilatorul stabileşte faptul că **Draw** şi **Delete** sînt metode care aparţin lui **BS_Semn**. Urmarea este că numai un caracter al şirului 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** şi care metodă **Draw** urmează a fi utilizată. Aceasta va fi stabilit doar atunci cînd tipul instanţierii apelante este cunoscut.

O metodă pentru toţi

Acest mecanism nu funcţionează însă de la sine. El trebuie iniţializat. 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 instanţiere cu metode virtuale a fost iniţializată. Răspunderea, ca mecanismul metodelor virtuale să fie iniţializat pentru fiecare instanţiere apelantă, cade pe umerii utilizatorului.

Iniţializarea are loc printr-un apel special de procedură. În antetul de procedură cuvîntul **Procedure** este înlocuit cu cuvîntul **Constructor**. Este avantajos să se modifice funcţionarea 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** şi **BS_String**. În ambele obiecte, metodele **Draw** şi **Delete** trebuie să fie virtuale. Este suficientă adăugarea cuvîntului **Virtual** în declaraţie.

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

Fără să fi scos în evidenţă 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 acelaşi nume a obiectului fiu **BS_String**. Cu **Delete** se întîmplă acelaşi lucru.

Această posibilitate, de a denumi acţiuni diferite cu acelaşi nume, este denumită polimorfism.

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

Virtual şi 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 prezenţi, în faţa definiţiilor 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 instanţieri 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 instanţieri atunci ele pot fi create şi eliminate din memorie cu ajutorul procedurilor deja cunoscute **New** şi **Dispose**. Metoda **Done**, inserată, serveşte la curăţarea 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 conţine nici o instrucţiune, ea îşi îndeplineşte menirea. Un Destructor poate fi imaginat ca un Unit, care nu exportă nimic, dar care prelucrează o parte de iniţializare.

Simultan în **BS_Tools** a fost definit un nou obiect: **BS_Intro**. El exportă trei noi date (l_String, l_Attr şi l_Length). Nucleul îl constituie funcţia **ReadString**. Cu ajutorul parametrilor transmişi prin intermediul constructorului **Init** este posibilă afişarea unui prompter într-o anumită poziţie pe ecran şi aşteptarea introducerii unui şir de caractere în dreapta acestuia. **NrMax** stabileşte lungimea maximă a şirului de editat. Este clară, şi în acest caz, utilizarea diferitelor metode moştenite.

Totuşi din motive de spaţiu 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 şi editarea de numere de tipul Real sau LongInt. Obiectul ar putea prelua intern convertirea în formatul dorit. Imaginabilă este şi stabilirea unui domeniu pentru valorile de intrare. Astfel ar putea fi introduse numere şi 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 moştenite nu funcţionează corect. Nefiind utilizabile trebuie definite din nou. Dar şi metoda **ReadString** trebuie modificată în cîteva locuri pentru ca datele interne să fie corect poziţionate.

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 comdiţii, parametrii interni X şi Y trebuie să rămînă nemodificaţi după apelul lui **ReadString**. Deoarece obiectul nu poate fi mutat în timpul editării este suficient dacă şirul prompter şi şirul de intrare sînt aranjate pentru mutare. Ele trebuie colectate deci împreună în parametrul intern Str.

Faptul că **Move** nu funcţionează 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 şi corecta uşor anomaliile: deoarece **Init** a fost declarat constructor, în metoda **Move** a lui BS_Semn (moştenire după două generaţii!) sînt utilizate deja metodele corecte **Draw** şi **Delete**. Problema ţine 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 puţine linii program se poate introduce un şir de caractere în orice poziţie dorită de pe ecran. Dacă trebuie introduse mai multe date pot fi deschise simultan mai multe obiecte. Atît timp cît acestea există în heap, şirul de intrare trebuie să fie oricînd disponibil. El poate fi afişat deci încă o dată la nevoie. De remarcat sînt şi procedurile extinse New şi Dispose. Primul parametru din paranteză specifică instanţierea pentru care trebuie rezervat sau eliberat spaţiu. Al doilea parametru specifică rutina cu care va fi executat procesul.

Obiectele odată definitivate mai aduc un avantaj programatorului. El nu mai trebuie să ştie decît ce fac obiectele. Cum funcţionează obiectele intern rămîne necunoscut utilizatorului şi nici nu-l interesează. Este suficient, de exemplu, să se ştie că obiectul **BS_String** afişează un şir în poziţia dorită de pe ecran. Dacă **BS_String** rezolvă acest lucru prin utilizarea procedurilor standard **Read** şi **Write** sau - aşa cum este cazul lui **BS_Tools** - prin scriere directă în memoria ecran, este neesenţial.

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 particularităţile 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.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;

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[1];
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 i,j : 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,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;

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;
      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
   ClrScr;
   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}
```

Protecţie prin parolă

O parolă eficientă ar trebui să conţină mai multe caractere, să nu fie prea simplă şi să fie schimbată la intervale regulate. Nume, date de naştere sau combinaţii "fără sens" de regulă nu ridică probleme deosebite în faţa unor "hackeri" împătimiţi. Acronimele sînt recomandabile, ale unui vers sau cîntec binecunoscut. De exemplu:

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

în listingul Unlock.asm, această secvenţă e folosită pentru a "deszăvorî" tastatura. Secvenţa e prelucrată, rezultatul prelucrării fiind un cod ce se memorează în parolă. Dacă aţi greşit cumva vreun caracter la tastarea parolei, puteţi oricînd reiniţializa tastînd <ESC>. Din valoarea compilată, este foarte greu de a deduce o secvenţă 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 "zăvoreşte" tastatura, nepermiţînd nici un acces la ea pînă ce nu se rulează Unlock. Cum cea mai frumoasă parolă nu face doi bani dacă fişierul iniţial de comenzi, Autoexec.bat, poate fi întrerupt cu <Ctrl-C> sau <Ctrl-Break>, zăvorîrea tastaturii din Config.sys, face imposibilă intruziunea cuiva care nu ştie parola. Desigur, Unlock ar trebui inclus în Autoexec.bat - altfel, nici Dvs. nu mai aveţi acces la tastatură...Aşa cum vă e prezentat, Lockdrv.sys poate fi folosit de fapt ca un cadru pentru a transforma programe rezidente în memorie şi care preiau anumiţi vectori de întrerupere, în device-drivere. Avantajul metodei: prin modul în care un device-driver se integrează în sistem, el ocupă puţin loc. Dezavantajul: spaţiul ocupat nu mai poate fi eliberat. Compilarea se pote 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  ; citeste 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 programm
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
                  driver-ului
56  |  inter      proc   far
57  |             push   ax
58  |             push   bx
59  |             push   cx
60  |             push   dx
61  |             push   ds
62  |             push   es
63  |             push   si
64  |             push   di
65  |             push   bp
66  |             lds    bx, RQadr
67  |             mov    al, RQptr.cmd
68  |             cmp    al, 0
69  |  black:     je          init
70  |             cmp    al, 4
71  |             je          read
72  |  ; eroare: comanda ineacceptabila
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 |  ; initializare driver
104 |  init:      mov    ax, cs  ; permisa 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,
                                208, 220, 233, 247,
13      262, 278, 294, 312, 330, 350, 370, 392,
                                416, 440, 466, 494,
14      524, 556, 588, 624, 660, 700, 740, 784,
                                832, 880, 932, 988,
15      1048,1112,1176,1248,1320,1400,1480,1568,
                                1664,1760,1864,1976,
16      2096,2224,2352,2496,2640,2800,2960,3136,
                                3328,3520,3728,3952,
17      4192,4448,4704,4992,5280,5600,5920,6272,
                                6656,7040,7456,7904);
18
19  procedure tratimp; interrupt;
20  begin
21      if contor = 0 then
22      begin
23          nosound;
24          indice: = (indice + 1) mod nrnote;
25          indnot: = tabnote ^ [indice,0];
26          if (indnot0) and (indnotU) then
27              sound(valnote[indnot]);
28          contor: = tabnote ^ [indice,1] and 127
29      end;
30      dec(contor)
31  end;
32
33  procedure cintec;
34  begin
35      if nrnote > 0
36      then
37          begin
38          getintvec(28,salv); indice: = nrnote-1;
39          contor: = 0; setintvec(28,@tratimp)
40          end
41      else
42          begin
43              nosound;
44              setintvec(28,salv)
45          end
46      end;
47
48  end.
49
```

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 acţiuni: - prima, funcţiile propriu-zise ale programului; - a doua, derularea unei melodii în paralel cu execuţia programului (ceea ce există la majoritatea programelor de joc).

În tabloul VALNOTE (programul MUZICA.PAS) sînt date frecvenţele 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 frecvenţe, al doilea este durata notei în unităţi de 55 ms.

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

mat. Ioan Cozac

Ştergere rapidă

Deseori, se doreşte ştergerea completă a unei dischete, fără însă a o reformata. Pentru aceasta, trebuie şterse toate fişierele din toate subdirectoarele - ba probabil printre ele mai sînt şi fişiere protejate la scriere, care trebuiesc întîi "deprotejate" pentru a putea fi şterse. Cu programul QFORMAT, o dischetă deja formatată poate fi rapid "reformatată". Programul şterge pur şi simplu cele două tabele FAT precum şi directorul rădăcină, după care suprascrie toate sectoarele dischetei. După ce s-a operat o astfel de ştergere, tot ce a fost pe dischetă este definitiv şi iremediabil dispărut. La startare, programul întreabă în ce unitate se va lucra (A sau B). Alte unităţi sînt blocate; o ştergere accidentală a harddisk-ului este exclusă. Funcţional, 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/$46/$ff);
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 operaţie care consumă timp şi efort din partea utilizatorului. Programul CHDIR.PAS uşurează această operaţie. 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 conţin în nume şirul specificat ca parametru. Dacă se găseşte un astfel de subdirector, numele complet al căii respective este afişat pe ecran. Dacă acesta este subdirectorul căutat, se apasă <RETURN> şi schimbarea de cale se efectuează. La acţionarea 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

Poşta redacţiei

... noi avem nevoie de foarte multă literatură de specialitate. Citind o carte de teoria proiectării sistemelor informaţionale (apărută la o editură românească, bineînţeles) am ales o listă de denumiri de cărţi de care aş avea nevoie în primul rînd. Lista o anexez, subliniind poate cele mai necesare cărţi.

Ovidiu Ciubotă, str. Florilor 6/2, apt. 35, 277068 Chişinău, Republica Moldova, URSS

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

Ne-ar părea tare rău ca această scrisoare să treacă prin conştiinţa noastră ca un simplu fapt divers. Este - şi trebuie să fie - mult mai mult. Îi rugăm pe toţi cititorii noştri să recepţioneze acest apel - chiar dacă scrisoarea a venit pe adresa noastră. Şi nu numai de la cititorul nostru (care se întîmplă să fie primul nostru corespondent de dincolo de Prut). Ci de la toţi cei ca el. Faptul că speranţele 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 aşteptăm - sîntem convinşi că vor fi bine utilizate toate cărţile ce vor fi trimise. Direct la Chişinău (preferabil!), sau întîi nouă.. Cei care consideră că pot trimite cărţi de specialitate, sînt rugaţi să o facă. Mulţi dintre noi au cărţi ce pot desigur prezenta interes - şi de care ne putem dispensa.. Sigur că unele sînt învechite şi nu mai au valoare - dar există şi multe cărţi care îşi păstrează valoarea în timp. Nu cred că există multe biblioteci de întreprindere care să nu aibă cărţi pe care s-au aşternut multe degete de praf. Şi care ar putea prezenta o reală valoare pentru colegii noştri de dincolo de Prut, prin simplul fapt că tratează nişte probleme de specialitate în limba română. Să fie chiar imposibil ...? Puţin sceptici - sincer, nu credem că va fi extrem de multă muncă de prestat - ne oferim şi noi ca 'mediatori'. Cărţile ce se vor aduna ca donaţii la noi promitem să le ducem într-un fel sau altul la cei cărora le sînt destinate. Şi nu vedem nici un motiv ca printre cei care fac donaţii să nu se afle şi edituri, instituţii, întreprinderi.

Apropo de instituţii - o trecere în revistă a fişierului nostru de abonaţi relevă un fapt puţin surprinzător: marea majoritate a celor care au făcut abonamente la revista noastră sînt persoane particulare. Cîteva firme particulare, cîteva (puţine) Centre de calcul, cîteva instituţii de stat - şi atît! Dragi cititori,, ştim că uneori este incomod, şi că uneori biblioteca întreprinderii la care lucraţi funcţionează prost sau nu funcţionează de loc. Dar mai ştim şi că sînt foarte puţini directori sau contabili şefi care să citească revista noastră. Neavînd solicitări exprese în acest sens de la subalterni, nici unul dintre aceştia nu va face abonamente la ilustra necunoscută revistă If! Iar bibliotecara - dacă există - este, probabil, cel puţin încurcată dacă trebuie să decidă dacă merită sau nu să facă abonament. Sigur că revista e scumpă - sîntem conştienţi de acest lucru, singura alternativă ar fi să renunţăm. Şi nu vrem să ne dăm bătuţi. O variantă posibilă ar fi creştera tirajului - dar pentru aceasta, difuzarea ar trebui să se întîmple cu totul altfel decît se întîmplă de fapt. Nu putem ieşi 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 preţul ridicat al hîrtiei şi tipăririi, nu ne permit pur şi simplu să ieşim într-un tiraj mai mare. Abonamentele sînt o certitudine - cu ele în spate, tirajul poate creşte. Şi dacă abonamentul este scump - poate că pentru întreprinderea sau biroul în care lucraţi nu este totuşi un efort semnificativ!

Nu vrem acte de caritate - dacă nu merită să supravieţuim, foarte bine, vom dispare, ca atîţia alţii. Poate am calculat greşit, poate că ne-am întins mai mult decît ne este plapuma, poate că pur şi simplu nu ne pricepem.

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

Al 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 Bucureşti
- Gliga Jenica, Cluj
- Nemeş Corina
- Miron Cornelia, Arad
- Sibiu, str. Bîrsei nr. 10, bl.39, ap.17

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

Mica publicitate

Vînd HC-85, şah-computer Mephisto, joystick-PC, unităţi 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 şi documentaţii pt. calculatoare Spectrum. Szentes Attila, str. Pescarilor, bl. 6, sc. A, apt. 9, 4400 Bistriţa.

Cumpăr microdrives cartridges (cartuşe cu bandă) pentru calculatorul Sinclair QL sau Spectrum şi software pentru QL. Tel. 90/754782.

În numărul viitor:

- **10 programe antivirus**
- **DTP şi fonturi**
- **DOS extendere**

Important !

Doriţi să aveţi garanţia că nu veţi pierde nici un număr din "if" ? Încheiaţi un abonament folosindu-vă de talonul alăturat, sau adresaţi-vă difuzorilor noştri autorizaţi:

Baia Mare
Omni Computer
(Cudalbu Octavian)
tel. 36433

Brăila
Teodorescu Constantin
tel. 47203

Bucureşti
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

Oneşti
Brumă Cornel
tel. 24222-236

Oradea
AbMod
tel. 60278, 44476

Rădăuţi
Burciu Florin
tel. 63483

Sighişoara
Beer Mihai
tel. 73638

Timişoara
Tomoroga Mircea
tel. 77422

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

Talon de comandă

Doresc să-mi trimiteţi |______| exemplare ale revistei **If**, pe o perioadă de |_____| apariţii, pe adresa:
Nume&prenume ....
Strada: .... Nr.: ..... Bloc: .... Ap.: .... Tel.: ....
Cod: .... Localitatea: .... Judeţ (sector) ....

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

If 6 / 91 Data: .... Semnătura. ....

Talon pentru mica publicitate

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

```
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|_|
```

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

Data. .... Semnătura .... **If 6 / 91**

Important !

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

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

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

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

Lista faxurilor autorizate de Ministerul Comunicaţiilor pentru a fi conectate în reţeaua naţională de telecomunicaţii

Listă aprobată la 5 iunie 1991

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

software

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

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

1. Bal-Con - evidenţa contabilă generală

2. Mi-Fi - evidenţa şi calculul amortizării mijloacelor fixe

3. Ge-Ma - gestiunea şi evidenţa contabilă a valorilor materiale (materiale, obiecte de inventar, produse finite).

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

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

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

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

Permite culegerea, prelucrarea şi vizualizarea datelor în regim multiuser, de la 1 la 8 terminale, conectate pe căile seriale COM1 -COMB şi 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 funcţionare (VDT-52S, etc.), prin utilizarea acestui produs achiziţionarea unor calculatoare IBM-PC compatibile în configuraţia obişnuită (2 căi seriale) nu exclude utilizarea eficentă a acestor terminale devenite disponibile.