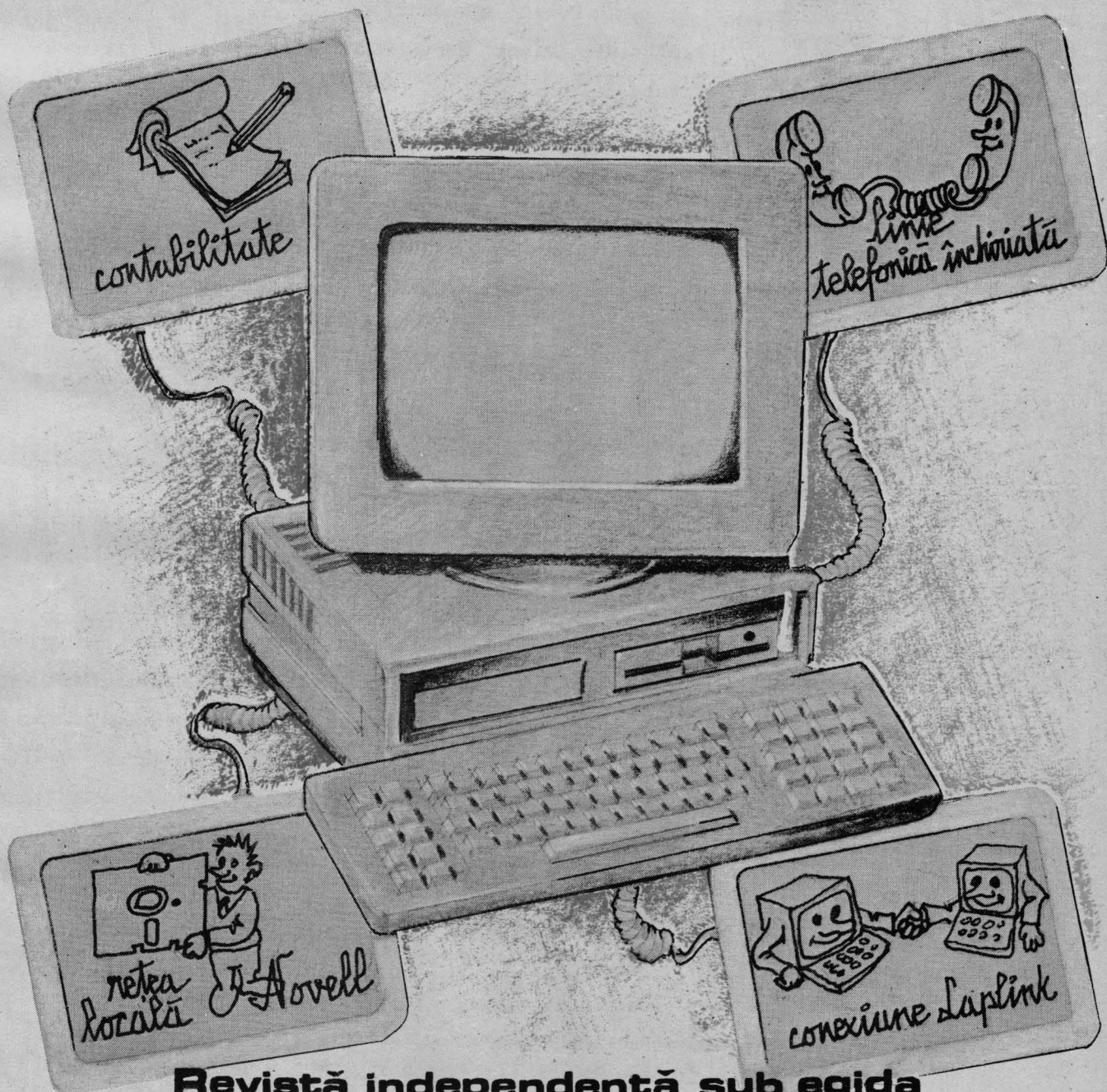


PC Magazin

Anul I Nr.1

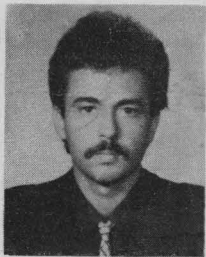
Martie 1990



**Revistă independentă sub egida
Asociației specialiștilor în baze și bănci de date
BUCUREȘTI 1990**

Colectivul de redacție al revistei se prezintă

Redactor șef și fondator al revistei: Adrian NEGRU



Profesia: matematician. Absolvent al Facultății de Matematică din cadrul Universității București

Preferințe: inteligența artificială, liniște sufletească

Vîrsta: 30 ani.

Redactor șef adjunct și fondator al revistei: Alexandru BABIN

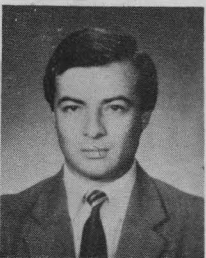


Profesia: inginer automatist. Absolvent al Institutului Politehnic București

Preferințe: micro sisteme PC, onestitate umană

Vîrsta: 34 ani.

Redactor responsabil de număr: Vasile AVRAM



Profesia: economist și cadru didactic asociat la Catedra de Cibernetică Economică din A.S.E. Absolvent al Facultății de Cibernetică Economică din A.S.E. București

Preferințe: baze de date, vînătoare sportivă

Vîrsta: 38 ani.

Redactori: Cătălin VOLOȘENIUC, Ionel RUSE

La elaborarea acestui număr au participat: Mihai Constantin; Marcel Vlădescu; Irina Negru; Irina Babin; Mihaela Olteanu; Gheorghe Sabău; Cristian Florin Popescu; Mihai Unghianu; Aurelian Cojocar; Aurelian Rusu; Viorel Dumitrache; Bodosi Imre; Dragoș Rișcanu; Costel Dinescu; Silviu Bârză; Vasile Vicoleanu; Marius Sturzoiu; Adrian Goicea; Tiberiu Spircu; Magda Negoită; Elena Popa; Cristian Groza; Mihai Trandafirescu; Virgil Sultanovici; Garbis Ohanesian.

Dedicația primului număr:

"Mamelor noastre Ioana, Silvia, Constanța care prin prinosul lor de durere și renunțare ne-au dăruit un loc sub soare."

ARGUMENT

De ce o revistă de informatică? Fără îndoială foarte mulți o așteptau de mult, simțeau nevoia unui suport informativ și orientativ, alții își pun problema utilității și finalității imediat argumentative a ei.

Fără îndoială că, printre speranțe, scepticism și entuziasm, sarcina editării acestei publicații este o întreprindere grea, delicată și mai ales un act de curaj; însă, într-un moment în care informatica se instaurează ca eminentă cenușie a oricărei economii moderne, al oricărui proces educațional și social, este timpul să scoatem la lumină o știință ținută fără logică în frâu, inhibată și dezgolită de suportul ei aplicativ, software și hardware, și care a supraviețuit printr-un efort absolut eroic al adepților conștienți ai beneficiului ei, ingineri, matematicieni, economiști, fizicieni, care fiecare în parte, după putință și căință și-au adus prinosul de cunoștințe teoretice și practice în ale existenței acesteia. Fără îndoială, avem un sumum de utilaj tehnologic, unele calculatoare indigene, altele importate prin eforturi valutare incredibile, licențe de tehnologie sau altele executate artizanal ori provenite pe diverse căi. În consens însă, marea majoritate a fluxului vital ce animă aceste unități metalice fără viață, software-ul, sau programele aplicative ce pun calculatoarele în lucru, au fost în mare parte interzise importului, cele existente fiind apanajul unei munci extraordinare a întregului potențial intelectual informatic al acestei țări. Putem afirma cu o îndreptățită mândrie că ceea ce avem efectiv valoros în momentul de față în informatică este bagajul extraordinar de cunoștințe teoretice a celor ce s-au dedicat acestei științe, o mare avere obținută de mulți dintre noi, fiecare în felul lui, izolat, sau în grup, acasă sau la locul de muncă, prin inimaginabil de multe moduri de a găsi din lume noutăți în documentație sau software. Cărți trecute din mână în mână, citite și apoi comentate, pachete software pe care s-a lucrat aproape orice, reviste sau articole citite și adaptate conștient propriilor noastre idei și formații științifice. De asemenea, multe mulțumiri celor ce de-a lungul anilor au putut publica lucrări în țară în folosul informaticii.

Neexistând o platformă informatică consecventă în planurile de învățămînt, dezvoltarea acestei ramuri științifice s-a făcut, funcție de posibilitățile de dotare, de profilul economic al cerințelor aplicative, în centre de calcul ale diverselor ramuri de activitate sau în institute specializate.

Asistăm astfel la un fenomen destul de ciudat și anume discrepanța dintre lăcașurile informatice sub aspectul următor: în centre sau unități cu posibilități de dotare tehnologică prioritară și preferențiată, contactul și practica cu noutățile s-a manifestat imediat prin cunoașterea și lucrul cu limbaje de programare moderne și cu pachete software la zi pe piața mondială, la fel ca și practica pe calculatoare moderne de la micro profesionale PC AT pînă la sisteme VAX/VMS; în timp ce în unități cu disponibilități reduse la investiție materială, dar cu un potențial teoretic ridicat, s-au accentuat studiile de mare complexitate matematică sau tehnologică.

Efortul fabricilor de calculatoare pentru utilizarea internă a centrelor informatice a fost și este extraordinar, ținînd cont de condițiile în care au putut lucra.

Argument

Se impune astfel, acum, corelarea și interclasarea disponibilităților și cunoștințelor din aceste două tipuri de entități informatice. Această unire, impune fără îndoială și coordonare competentă, ce însă, nu trebuie înțeleasă greșit, prin restrângerea independenței de acțiune creatoare și inițiativă a fiecărui nucleu informatic în parte. Nu trebuie repetată greșala centralismului prin crearea unei Mecca a informaticii care să dea verdictul sub formă de tehnologie și programe fiecărui centru de calcul sau unitate de învățămînt, îngrădindu-i astfel întreaga independență creativă. Să ne dorim instituirea unei alianțe științifice a tuturor forțelor informatice, prin programe realizate în comun, un flux ușor de transfer al informațiilor la nivel național și de ce nu, a unei piețe a concurenței soft și hard.

Este adevărat că explozia tehnologică informatică mondială atinge astăzi cote greu de imaginat, însă potențialul nostru teoretic creativ în domeniul software, hardware și matematic este de o valoare oricînd posibil de pus pe picior de egalitate cu orice chintesență de probitate mondială. Trebuie însă colerate aceste potențe, stimulate și încurajate, mai ales că majoritatea sînt tineri.

Apariția acestei reviste trebuie privită ca unul din multele începuturi ale afirmării informaticii în viața noastră și-și propune a se institui ca o publicație generală pentru tineret, fără dorința de monopol publicistic. Salutară ar fi și apariția altor reviste de specialitate pe profil ca, de exemplu, economică informatică, statistică informatică, tehnologie informatică, orientate în probleme concrete ale activităților pe care le reprezintă.

De asemenea, nu va trebui niciodată considerat aceste apariții ca o risipă inutilă de hîrtie sau efort uman, pentru că trebuie să nu uităm de acum nici o clipă că apanajul unei societăți moderne este informația și libera circulație a acesteia, putința de expresie a ideilor și aplicare a lor.

De exemplu, Ungaria cu o populație echivalentă cu jumătate din a țării noastre are circa zece publicații informatice la nivel național.

Mult timp s-a privit programarea ca un simplu proces de codare a informației în limbajul înțeles de un anumit calculator. Această optică este greșită, programarea fiind un act de creație, fiecare program înglobînd în el cunoștințe complexe și efort intelectual, putîndu-se alinia creațiilor ce se bucură de drepturi de autor și de aceea se dorește stimularea ei liberă. Mai mult, trebuie căutată o concepție informatică unitară care să suplinească, cel puțin pentru moment, lipsurile tehnologice prin aplicații scrise clar, cu surse de program perfectabile și supuse dezbaterii publice de specialitate.

Probabil că într-un viitor apropiat se va generaliza introducerea informaticii în programele școlare, astfel ca fiecare elev sau student să fie capabil să-și scrie singur programe ca un ajutor al muncii și educației sale, pentru că mulți au sau vor avea acasă cîte un home-computer iar acest lux va deveni treptat un mijloc de lucru cotidian precum creionul sau planșeta de desen.

Să ne amintim cît de iubită și căutată a fost și este "Gazeta matematică", care de-a lungul celor peste 90 de ani de existență și-a păstrat, indiferent de contorsiunile istorice și sociale, neîntinat prestigiul și probitatea sa ca mesager al conștiinței matematice în rîndul tuturor și cîți dintre noi nu am rezolvat sau rezolvăm și acum probleme cu aceeași plăcere și frenezie tinerească.

Am dori ca și revista noastră de informatică să-și găsească drum spre căutările și interesele tuturor și să putem afla acea formulă care s-o facă de un

Argument

real folos celor ce-o citesc. De aceea, ca structură, ea se împarte în rubrici de sine stătătoare, articole de număr, probleme propuse și rezolvate, noutăți, teme, recenzii, sinteze ca și o poștă a redacției pentru legătura cu publicul larg cititor. Rubricile fixe ale revistei cuprind cursuri de învățare sau perfecționare a limbajelor moderne de programare: C, Pascal, Prolog precum și Basic, pentru că mulți posesori de calculatoare personale au implementat Basic-ul ca limbaj standard.

De asemenea, multe articole sub formă de curs sau de sine stătătoare sînt dedicate cunoașterii în detaliu a microcalculatoarelor personale echipate cu microprocesoare din familia Intel 8086, 80186, 80286, 80386, 80486, cît și a sistemului VAX/VMS atît sub aspect software cît și hardware. Există rubrici destinate atît programatorilor începători cît și avansați. Un spațiu amplu este dedicat unui domeniu de avangardă al informaticii: inteligența artificială, prin introducerea în concepțiile programării logice și a tehnicilor ei specifice.

Fundamente matematice ale arhitecturii generației a V-a de calculatoare ca teoria fuzzy, teoria ergodică, categorii și toposuri, algoritmi de căutare (alfa-beta, mini-max, etc) vor fi abordate pe larg în paginile revistei. O deosebită atenție se va acorda problemelor rezolvate, ce nu sînt altceva decît programe scrise în diverse limbaje de programare, cît și problemelor propuse, ele adresîndu-se ca suport didactic în special elevilor și studenților ce urmează cursuri de programare. Vor urma probleme deschise și propuneri de studii în diverse domenii de informatică ce se adresează tuturor informaticienilor ca și noutăți sau recenzii din această activitate.

Toate expunerile se vor cît mai clare, ele adresîndu-se celor ce fac primii pași în descifrarea tainelor acestei științe cît și specialiștilor mai puțin inițiați cu anumite domenii informatice.

Încecarea noastră este ca paginile revistei să suscite interesul unei mase largi de cititori, făcînd apel ca toți cei ce au sugestii de îmbunătățire a formei sau a conținutului, s-o facă, așteptînd articole, probleme și curiozități de la toți cei ce doresc să-și aducă aportul publicistic și profesional în paginile acestei publicații. De asemenea, poșta redacției se va institui și ca mijloc de comunicare între toți cei interesați în schimburi de idei, software sau documentație.

Așa cum informatica pătrunde astăzi în toată activitatea socială și economică, este posibil ca foarte curînd să asistăm la introducerea ei în procesul general de învățămînt ca obiect de studiu, alături de matematică și fizică, precum și a creșterii ponderii ei în sfera specializării anumitor domenii care să asigure larga lor deschidere și compatibilitate la nivel mondial.

Cu speranța ca în acest an al începutului libertății noastre revista să-și aducă prinosul de bine întru împlinirea noastră, ne punem cu acest număr de avangardă în consensul general de renaștere națională.

Adrian NEGRU /

Fenomenul "Home-Computer" Calculatoare personale profesionale de la proiectare la realizare (I).

Elemente de arhitectură a unui microsystem profesional

1. Particularități constructive ale unui microcalculator

Înțelegem modul de funcționare a familiei de microcalculatoare personale, profesionale dacă putem investiga arhitectura sa intimă și mai ales dacă cunoaștem microprocesorul ce acționează ca un adevărat creier al computerului. A fost aleasă familia microprocesoarelor 8086, care introduce conceptul de calcul pe 16 biți și mai nou pe 32 biți (80386), ceea ce înseamnă că acest calculator poate lucra cu 16 biți sau 32 biți în același timp.

Cele cinci părți cheie ale unui microcalculator sînt: **procesorul, memoria, sistemul de intrări/ieșiri (I/O), suportul magnetic disc și programele.** O scurtă imagine introspectivă asupra funcționării acestor componente ne fixează ideile de ansamblu asupra lor pe care le vom trata pe parcursul acestui capitol.

Astfel, procesorul este creierul calculatorului, dispozitivul capabil să ducă la îndeplinire instrucțiunile programelor sistemului. Dacă el este cel care este capabil să efectueze calcule matematice sau operații logice. Cînd vorbim despre un calculator mare numim procesorul său **unitate centrală (C.P.U.)**; în cazul unui microcalculator acesta este cunoscut ca microprocesor.

Memoria este cîmpul de lucru al sistemului. În ea au loc toate activitățile care se petrec la un moment dat în calculator. Eficiența unui sistem este dictată atît de calitatea procesorului cît și de cantitatea de memorie internă pusă la dispoziție. Majoritatea microcalculatoarelor din familia PC dispun de o memorie internă de 640 KO, modelele constructive mai noi măririndu-și această capacitate pînă la 1 MO sau 16 MO. Sistemul de intrări/ieșiri este modul în care calculatorul primește sau exportă date. De fiecare dată, cînd introducem sau exportăm aceste date, calculatorul apelează la serviciile unităților periferice, dintre care: tastatura, imprimanta, linia de comunicație asincronă, discurile.

Suportul magnetic disc este locul unde calculatorul își păstrează datele atunci cînd ele se află în memoria calculatorului.

Există însă și alte suporturi magnetice (banda, caseta) dar discul este cel mai eficient și important. În ultimii ani firmele constructoare de suporturi magnetice disc s-au preocupat tot mai mult de găsirea unor soluții tehnologice care să mărească capacitatea de stocare a informației pe disc. Astfel, s-a ajuns la discuri rigide din familia **WINCHESTER** cu capacitatea între 20 MO-400 MO; iar mai nou folosind efectul Bernoulli în designul capetelor de disc s-a ajuns la o cantitate între 1 GO-6 GO de memorie pentru discuri convenționale; în timp ce noile discuri optice exploatate atît în scriere cît și în citire pot avea pînă la 16 GO.

Programele sînt cele care fac calculatorul să meargă, cele care îl transformă dintr-o cutie de metal într-o unealtă plină de viață. Instrucțiunile care compun programele dictează pas cu pas activitatea unui sistem. Cele două tipuri de programe existente la un moment dat în calculator sînt programele de

aplicații și programele de sistem.

O parte din programele de sistem de care microcalculatorul are nevoie în permanență sînt păstrate permanent în memoria calculatorului, definind așa numitele programe ROM (Read-Only-Memory). Aceste tipuri de programe supervizează și controlează munca calculatorului punînd la dispoziție servicii utilizabile de către orice program aplicativ. Întreg modulul sursă al acestor programe sistem este denumit Bios (Basic Input/Output Service). Alte programe de sistem avînd la bază serviciile Bios pot să nu fie rezidente în memorie, ca de exemplu sistemul de operare DOS (Disc Operating System).

2. Structura datelor

În această secțiune o să trecem în revistă formatul fundamental al datelor pe care le folosește un microcalculator. Unitatea fundamentală de stocaj a datelor în calculator este bitul. În majoritatea microcalculatoarelor 8 biți sînt combinați pentru a forma un octet, fiecare bit în octet putînd lua 2 valori, 0 sau 1, permițînd astfel 256 combinații a celor 8 biți. Astfel, cele 256 simboluri diferite ce pot fi exprimate într-un octet pot reprezenta valoarea unui caracter al setului ASCII sau o valoare întregă între 0 și 256. De obicei, sîntem obișnuiți să scriem aceste numere în formă zecimală, dar ele pot fi scrise în binar sau hexazecimal fără a schimba valoarea din interiorul octetului. Este echivalentă formularea: un număr variază de la 0 la 256, sau un număr variază de la 00 la FF în hexazecimal, sau de la 00000000 la 11111111 în binar.

Deci, cînd conținutul unui octet este exprimat în binar avem nevoie de 8 cifre (0 sau 1), fiecare cifră corespunzînd unui anumit bit, biții fiind numerotați de la 0 la 7. Modul de evaluare al acestor biți este în baza 2, astfel bitul 0 are valoarea 1, bitul 1 are valoarea 2, bitul 2 are valoarea 4 etc, bitul 7 are valoarea 128 (2^7) (bitul 0 se numește bit inferior iar bitul 7 - bit superior).

De ce este nevoie de această reprezentare binară? O motivație rezidă în faptul că sistemul păstrează informații în octeți de stare în memorie sau regiștri de stare pe cipurile plăcilor sistemului. Anumite informații se pot găsi în secvențe particulare de biți în cadrul unui octet, astfel, într-un octet de stare al perifericelor biții 6 și 7 dau informații asupra numărului logic al imprimantei în timp ce biții 0 și 1 specifică numărul logic al unității de disc.

Un grup de biți luați împreună este referit ca un cîmp. Dacă pentru numerele binare fiecare bit are o valoare de 2 ori mai mare decît precedentul său, pentru un număr hexazecimal fiecare cifră a sa este de 16 ori mai mare decît precedentă. Utilitatea numerelor hexazecimale stă în faptul că o cifră hexa descrie conținutul unei jumătăți de octet; astfel, numărul hexazecimal F6 are echivalentul binar 1111 pentru cifra F și 0110 pentru cifra 6. Valoarea F6 pe octet fiind 11110110 în binar. Echivalentul simbolurilor hexa în zecimal este: A(10), B(11), C(12), D(13), E(14), F(15). Caracterele sau datele text - litere ale alfabetului sau simboluri sînt date de importanță deosebită într-un calculator. Multe pachete software integrate ca Lotus, Word Star, Multimate, Tex sînt orientate pentru lucrul pe șiruri de caractere. Fiecare caracter ocupă cei 8 biți ai unui octet ele fiind standardizate de codul ASCII împărțit în două părți: codul standard reprezentînd caractere cu valoarea între 0 și 127 și codul extins cu caractere în gama de valori 128-255. Nu toate caracterele ASCII sînt tipăribile, o parte a lor avînd exclusiv funcții de control și anume cele care poartă codul între 0 și 31.

Elemente de arhitectură a unui microsistem profesional

O tabelă a acestor caractere este prezentată în capitolul dedicat tastaturii, caracterele în codul 1-4 fiind folosite în probleme de comunicație, cele din gama 28-31 fiind marcatori ai limitelor de fișier în timp ce codurile 17-20 se ocupă cu controlul imprimantelor.

3. Dispozitive hardware ale unui microcalculator

Din punct de vedere fizic orice microcalculator se compune din trei părți. Prima parte este numită "Unitate centrală de sistem" în ea fiind încorporate principalele unități funcționale ale calculatorului. La dispoziția utilizatorului este pusă o unitate fizică numită tastatură, în timp ce dialogul între sistem și programator se face pe ecranul grafic al unui monitor. Alimentarea sistemului se face convertind curentul alternativ în curent continuu, 4 voltaje diferite 12 volți; - 12 volți; + 5 volți; - 5 volți. Modelelor mai vechi de microcalculatoare li se atribuia o putere de 65 W ulterior aceasta ajungând la 130 W pentru modelele XT și 200 W pentru modelele AT. Placa de sistem a unui microcalculator cuprinde un circuit imprimat pe care sînt plantate majoritatea cipurilor electronice care pune în mișcare calculatorul. Astfel, aici apar microprocesorul, ceasul de sistem care acționează ca un metronom de coordonare a activității, cipurile de memorie și procesoare specializate în calcule matematice, servicii video etc.

O altă parte a UC o formează dispozitivul pentru discuri care sînt de fapt singurele sisteme mecanice ale Unității de sistem. Urmează apoi conectori opționali (interfețe) și conectorii de bus, ceea ce conferă deschiderea calculatorului spre "lumea" exterioară. Toate cipurile siliconice ca și celelalte părți ale calculatorului sînt conectate între ele astfel încît își pot transmite reciproc semnale și date. Constructiv, aceasta se realizează printr-un canal de comunicație comun alcătuit dintr-un număr de fire prin care datele trec simultan numit și bus. Busul unui microcalculator are 62 de linii separate, identificate prin codurile A1-A31 și B1-B31, conectorii A fiind situați în partea dreaptă, iar B în partea stîngă. Cinci linii sînt utilizate pentru cele 4 voltaje diferite, alte 3 folosind ca linii de masă. Opt linii sînt folosite pentru circulația biților de date ale unui octet, astfel că busul poate transfera cîte un octet de date odată. Douăzeci de linii sînt folosite pentru adrese indicînd cu ce parte a sistemului se lucrează la un moment dat; adresarea este folosită în două moduri și anume: o adresă de memorie și una pentru porturile de intrări/ieșiri. Liniile care au rămas sînt utilizate pentru diverse secvențe de control. O linie indică dacă busul este liber sau ocupat, altele se ocupă cu serviciile de întreruperi ce semnalează care parte a calculatorului cere control. Aceste linii sînt folosite, de exemplu, pentru accesarea porturilor de comunicație serială COM1 și COM2. Modele constructive mai noi din gama AT mai adaugă busului încă 36 de linii numerotate C1-C18 și D1-D18, 8 dintre aceste linii sînt afectate datelor, 8 linii de adresă și 5 serviciilor de întrerupere.

Adaptoarele opționale ca cipul video și cipul de tastatură sînt discutate în capitolele afectate lor.

4. Microprocesorul

Microcalculatoarele echipate cu un 286 sînt, în general, asemănătoare cu un PC normal în sensul compatibilității cu procesoarele 8088 și 8086; mai concret, 286 lucrează cu un bus de memorie de 16 biți ceea ce-l apropie de 8086 și nu de 8088 care este echipat cu un bus de 8 biți dar cu care se aseamănă în modul de încărcare și prelucrare a programelor.

Elemente de arhitectură a unui microsistem profesional

Ceea ce, însă, îl departajează net pe 286 de celelalte microprocesoare (mp) din familie este viteza de execuție a programelor.

Să stabilim ce înseamnă exact, în vorbirea curentă, o frază ca aceea "acest microcalculator folosește un ceas de 8 Mhz". Toate acțiunile unui mp sînt gestionate de un ceas ce acționînd ca un metronom corelează și coordonează contactul mp cu mediul extern, în sensul măsurării temporale exacte a fiecărei operații executate. Măsura unui ceas de sistem o înțelegem prin aceea că un ceas de 8 Mhz "ticăie" de 8 milioane de ori pe secundă. Astfel, cu cît frecvența unui ceas este mai mare cu atît mai rapid este un mp în execuție. Pe un 286 o operație de înmulțire durează circa 20 de cicli în timp ce pe 8088 aceeași operație durează 120 cicli, un ciclu fiind echivalent cu durata dintre 2 semnale ("ticuri") consecutive ale ceasului. Un PC standard utilizează un ceas de 4,77 Mhz, un PC 286 un ceas de 6 Mhz sau 8 Mhz (în cazul PC-ului Compaq Deskpro-286); ultimul fiind cu 67% mai rapid decît cel standard. Și totul nu se oprește aici.. PC-urile echipate cu mp 386 au viteza ceasului în gama 10-12,6 Mhz fiind de 4-10 ori mai rapide decît un PC original.

Așa cum spuneam, alte performanțe ale familiei de PC-uri echipate cu mp 286, 386 apar cînd comutăm în modul protejat. Protecția permite sistemului de operare (MS-DOS, XENIX, OS/2) a ridica bariere în jurul unui program în execuție pentru a-l proteja de accesul altor programe la el sau chiar al sistemului de operare, nepermițînd alterarea zonei de memorie aferentă lui.

Protecția apare ca o noutate și ca o cheie de siguranță în arhitectura mp eliminînd situațiile în care un program cu erori putea bloca sistemul sau putea distruge zone de memorie ce nu-i aparțin.

Problema universală a utilizatorilor de PC-uri, de orice fel, este legată de memorie, de dimensiunea ei, de mărimea spațiului de adresare și de suporturile magnetice, ca unități permanente de stocare și memorare.

Modul protejat prin posibilitatea de gestionare a unei memorii extinse și prin mecanismul de memorie virtuală permit instalarea în sistem de pînă la 16 megabytes de memorie și a unui spațiu de adresă accesibil unui program de pînă la 1 gigabyte (echivalentul a peste 1 billion de octeți). Pînă acum, PC-standard foloseau o memorie fizică de la 16 k la 640 k, cea mai satisfăcătoare fiind cea de 640 k unde utilizatorul își poate instala un disc virtual de mărime dorită. Avantajul unei astfel de memorie virtuală constă în aceea că ea se prezintă ca un spațiu continuu (sau contiguu) de octeți accesibili direct de un program, adresele fiecărui octet fiind adrese gestionate de program, adică adrese logice și nicidecum adrese fizice, fixe în spațiul real de adresare.

Instalarea unei zone de memorie virtuală se face prin intermediul sistemului MS-DOS versiunea 3.XX cu care este echipată marea majoritate a PC-urilor cu ajutorul programelor VDISC.EXE (pentru versiunea 3.10) și RAMDRIVE.EXE (3.2X..) ce nu sînt altceva decît niște programe ce simulează ca suport disc o zonă de memorie, de unde și denumirea de disc virtual dată procedurii de vizualizare.

Zona se recunoaște printr-un nume dat ei; de obicei D:, E:, numele simbolice A:, B: fiind rezervate pentru unitățile fizice de floppy-discuri, iar C: unității de hard-disk sau disk rigid. Dimensionarea și sectorizarea discului se face soft. De exemplu:

```
RAMDRIVE D: 3208512
```

instalează în memoria standard un disc virtual D:, cu capacitatea de 320 k avînd 8 sectoare pe pistă a cîte 512 octeți de sector.

Elemente de arhitectură a unui microsystem profesional

Să prezentăm în continuare, pe scurt, trăsăturile fundamentale ale componentelor dintr-un microprocesor.

Microprocesorul INTEL 80286 are o adresă de 24 biți, o interfață de memorie de 16 biți, un set de instrucțiuni extinse, DMA, suport hardware de înmulțire și împărțire în virgulă fixă, un manager de memorie integrat, 4 nivele de protecție a memoriei, un gigabyte (1.073.741.824 octeți) de spațiu virtual de adresă pentru fiecare task și 2 moduri de operare:

- modul real compatibil cu 8086;
- modul protejat, compatibil cu microprocesoarele 80286, 80386.

În modul real, memoria fizică este un vector continuu pînă la un megabyte, adresa fizică făcîndu-se de pînă la 20 biți. Primii 16 biți de adresă formează adresa selectorului sau a segmentului de memorie adresat, ultimii 4 biți ai adresei de segment pe 20 biți fiind întotdeauna zero, de aceea adresa segmentului de memorie începe întotdeauna la o adresă multiplu de 16.

În modul de adresare real toate segmentele de memorie au o mărime de 64 Kb și pot fi accesate în citire sau scriere. O excepție (derivă) sau o întrerupere poate avea loc dacă datele sau instrucțiunile unui program depășesc zona limită de 64 Kbyte.

Dacă informația conținută într-un segment de memorie nu folosește toți cei 64 Kbyte ai unui segment partea nefolosită poate fi acoperită de un alt segment pentru economisirea memoriei fizice.

În modul protejat adresa este un pointer de 32 biți format dintr-un selector pe 16 biți și componenta de deplasare. Selectorul, în acest caz, desemnează un index într-o tabelă de memorie rezidentă și nu primii 16 biți ai unei adrese de memorie reală. Adresă de bază pe 24 biți a segmentului dorit este obținută din tabele în memorie.

Deplasamentul de 16 biți este adăugat adresei de bază a segmentului pentru a forma adresa fizică. Ca observații: tabelele de memorie conțin valori pe 8 octeți numiți descriptori.

Să enumerăm câteva din performanțele sistemului. Un mp 80286 poate lucra la 6 Mhz, 8 Mhz și 10 Mhz. Cel ce lucrează, de exemplu, la 6 Mhz are un ciclu de ceas de 167 nanosecunde.

Un ciclu de bus pe 16 biți are nevoie de 3 cicli de ceas și după fiecare dată un semnal de wait ceea ce însumează 500 u.s., evident un timp de 2 ori mai mic decît un ciclu de bus pe 8 biți care are nevoie de 6 cicli de ceas și de 4 stări de wait cca 1000 u.s.; iar un ciclu de bus pe 16 biți are nevoie de 12 cicli de ceas + 10 stări de wait de I/O echivalentul a 2000 u.s. Controlerul de DMA lucrează la 3 Mhz, echivalentul unui ciclu de ceas de 333 u.s., toate transferurile de date DMA pe bus reclamînd 5 cicluri de ceas sau 1,66 microsecunde.

În general canalele DMA 0, 1, 2, 3 sînt folosite pentru transferare de date pe 8 biți, iar canalele 5, 6, 7 procesează transferul de biți. Canalul 4 este folosit pentru accesul canalelor 0 și 3 la microprocesor.

Microprocesorul calculatorului are trei moduri de comunicare cu lumea circuitelor din afara sa. O comunicare specială o are cu procesorul matematic 8087 prin secvența ESCAPE, celelalte două moduri fiind comunicarea cu memoria și folosirea porturilor. Deci memoria este locul unde microprocesorul își găsește instrucțiunile unui program la fel ca și datele. Atît datele cît și instrucțiunile sînt stocate în memorie de unde microprocesorul le ia și le prelucerează intern. Deschiderea acestuia spre exterior se face la nivelul porturilor, acestea

Elemente de arhitectură a unui microsistem profesional

acționând ca o linie telefonică prin care microprocesorul poate comunica. Fiecărei componente a calculatorului cu care microprocesorul poate comunica i se atribuie un număr de port pe care acesta îl folosește ca pe un număr de telefon pentru a accesa partea de care are nevoie.

La dispoziția microprocesorului sînt puse 65536 numere de port utilizabile din care numai o parte sînt afectate, celelalte rămînînd disponibile pentru dezvoltări opționale ulterioare ale calculatorului. Accesarea porturilor de către microprocesor se face prin două comenzi existente în limbajele de asamblare cît și în limbajele de nivel înalt: comanda OUT care trimite date la un port și comanda IN care citește date la un port. Uneltele necesare microprocesoarelor în rezolvarea problemelor sale sînt regiștrii și stivele. Regiștrii sînt o zonă de memorie destinată stocării datelor, cu care lucrează la un moment dat microprocesorul și se constituie ca parte internă a acestuia cu funcții speciale. Primul grup de 4 astfel de regiștrii, numite de uz general, utilizate în special în calcule, sînt organizate ca doi octeți (16 biți) și cunoscute sub numele AX, BX, CX, DX.

Pentru microprocesoarele Intel 80386 și 80486 aceste registre sînt organizate pe 32 biți (4 octeți) și denumite EAX, EBX, ECX, EDX. Fiecare dintre ele este utilizat de către program ca zonă temporară de stocaj, utilizatorul putînd accesa și jumătăți ale lor (cîte un octet) denumite, în cazul lui AX de exemplu, AH (octetul superior, biții 8-15) și AL (octetul inferior, biții 0-7). La fel, BX se împarte în BH și BL, CX în CH și CL, DX în DH și DL. În cazul microprocesoarelor 80386 și 80486 EAX și EAX au cîte 16 biți. Următorul grup de regiștri ajută microprocesorul în adresarea exactă a zonei de memorie cu care se lucrează. Aceștia sînt cunoscuți ca regiștrii de segment fiecare permițînd accesul la un segment de memorie ce măsoară 64 k octeți de memorie. Registrul de cod al segmentului, notat CS, indică unde este localizat în memorie un program. Registrul de date, notat DS, localizează zona de date din memorie pe care o folosește programul, în timp ce registrul SI suplimentează segmentul de date. Registrul de stivă, notat SS, localizează segmentul stivă din memorie. Ultimul grup de 5 regiștri permit localizarea în memorie a anumitor octeți de date dorite, utilizarea lor împreună cu regiștrii de segment permițînd localizarea exactă a datelor din memorie.

Pointerul de instrucțiune sau registrul IP acționează ca un contor al instrucțiunilor unui program informînd microprocesorul despre instrucțiunea ce se execută la un moment dat.

Pointerul de stivă, notat SP, și pointerul de bază, notat BP, sînt regiștrii care țin evidența datelor care sînt stocate în stivă. Indexul de sursă, notat SI, și index destinație, notat DI, sînt ultimii doi regiștrii ce permit programelor manipularea unor cantități mari de date dintr-un loc în altul. Un ultim registru, numit de semafoare, informează programul prin configurația biților din el despre starea calculatorului la un moment dat (rezultatul unei operații aritmetice, semnalul depășirii aritmetice, permisul de activare a unei întreruperi, paritate etc.).

Stiva servește ca zonă de păstrare a informațiilor despre activitatea calculatorului. Deci apelul unei subrutine este semnalizat în stivă de informații care spun unde a fost întrerupt programul în lucru pentru a fi reluat după execuția rutinei apelate.

5. Întreruperi

Întreruperea este capacitatea sistemului de a suspenda temporar orice activitate ce este în curs, redirectarea atenției sale către semnalul generator al întreruperii, rezolvarea cauzelor generatoare acestui semnal și reîntoarcerea la activitatea mai devreme abandonată. Întreruperile nu sînt altceva decît niște rutine deja scrise în sistem la care calculatorul face apel pentru a rezolva o anumită problemă. Acestea sînt de 2 categorii: întreruperi hardware și întreruperi software.

Cele hardware sînt inițiate de componentele fizice ale sistemului propriu de pe placa de sistem sau de la canalul de intrări/ieșiri. Ele sînt cunoscute ca întreruperi bios (apăsarea unei chei de la tastatură, semnal de la imprimantă), ele nefiind corelate cu activitățile software. La semnalarea unei întreruperi valoarea adresei CS:IP este salvată în stivă la fel ca și registrul de semafoare; după care adresa din memorie a rutinei de întrerupere este încărcată în CS:IP și i se predă controlul. Rutinele întreruperilor mai sînt cunoscute și ca HANDLERE de întreruperi, ele terminîndu-se întotdeauna cu instrucțiunea IRET ce anulează procesul care a startat rutina restaurînd din stivă valorile originale pentru CS:IP care predă, astfel, controlul programului întrerupt anterior.

Cealaltă clasă de întreruperi software nu întrerup fizic nimic. Ele sînt organizate în mod esențial ca proceduri apelabile de un program pentru diverse nevoi. Ele se află scrise în sistemul de operare, accesarea lor făcîndu-se printr-un mecanism specific sistemului hardware care poate prelua controlul unei întreruperi software.

Adresele de întreruperi mai sînt numite și vectori. Fiecare vector are lungimea de 4 octeți. Primii doi octeți păstrează valoarea lui IP iar ceilalți doi a lui CS.

Cei mai de jos 1024 octeți de memorie conțin vectori de întrerupere, zonă referită adesea ca **tabelă de vectori** ce poate acomoda 256 astfel de adrese. Astfel vectorul lui INT0 se află la adresa 0000:0000, INT1 la adresa 0000:0004, INT2 la adresa 0000:0008. De exemplu, dacă vrem să ne uităm la adresa 0000:0020 care păstrează vectorul pentru INT8H (programarea datei și a orei) vom găsi valoarea F000:FEA5 care se constituie ca adresa de start în ROM a rutinei ce execută INT8.

Cipul 8259 este folosit la microcalculatoarele profesionale pentru gestionarea întreruperilor hardware. Deoarece mai multe cereri de întreruperi pot fi apelate simultan, cipul pune la dispoziție un nivel de priorități al acestuia. Există opt nivele de priorități cu excepția mașinilor AT unde sînt 16, apelurile la nivelurile lor fiind abreviate cu IRQ0 - IRQ7 (sau IRQ15). Prioritatea de cel mai înalt nivel este zero cererile 0-7 intrînd în vectorii pentru INT8H - INTFH; pe AT cererile 8-15 sînt servite de INT70H - INT77H.

Deci cipul 8259 are regiștrii de un octet care controlează și dictează cele 8 linii de întrerupere hardware. Regiștrul de servicii al întreruperii (IRK) schimbă un bit pe 1 cînd linia de întrerupere corespunzătoare semnalează o cerere. Imediat cipul semnalizează automat dacă altă întrerupere este în progres consultînd registrul de serviciu (ISK) pentru această informație.

Înainte de invocarea întreruperii, registrul de mască (IMK) este verificat pentru a vedea dacă întreruperea de nivelul cerut este permisă sau nu. Una din facilitățile mari puse la dispoziție de sistem este libertatea dată programatorului de a scrie propria sa întrerupere sau modificarea uneia existente. Funcția 25H a întreruperii 21H permite semnalarea unui vector de întrerupere utilizator la o adresă specifică. Adresa are două cuvinte lungime, valoarea

segmentului pentru CS și deplasamentul pentru IP. Pentru a seta vectorul astfel încât el să se constituie ca un pointer al rutinei utilizator se pune segmentul în care se află rutina în DS și deplasamentul rutinei în DX. Apoi numărul întreruperii se pune în AL după care are loc apelul funcției.

Cele 16 nivele de întreruperi de sistem sînt generate de două controlere de întrerupere 8255 A și de NMI mp. Toate întreruperile pot fi mascate, inclusiv NMI.

Subsistemul ROM este format din 2, 4 sau 8 module de 8 biți ROM/EPROM a cîte 32 k. Codul pentru adresele pare și impare sînt situate în module separate, iar memoria ROM este localizată la sfîrșitul primului și ultimului 1 M de spațiu de adresă, adică la adresa hexa 0F0000 și adresa hexa FF0000. Memoria ROM nu este supusă verificării de paritate, iar timpul de acces la ea este de 150 ns și are un ciclu de 230 ns.

Subsistemul RAM în cei 16 mega în spațiul de adresă de memorie RAM începe la adresa hexa 000000 putînd fi prezentă în module de 256 Kb, 512 Kb, 640 Kb, 1 M sau 4 M. Timpul de acces la memorie este de 150 us și un ciclu durează 275 us. Programul de inițializare a memoriei RAM procesează în felul următor:

- inițializează canalul 1 ca timer cu o perioadă de 15 microsecunde;
- procesează o operație de scriere a memoriei la nivelul oricărei locații de memorie.

6. Memoria

Memoria calculatorului este zona unde se stochează informația care include atît instrucțiunile de program cît și datele avînd ca unitate constructivă de stocaj și de măsură octetul. Orice octet poate lua 256 valori distincte, biții care îl organizează putînd reprezenta un număr, codul unei litere din alfabet sau o instrucțiune particulară de mașină, funcție de modul de interpretare a semnificației. Octeții din memorie se organizează astfel încît să creeze unități de informație specifice, cum ar fi cuvîntul organizat pe doi octeți consecutivi sau un șir de caractere reprezentați ca o succesiune de octeți ce interpretează un text. Utilizatorul are acces la memoria calculatorului printr-o adresă ce este un număr asociat fiecărui octet.

Dacă un șir de caractere este stocat în memorie caracter cu caracter în ordinea în care este scris, numerele au un mod de stocaj diferit în sensul următor: numărul 2650 apare în calculator pe un cuvînt în ordinea 0562. Principiul se aplică și formulelor mai mari de doi octeți și anume pentru cuvinte duble reprezentate pe 32 biți.

Dacă numerele sînt reprezentate în format hexazecimal în memorie, reprezentarea se face nu prin inversarea fiecărei cifre hexa ci prin inversarea octeților; astfel numărul A0F1 se reprezintă în memorie sub forma F1A0. Modul de adresare al memoriei a fost standardizat prin definirea așa numitelor segmente, unități de 64 k ce pot fi accesate prin așa numitele adrese de segment care nu sînt altceva decît cuvinte pe 16 biți. Aceste adrese sînt astfel combinate încît pot accesa 1048576 octeți de memorie în modul de lucru real. Vom înțelege prin spațiul de adresă reală posibilitatea de a accesa în orice moment oricare zonă fizică a memoriei, spre deosebire de modul protejat (procesoarele 80286, 80386, 80486) în care se poate adresa o zonă de memorie virtuală între 16 MO-16 GO. Deci posibilitatea de adresare a unui mega de memorie în mod real se realizează prin crearea unei adrese pe 20 de biți din care una reprezintă adresa de segment, iar cealaltă deplasamentul în cadrul segmentului. Să presupunem că adresa începutului

Elemente de arhitectură a unui microsistem profesional

de segment este ABCD și vrem să aflăm adresa completă a octetului aflat la 1234 de octeți depărtare de începutul relativ al segmentului. Atunci adresa efectivă a acestui octet se formează astfel: se scrie pe 5 cifre hexa adresa segmentului adăugînd cifra 0 în coada acestuia: ABCD devine ABCD0. La acest număr se procesează adunarea aritmetică a deplasamentului 1234 obținîndu-se adresa efectivă:

ABCD0

01234

ACF04

Un alt mod de notare condensat a acestei adrese se face: ABCD:1234 ea constituind de fapt și adresa absolută. Indiferent dacă este vorba despre cod de program sau dacă acestea sînt memorate în cadrul aceluiași segment sau în segmente distincte a căror adresă este păstrată întotdeauna în registrul CS (codul de program), DS sau ES în cazul datelor.

Deplasamentul unei locații de memorie în cadrul segmentului se poate păstra în registrele generale sau în registrele de index. Dacă în primele tipuri de microcalculatoare datorită restricțiilor tehnologice, compilatoarele Basic, Fortran, Pascal impuneau existența datelor și a codului de program (instrucțiuni) în limita strictă a unui segment, 64 k, tipurile constructive mai noi permit organizarea memoriei în așa numitele modele. Modelul "slave" cuprinde datele și codul în 64 k, modelul "medium" păstrează datele și codul în segmente distincte de 64 k, modelul "large" păstrează datele pe un segment și codul poate să fie mai mare decît un segment în timp ce modelul "huge" permite ca atît datele cît și codul să depășească respectiv 64 k. Numai ultimul model permite definirea de șiruri de caractere și vectori mai mari de 64 k. Fiecare segment este aliniat la o adresă multiplu de 16 numită adresă de paragraf.

Două părți esențiale ale memoriei sînt zona ROM și zona RAM. Memoria ROM este o zonă care conține informații predefinite și care nu poate fi scrisă sau schimbată de programele noastre, nici ștearsă la închiderea calculatorului sau scoaterea sa de sub tensiune. Ea este cunoscută și ca zona ROM-BIOS avînd trei părți componente. Prima parte cuprinde o serie de programe de test și inițializare cunoscute sub numele de POST (power - on; self - test). A doua parte conține rutinele serviciilor de intrare/ieșire care execută un control detaliat al părților componente ale calculatorului, ale perifericelor de intrare/ieșire (unitățile de discuri, imprimanta). A treia parte a ROM-BIOS-ului este specifică calculatoarelor personale ale firmei IBM și formează compilatorul ROM-BASIC.

Memoria RAM este așa numita memorie volatilă în care sînt încărcate datele și programele care permit calculatorului să funcționeze și care este ștearsă odată cu închiderea sau scoaterea de sub tensiune a calculatorului. Nu tot RAM-ul este la dispoziția utilizatorului, o parte a sa fiind ocupată de sistemul de operare, servicii I/O, rutine de întrerupere rezidente și blocuri de control de memorie. Ne punem firesc întrebarea de cîte memorii RAM dispun utilizatorul unui microcalculator. Zona BIOS păstrează la adresa 0040:0013 doi octeți ce raportează numărul de k de memorie utilizabilă. Pe mașinile AT există un cip care informează în registrele 15H și 16H asupra cantității de memorie instalată (valoarea 0100H echivalentă cu 256 k; 0200H - 512 k, 0280H - 640 k).

Memoria auxiliară peste 1 MO se află închisă în registrele 30H și 31H în unități de 512 k pînă la 15 MO. Întreruperea BIOS 21H întoarce în AX numărul de KO din sistem pe care îi citește din registrele cipului 8255 amintite mai sus.

Pe microcalculatoarele AT funcția 88H a întreruperii 15H verifică extensia de memorie din afara spațiului real de adresă, adică existența memoriei de peste 1 MO. În AX este întors numărul de blocuri de 1 k existent peste 1 MO. Să vedem în continuare modul de accesare și funcționare a memoriei extinse (de peste un MO). Este formată din trei părți, cipurile de memorie ce se instalează opțional pe placa sistem împreună cu două programe software de gestiune a ei, managerul de memorie extinsă (EMM) și un program de aplicație care utilizează această memorie. La startarea calculatorului EMM este activat și pregătește terenul pentru operațiile cu memoria extinsă. Sarcina sa principală fiind să găsească în spațiul de memorie a calculatorului o zonă nefolosită (64 k) în care el depune informații despre această memorie suplimentară. În continuare EMM împarte acest cadru de 64 k în 16 ferestre fiind gata de a asigura orice program de aplicație care știe să îl folosească schimbul cu date din memorie în interiorul și în afara ferestrelor. Limitarea acestei tehnici stă în aceea că schimbul se poate face numai cu date și nu cu cod de program.

7. Unitatea de disc

Cum memoria RAM a calculatorului este volatilă avem nevoie de un suport pe care să putem păstra permanent atât programele noastre de aplicație cât și programele de sistem care pun calculatorul în funcțiune. Aceste unități magnetice ce funcționează ca unități permanente de memorie sînt unitățile de disc. Două probleme se impun a fi cunoscute și anume tehnologia de înregistrare și metoda de acces rapid la disc. Tehnologia are la bază înregistrarea magnetică. Aceeași metodă de înregistrare a benzilor magnetice sau a casetelor video.

Metoda de înregistrare magnetică digitală folosită ca metodă cvasi modernă se face pe o suprafață magnetică sensibilă, în mod normal oxid de fier, care cu cât este mai subțire cu atât este mai eficientă, ce acoperă un suport din material plastic flexibil pentru benzi și floppy discuri sau platane de aluminiu rigide pentru discurile flexibile (hard).

Suprafața de înregistrare este tratată ca o succesiune de puncte ce primesc un echivalent magnetic de zero sau unu. Locațiile acestor puncte nefiind perfect determinate metodele de înregistrare fac apel la așa numiții marcatori ce permit determinarea exactă a pozițiilor de înregistrare. De aceea apare necesitatea formatării acestor discuri înainte de utilizare, procesul stabilind tocmai acești marcatori de sincronizare. Viteza de rotație a unei diskete este de 300 kPM iar a unui disc rigid de 3600 kPM (a 16-a parte dintr-o secundă/rotație).

Capul de citire al discului se mișcă liniar deasupra mesei rotative pentru o disketă fiind necesar 1/6 dintr-o secundă pentru a ajunge la locația dorită și 1/25 dintr-o secundă la hard disc. Combinația dintre mișcarea circulară a suprafeței discului și cea a capului de disc determină accesul foarte rapid la informație numit și acces random sau nedefinit datorită posibilității de găsire în orice moment a oricărei informații la disc spre deosebire de bandă unde localizarea unei informații presupune desfășurarea benzii pînă la acel loc (acces secvențial). Ca organizare, suprafața discului este împărțită în cercuri concentrice numite piste (tracks) dinspre exterior spre interior pista numărul 1 fiind cea mai exterioară. Numărul de piste variază de la caz la caz terminologia fiind: dublă densitate pentru 40 piste, HI - densitate pentru 80 piste, discurile rigide avînd între 300 și 600 piste fiecare fiind identificată printr-un număr (0 este numărul pistei cea mai exterioară).

Distanța dintre aceste piste este foarte mică astfel că o disketă cu

Elemente de arhitectură a unui microsistem profesional

dublă densitate are 48 piste pe doi centimetri iar cea cu HI - densitate are 96 piste pe doi centimetri. La rîndul ei, fiecare pistă este împărțită în sectoare care pot fi în număr de 8 sau 9 în mod normal, 15 pentru HI - densitate și 17 pentru discurile rigide. La rîndul ei, mărimea unui sector este variabilă între 128 octeți și 1024 octeți 512 octeți fiind standardizată. Citirea sau scrierea discului se face în unități de sector fiecare sector al unei piste fiind numerotat începînd cu cifra unu sectorul numărul zero fiind rezervat pentru identificare și nu pentru stocaj. Ultima problemă tehnică este cea a numărului de fețe înregistrabile a unui disc, floppy discurile putînd fi înregistrate pe o față sau pe amîndouă în timp ce discurile rigide conțin mai multe platane pe un suport, deci pot avea un număr variabil de fețe. Deci, capacitatea de memorare a discului se obține înmulțind numărul de octeți dintr-un sector cu numărul de sectoare de pe o pistă și cu numărul de piste de pe o față și cu numărul de fețe.

Ca organizare constructivă o disketă are 2 orificii unul central care se fixează pe unitatea de disc și altul ce se constituie ca un punct de referință la începutul unei piste. O cămașă dintr-un material special protejează disketa permițînd și rotirea ei în timp ce o deschidere ovală în acest material permite capului de disc citirea sau scrierea pe o disketă. Pe o parte a cămașii protectoare există o tăietură care dacă este acoperită scrierea pe disketă este imposibilă. Tipurile de diskete folosite sînt de 8, 5 1/4, 3 1/2, în varianta de simplă sau dublă densitate. În finalul secțiunii vom dedica o rubrică informativă specială asupra hoilor tehnologii de discuri și unități de memorare (discuri optice, unități laser, acces Bernoulli). Modul de organizare a unei diskete se poate face cu programul Format (sectorizare software) care specifică numărul de părți, piste, sectoare ce se vor ștanța pe disc permițînd, în orice moment, schimbarea lor printr-o nouă procedură Format. Există însă posibilitatea ca o disketă să fie formatată din fabrică fără posibilitatea unei noi sectorizări (disc optic). Discurile rigide (hard disc) sînt protejate de contactul cu aerul sau cu praful deci, neputînd fi mutate din interiorul calculatorului, cele mai cunoscute tipuri fiind cele de 10 MO și 20 MO, 30 MO, existînd în noile tehnologii hard discuri încorporate pe unități detașabile calculatorului, cu capacități între 60-400 MO.

Discurile de 10 MO au 305 cilindri (denumirea echivalentă a unei piste) și 17 sectoare/cilindru, iar cele de 20 MO avînd 615 cilindri. Modul de exploatare a discurilor trebuie văzut întotdeauna în perspectiva sistemului de operare sub care ele sînt gestionate.

Majoritatea calculatoarelor personale profesionale lucrînd sub sistemul de operare DOS este imperios necesară înțelegerea tehnicii de lucru și organizarea discului din perspectiva acestui sistem.

Astfel DOS divide discul în 2 părți: o zonă în care sînt păstrate informațiile despre disc și o zonă de memorare a datelor. Zona de sistem a DOS-ului, care ocupă 2% din suprafața totală, se împarte, la rîndul ei, în 3 părți: înregistrarea de încărcare (boot), zona de alocare a fișierelor (FAT), directorul rădăcină. Zona de boot conține un program care are sarcina încărcării sistemului de operare DOS în memoria calculatorului. Această zonă apare pe orice disc raportînd eroarea în cazul încercării utilizării unui disc oarecare (fără sistem de operare pe el) drept disc sistem. Acest program, care începe de obicei într-un singur sector de disc, cercetează existența pe disc a 2 fișiere IBMBIO.COM și IBMDOS.COM care informează dacă unitatea în uz este de disc sistem sau nu. De asemenea, programul conține o tabelă de elemente de organizare a discului precum

Elemente de arhitectură a unui microsistem profesional

și mărimea zonei FAT și a directorului. Comanda Debug cu opțiunile L0001, 00L200 permite vizualizarea structurii programului de boot. Următoarea zonă FAT este reprezentată printr-o tabelă în care sînt înregistrate informații asupra fiecărei porțiuni de disc.

Pentru managementul spațiului de date disc sistemul DOS îl divide în unități logice numite clustere. Cînd un fișier este înregistrat pe o porțiune de disc acest spațiu disc este asignat (atribuit) fișierului în aceste clustere. Mărimea unui astfel de cluster variază de la un format de disc la altul, cu lungimi între 1 sector și întreg spațiul disc.

Mărimile implicite alocate clusterelor sînt definite, în general, după cum urmează: disketele simplă densitate - 1 sector, disketele dublă densitate și dublă față - 2 sectoare, discurile rigide de 10 MO - 8 sectoare (4096 octeți), discuri rigide de 20 MO - 4 sectoare (2048 octeți).

Alocarea clusterelor este sarcina FAT-ului care ține evidența lor prin numere într-o tabelă. Numărul înregistrat în tabelă indică dacă clusterul căruia îi este asociat este ocupat cu date sau nu (valoarea 0 dacă este liber). Mai mult, numerele în ordine secvențială indică și legătura dintre mai multe clustere dedicate aceleiași zone de date fișier.

Valoarea primului număr identificator de cluster este 2 valorile 0 și 1 fiind rezervate de DOS. Intrările în FAT sînt numere pe 12 biți putînd acomoda 4 k numere, iar pentru AT acesta este mărit pentru a ține evidența a 10000 de clustere (numere pe 16 biți). Pentru orice fișier intrarea în directorul său indică numărul primului cluster alocat acestuia fiecare alt cluster conținînd numărul următorului din lanțul celor alocate fișierului sau marcatorul de sfîrșit de lanț (sfîrșit de fișier) care este FFF (pentru FAT-uri cu intrare pe 12 biți) sau FFFF (pentru FAT-uri pe 16 biți). Porțiunile din disc inutilizabile (sectoare stricate) sînt identificate în FAT prin valorile FF7 (sau FFF7). Codurile numerice între FF0 și FFF (sau FFF0 și FFFF) sînt rezervate pentru utilizări viitoare.

Metoda codificării numerelor pe 12 biți în 3 octeți este adaptată folosirii lor din limbaj de asamblare dar este destul de dificil să le identificăm dacă încercăm o vizualizare hexa a lor.

Primul octet al FAT-ului conține identificatorul formatului de disc (codul FE identifică o disketă simplă densitate de 160 k) sistemul DOS țînînd întotdeauna, pentru siguranță, două copii ale tablei FAT.

Ultima parte a zonei sistem a discului este directorul rădăcină care există pentru orice disc (subdirectorii sînt opționali dar rădăcina nu). Sarcina directorului este evidența tuturor fișierelor din disc.

Pentru fiecare fișier există o intrare în director (poziție) care conține numele fișierului pe 8 caractere, extensia acestuia pe 3 caractere, mărimea în octeți a fișierului, precum și data ultimei modificări (creări) a fișierului. Instrucțiunea DOS ">DIR" listează toate fișierelor înregistrate în director. Mai există încă 2 informații înregistrate în director pentru fiecare fișier și anume: numărul clusterului de început a datelor din fișier și atributul de fișier (putem avea următoarele categorii de fișiere: fișiere sistem; ascunse; exploatabile doar în citire; normale; arhivate).

Orice intrare într-un fișier are 32 de biți mărimea directorului rădăcină variînd în funcție de disc astfel: 4 sectoare pentru disketă simplă densitate, 7 sectoare pentru o disketă dublă densitate, 32 de sectoare pentru un disc de 20 MO (acomodînd, astfel, 512 intrări de fișier).

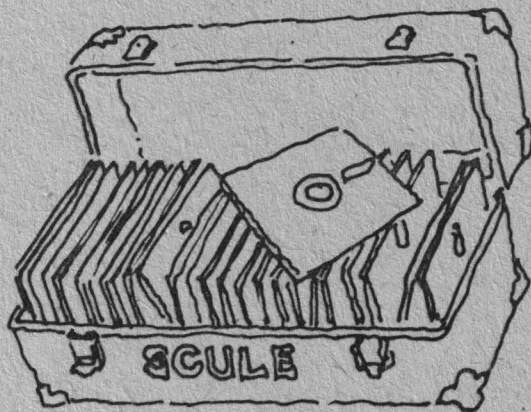
Subdirectorii sînt o mixtură între fișier și directorul rădăcină. Astfel,

Elemente de arhitectură a unui microsistem profesional

evidența oricărui subdirector este ținută de FAT în tabela pentru un fișier oarecare. În schimb, atunci când se predă controlul unui subdirector acesta se comportă ca un director rădăcină în raport cu fișierele care sînt în componența lui. Există însă deosebiri efective între directorul rădăcină și subdirectori, și anume: rădăcina este unică și cu lungime fixă în timp ce putem avea oricîți subdirectori care pot avea lungimi diferite în funcție de numărul de fișiere existent în ele.

În cîmpul rezervat numelui de fișier există două coduri ce se folosesc în primul octet al numelui. Dacă octetul este 0 intrarea în director (și următoarele) n-a fost folosită niciodată (permite programului DIR să se oprească după investigarea tuturor fișierelor existente nefiind nevoie să parcurgă toată rădăcina).

Codul E5 este utilizat pentru a marca un fișier șters (nici o informație din fișier nu e alterată la ștergere în afară de primul octet din numele fișierului și spațiul său alocat în FAT).



Fundamente de matematică ale programării logice în inteligența artificială (I).

Elemente de calculul predicatelor în inteligența artificială

1. Relații bine definite

Programarea logică în inteligența artificială comportă un formalism calculatoriu, avînd la bază trei elemente constitutive ce formează așa numitul sistem de producție al IA(*) și anume: regulile de producție sau combinaționale, o bază globală de date și un sistem de control.

Aceste trei mari entități acționează global asupra unui produs al IA în sensul că regulile acționează asupra bazei de date fiecare fiind realizată sau nu în bază, de satisfacerea lor depinzînd condițiile de aplicabilitate ale regulei respective, condiții dictate de sistemul de control, care dictează atît începutul cît și sfîrșitul unei sesiuni de lucru cu sistemul, în funcție de satisfacerea unor reguli de început și sfîrșit impuse. Ceea ce este particular oricărui sistem al IA constă în posibilitatea de acces la baza de date a oricărei reguli de producție, fără restricții, aceste reguli nefiind nici apeluri tip subrutină la alte baze de date, ele acționînd strict în cadrul bazei sistemului, adăugarea de fapte la elementele bazei putîndu-se face în cadrul acțiunii regulilor prin așa numitele "asertiuni de fapte obținute prin instantări" ale regulilor.

Astfel, rezolvarea unei probleme de IA folosind un sistem de producție presupune specificarea bazei de date, a regulilor și a controlului strategiilor, realizare ce este cunoscută ca problema reprezentării în IA.

Însă această problemă a reprezentării de multe ori, în cazuri concrete devine dificilă, informațiile ce trebuiesc codificate în baza de date făcînd parte din elemente curente ale limbajului natural ce nu se pot ordona în tabele matriceale sau în liste cu o anumită legătură sau structură, utilizarea diversă a entităților reclamînd capacitatea sistemului de manipulare a unor mulțimi logice de afirmații sau fapte.

Astfel, aspectele fundamentale ale formalismului logic, ca reprezentările unor aserții, deducerea inferențelor într-o mulțime de expresii, stabilirea de dezvoltări recursive ale elementelor unui limbaj sînt realizate pe două nivele și anume: i) nivelul individual al programării logice sau calculul predicatelor de ordin unu și ii) nivelul doi sau al predicatelor de ordin II sau de mulțimi, ce se ocupă cu studiul global al mulțimilor și proprietăților lor, incluzînd calculul lambda și variabile predicative, ce permite tratarea funcțiilor și a relațiilor ca date de ordin 1 sau, altfel spus, ca obiecte primare.

În cele ce urmează ne vom ocupa de dezvoltarea programării logice pe cele 2 nivele, urmărind evoluția firească a calculului, deci pornind cu analiza predicatelor de ordin unu, precum și a limbajului și metodelor logice generate de ele, ocupîndu-ne de folosirea lor în sistemele de producție ale IA.

Aceste predicate, ca elemente primare constitutive ale unui limbaj predicativ, pot fi privite ca literele unui alfabet de simboluri, simboluri caracterizate de o anumită sintaxă ce le permite asocierea într-o formă definită

(*) Prin IA vom înțelege conceptul de inteligență artificială.

prin expresii legale ale limbajului. Astfel, o expresie formată din simboluri logice predicative ce este legitimă pentru limbaj o vom numi relație bine definită (**rbd**) ținând cont că, constitutiv, componentele limbajului generat de calculul predicatelor de ordin unu sînt: {simboluri predicative (sau nume de predicate), simboluri variabile (sau variabile ce țin locul unor entități), simboluri funcție (sînt funcții definite recursiv cu ajutorul concatenării de predicate), simboluri constante (denumite și fapte în baza de date)}, la care se adaugă mulțimea unor simboluri formale fără interpretare, ca paranteze, virgule, semne de punctuație din limbajul curent, într-o ordine dictată de regulile de asociere ale limbajului.

Să privim în continuare mai detaliat elementele de sintaxă generale ale limbajului calculului predicativ.

Elementele constitutive ale limbajului sînt definite prin așa numitele **formule atomice**, constituite din **simboluri predicative** și **termeni**, unde termenii pot fi constante, simboluri funcții sau simboluri variabile. În general simbolurile predicative reprezintă o relație din domeniul limbajului curent, în cele mai multe cazuri predicatul fiind identificat cu verbul unei propoziții, dar existînd cazuri cînd poate fi adjectiv sau chiar substantiv, în funcție de modul de reprezentare a frazei sau propoziției respective. Astfel, o expresie predicativă are forma:

$$(1) \quad \text{predicat} (\text{termen}_1, \text{termen}_2, \dots, \text{termen}_n)$$

expresia (1) constituind un **rbd** în calculul predicatelor dacă ei i se dă o interpretare prin stabilirea unei corespondențe între elementele constitutive ale lui (1), ce sînt și părți ale limbajului predicativ și relațiile, entitățile, funcțiile din domeniul vorbirii curente.

Astfel, dacă unui simbol predicativ i se poate asocia verbul, adjectivul sau substantivul unei fraze la fel unui simbol constant i se asociază o entitate (de obicei substantiv) din vorbirea curentă, aceste asignări definind nimic altceva decît semantica limbajului predicativ. De obicei, în elaborarea unei aplicații ne situăm într-un anumit domeniu al limbajului curent, cunoscînd din logica îmbinării frazelor și un istoric al entităților folosite, deci știm, în virtutea unei logici elementare dacă o expresie predicativă nou formată este sau nu **rbd**, ei asociindu-i valoarea de adevăr (A) dacă este o **rbd**, sau o valoare (F) (fals) dacă nu este o asociere logică a entităților părții de cunoaștere afectată aplicației.

Să analizăm în continuare, din punct de vedere al formalismului lexicografic, scrierea formelor de **rbd** ale limbajului calculului predicativ, sub formă de formule pe care le vom denumi **atomice**, formule de tip (1).

Să presupunem că vrem în continuare să prezentăm 2 formule **atomice**, una din domeniul geografiei și alta din domeniul relațiilor unui arbore genealogic. De exemplu faptul că există un drum între BUCUREȘTI și BRAȘOV poate fi reprezentat prin:

$$(2) \quad \text{drum} (\text{BUCUREȘTI}, \text{BRAȘOV}, 173)$$

al treilea termen al relației reprezentînd și distanța dintre termen 1 = București și termen 2 = Brașov.

Aici atît București, Brașov cît și 173 sînt simboluri constante. Formula (2) poate fi cuantificată și sub forma:

$$(2') \quad \text{drum} (\text{Oraș}, \text{Oraș}, \text{Distanță})$$

unde Oraș este simbol variabil, iar Distanță este un întreg variabil sau mai formal:

$$\text{drum} (X, X, D)$$

Elemente de calculul predicatelor în inteligența artificială

cu X = simbol și D = întreg.

În domeniul arborelui genealogic dorim reprezentarea formală a următoarelor forme lexicale:

"Cineva este părintele cuiva";

"O persoană este căsătorită cu altă persoană";

"Cineva este copilul cuiva";

"O persoană este urmașul altei persoane".

Avem posibile construcții:

părinte (Ion, Maria); părinte (Ioana, Maria);

părinte (Ion, Dan);

părinte (Dan, Monica);

înțelegând că "Ion este părintele lui Maria" sau pentru un formalism variabil părinte (X, Y), "X este părintele lui Y". La fel:

căsătorit (Dan, Carmen);

căsătorit (Ion, Ioana);

căsătorit (Carmen, Dan).

înțelegând că predicatul căsătorit (X, Y) are sensul: "X este căsătorit cu Y" și la fel "Y este căsătorit cu X", ambele constituindu-se ca $rb\bar{d}$ în domeniul discursului. Se impune în ambele predicate, distincția $X \neq Y$, altfel asistăm la ambiguități logice, formulele părinte (X, X) sau căsătorit (X, X) nefiind logice din punctul de vedere al realității bunului simț. Astfel de formule atomice nu mai sînt $rb\bar{d}$, valoarea lor de adevăr fiind F (fals). Astfel, căsătorit (Dan, Dan) are valoarea fals nefiind deci un $rb\bar{d}$.

De asemenea, în funcție de relațiile asociative în domeniul discursului real, formulele atomice predicative pot fi false în funcție de asocierea termenilor. Astfel:

drum (București, Brașov, 5000)

este o relație falsă, deci nu este o $rb\bar{d}$, bunul simț dictîndu-ne imposibilitatea parcurgerii distanței de 5000 km între București și Brașov (o astfel de relație poate fi o $rb\bar{d}$ deci o formulă adevărată, într-un domeniu al excentricității, de exemplu, deci universul realității curente joacă rol esențial în sfera sistemelor de producție ale IA alese).

Formulele atomice pot avea ca termeni constitutivi și simboluri funcții sau, așa cum le defineam, concatenări de predicate, ca de exemplu:

căsătorit (părinte($X, Maria$), părinte($Y, Maria$))

înțelegând că X și Y sînt simbolurile care se substituie numelor părinților lui Maria. Putem însă ajunge la ambiguitate în sensul că X și Y pot lua aceeași valoare, Ion, ajungînd la un fals de forma căsătorit (Ion, Ion), deci se impun mențiuni suplimentare predicative reprezentabile prin formulele:

bărbat (Ion), femeie (Ioana).

Astfel căsătorit (părinte ($X, Maria$), părinte ($Y, Maria$)) devine un $rb\bar{d}$ sub restricțiile bărbat (X), femeie (Y).

2. Calculul propozițional și cuantificare

Am văzut în ultimele exemple că, în domeniul vorbirii curente formulele atomice singulare cu termeni simboluri constante nu sînt suficiente în acoperirea sensurilor valențe ale realității reprezentate.

Apare atunci necesitatea naturală de compunere a formulelor atomice, care primate ca entități binare pot fi false sau adevărate. Astfel, sarcina compunerii formulelor atomice devine ușoară în sensul că fiind date 2 formule f_1 și f_2 , $f_1 \circ f_2$

Elemente de calculul predicatelor în inteligența artificială

poate fi o nouă formulă atomică în care simbolul "o" al compunerii ia valori în mulțimea operanzilor din calculul propozițional astfel:

"o" \in { \neg , \vee , \Rightarrow , \wedge , \Leftrightarrow , }.

Dacă "o" = " \vee ", atunci $f_1 \vee f_2$ se numește disjuncția formulelor f_1 și f_2 , avînd proprietatea că, dacă f_1 și f_2 sînt rbd-uri atunci $f_1 \vee f_2$ este un rbd (relație bine definită). Dacă "o" = " \wedge ", atunci $f_1 \wedge f_2$ se numește conjuncția formulelor f_1 și f_2 cu proprietatea că dacă f_1 și f_2 sînt rbd-uri atunci și $f_1 \wedge f_2$ este un rbd.

Dacă "o" = " \neg ", atunci f_i se zice negarea formulei f_i , fiind un rbd dacă f_i este rbd.

Dacă "o" = " \Rightarrow " atunci $f_1 \Rightarrow f_2$ se numește implicație cu proprietatea că pentru f_1, f_2 rbd-uri și $f_1 \Rightarrow f_2$ este un rbd. Vom vedea că, datorită valorii de adevăr a operatorului " \neg " putem genera rbd cu valori de adevăr absurde, dacă implicația are loc în domeniul disjuncte ale vorbirii.

Considerăm utilă o reamintire a valorilor de adevăr generate de operatorii propoziționali aplicați la două formule f_1 și f_2 cu valori de adevăr alternative.

Tabela " \vee ":

f_1	f_2	$f_1 \vee f_2$
A	F	A
F	A	A
A	A	A
F	F	F

Tabela " \wedge ":

f_1	f_2	$f_1 \wedge f_2$
A	F	F
F	A	F
A	A	A
F	F	F

Tabela " \neg " (non):

f_i	$\neg f_i$
A	F
F	A

Tabela " \Rightarrow ":

f_1	f_2	$f_1 \Rightarrow f_2$
A	F	F
F	A	A
A	A	A
F	F	F

Să facem o observație utilă în cele ce urmează și anume că implicația $f_1 \Rightarrow f_2$ poate fi înlocuită cu formula echivalentă $\neg f_1 \vee f_2$, avînd valori echivalente. Mai putem, de asemenea, introduce și o funcție φ_f caracteristică, asociată formulelor atomice f astfel:

$$\varphi : M_{\text{rbd}} \longrightarrow \{0,1\}$$

$\varphi(f) = 0$ dacă valoarea rbd-ului f este F
 $\varphi(f) = 1$ dacă valoarea rbd-ului f este A

unde M_{rbd} este mulțimea rbd-urilor sistemului folosit. Atunci avem proprietățile lui φ :

i) $\varphi^2(f) = \varphi(f)$;

ii) $\varphi(f_1 \vee f_2) = \varphi(f_1) + \varphi(f_2) - \varphi(f_1) \cdot \varphi(f_2)$;

absurde, dacă implicația are loc în domenii disjuncte ale vorbirii.

Considerăm utilă o reamintire a valorilor de adevăr generate de operatorii propoziționali aplicați la două formule f_1 și f_2 cu valori de adevăr alternative.

iii) $\varphi(\neg f) = 1 - \varphi(f)$;

iv) $\varphi(f_1 \wedge f_2) = \varphi(f_1) \cdot \varphi(f_2)$.

Demonstrarea formulelor i-iv este imediată ținând cont de tabelele de adevăr precedente.

Echivalența formulelor $f_1 \Rightarrow f_2 \Leftrightarrow \neg f_1 \vee f_2$ se scrie:

$$\begin{aligned} \varphi(f_1 \Rightarrow f_2) &= (\neg f_1 \vee f_2) = \varphi(\neg f_1) + \varphi(f_2) - \varphi(\neg f_1) \cdot \varphi(f_2) = \\ &= 1 - \varphi(f_1) + \varphi(f_2) - (1 - \varphi(f_1)) \cdot \varphi(f_2) = \\ &= 1 - \varphi(f_1) + \varphi(f_2) - \varphi(f_2) + \varphi(f_1)\varphi(f_2) = \\ &= 1 - \varphi(f_1) + \varphi(f_1) \cdot \varphi(f_2) \end{aligned}$$

deci introducînd formula:

v) $\varphi(f_1 \Rightarrow f_2) = 1 - \varphi(f_1) + \varphi(f_1) \cdot \varphi(f_2)$

se verifică cu tabel certitudinea ei.

În consecință putem demonstra imediat cîteva echivalente uzuale de care vom avea nevoie în cele ce urmează, dintre care:

a) legile De Morgan: $\neg(f_1 \wedge f_2) \Leftrightarrow \neg f_1 \vee \neg f_2$;

$$\neg(f_1 \vee f_2) \Leftrightarrow \neg f_1 \wedge \neg f_2$$

b) legea inversiunii: $f_1 \Rightarrow f_2 \Leftrightarrow \neg f_2 \Rightarrow \neg f_1$;

c) legea distribuției: $f_1 \wedge (f_2 \vee f_3) \Leftrightarrow (f_1 \wedge f_2) \vee (f_1 \wedge f_3)$;

$$f_1 \vee (f_2 \wedge f_3) \Leftrightarrow (f_1 \vee f_2) \wedge (f_1 \vee f_3)$$

Dacă pînă acum am stabilit relații globale asupra formulelor atomice indiferent de mulțimea variabilelor asociate lor vom face o extensie de limbaj, în sensul următor:

Fie f un predicat căruia îi dăm o interpretare semantică (de exemplu f este substantivul bărbat).

Cum predicatul "bărbat" poate lua o mulțime de valori în domeniul studiat îl putem reprezenta funcțional ca bărbat (X) cu X variabilă în domeniul entităților unei baze de date asociată sistemului IA în contextul căruia s-a definit formula predicativă "bărbat (X)".

Avem următoarele 2 situații posibile:

a) bărbat (X) - poate fi adevărată indiferent de mulțimea valorilor lui X în domeniul entităților;

b) bărbat (X) - poate fi adevărată pentru cel puțin o valoare a lui X în domeniul entităților.

Cele 2 proprietăți le putem exprima formal cu ajutorul cuantificatorilor astfel:

a) $(\forall)X \Rightarrow \text{bărbat}(X)$;

b) $(\exists)X$ astfel ca bărbat (X).

În general pentru un predicat oarecare P , avînd o semnificație dată, semnific (P), propoziția " $(\forall)X, P(X)$ " are valoare de adevăr A pentru toate valorile lui X în domeniul entităților, iar " $(\exists)X$ astfel încît $P(X)$ " are valoare de adevăr A pentru cel puțin o atribuire de entitate lui X . Trezînd la cuantificarea compunerilor de predicate, deci a rbd-urilor,

dacă variabilele cuantificate sînt globale rbd-ului spunem că ele sînt variabile mărginite, altfel le vom numi variabile libere.

Prin cuantificarea globală vom înțelege că dacă o variabilă X participă în formulele f_1 și f_2 atunci participă în $f_1 \circ f_2$. De exemplu să stabilim cuantificarea propoziției "Toate găinile albe fac ouă" putem distinge predicatele:

1) f_1 : Găină (X) - cu X parcurgînd mulțimea găinilor;

2) f_2 : Ou (X, Alb) - cu X parcurgînd mulțimea găinilor;

- semnificația predicatului este: Orice " X alb face ouă".

Avem implicația cuantificată:

$$(\forall)X \text{ Găină}(X) \Rightarrow \text{Ou}(X, Alb) (*)$$

Cum variabila X parcurge o mulțime de entități definită, Q vom numi mărginită. Relații de tipul (*) în care variabilele sînt mărginite le vom numi propoziții, observînd că un rbd cuantificat rămîne un rbd.

Să mai remarcăm că ne referim la predicate de ordin 1, deci cuantificarea poate avea loc doar asupra simbolurilor variabile și nu asupra simbolurilor predicat sau simbolurilor funcție, deci o relație " (\forall) predicat, predicat (Termen)" nu constituie un rbd iar, de multe ori, analiza propozițională a unui rbd cuantificat nu poate fi făcută practic dacă cuantificatorul parcurge o mulțime infinită de entități.

Practic, orice propoziție în vorbirea curentă poate fi cuantificată dacă mai ținem cont și de echivalențele între legile propoziționale anterioare. Un exemplu:

Pentru orice mulțime X , \exists o mulțime Y , a.f. numărul de elemente al lui Y este mai mare ca cel al lui X ". Avem predicatele Mult (X) ce reprezintă mulțimile din domeniul entităților, Număr (X, N), unde N este numărul de elemente din X și Maimare (N, M) unde M este mai-mare ca N . Atunci forma cuantificată a propoziției este:

$$(\forall)X \text{ M}(X) \Rightarrow (\exists)Y, (\exists N)(\exists M), \\ \text{M}(Y) \wedge \text{Număr}(X, N) \wedge \text{Număr}(Y, M) \wedge \text{Maimare}(N, M)$$

3. Rezoluție generală

Pornind de la o mulțime de rbd-uri ce constituie baza semantică a unui limbaj, putem defini noi rbd-uri formulînd diverse reguli și condiții numite reguli ale inferenței.

Vom numi rbd-urile obținute prin derivarea teoremelor de bază, iar regulile de inferență care au dus de la un rbd-inițial la rbd-ul derivat final un arbore de demonstrație sau mai pe scurt o demonstrație a teoremei.

În multe din problemele programării logice se urmărește găsirea recursivă a unui arbore de demonstrație pentru o anumită teoremă sau "realizare".

Vom vorbi în capitolele ce urmează pe larg despre regulile inferenței în cadrul rezolvării unor clase de probleme. Ne vom limita, în continuare, la definirea regulilor fundamentale și acceptarea lor fără demonstrație pînă în momentul cînd bagajul de cunoștințe ne va permite conștientizarea demonstrațiilor lor. Dintre cele mai importante sînt regulile **modus-ponens** (realizează rbd f_2 din f_1 cu $f_1 \Rightarrow f_2$), regula **specializării universale** (deduce f (termen constant) din $(\forall)X, f(X)$), unificarea (procedeul fundamental) ce constă în esență în găsirea unor substituții pentru variabile pentru care expresiile asupra variabilelor cărora le sînt aplicate substituțiile devin identice.

O regulă importantă a inferenței o constituie regula rezoluției. Ea

operează asupra unor rbd-uri organizate ca o disjuncție de literali (prin literali se înțeleg formulele atomice precum și negațiile lor). Vom conveni să numim astfel de rbd-uri, reguli clauzale sau pe scurt clauze. Vom arăta că aceste clauze sînt forme general valabile de expresie a rbd-urilor, folosirea rezoluției în demonstrarea unei teoreme presupunînd transformarea rbd-urilor de la care se pornește demonstrarea, în clauze. Formal, vom arăta că dacă rbd-urile:

$$\{f_1, \dots, f_n\} \Rightarrow f \text{ cu } f \text{ rbd, atunci}$$

$$\{cl(f_1), \dots, cl(f_n)\} \Rightarrow f, \text{ unde } cl(x) = \{\text{clauzele ce derivă din rbd-ul } x\}.$$

Vom face următoarea convenție generală. Dacă $f(x,y)$ este un literal sau un rbd, dacă $(\exists)A$ și B entități pentru care $x=A$ și $y=B$ are sens, adică A face parte din mulțimea valorilor posibile ale lui x și analog B face parte din mulțimea valorilor posibile ale lui y , unde A și B sînt simboluri constante, atunci vom numi $f(A,B)$ o instantare fundamentală sau particulară a rbd-ului $f(x,y)$ pentru substituția $(x,y) = (A,B)$, iar o clauză $cl((A,B))$, clauză particulară.

Cele trei aspecte ale rezoluției de care ne vom ocupa în continuare sînt:

- i) conversia și scrierea unui rbd sub formă de clauze deci sub formă de disjuncții de rbd-uri;
- ii) acțiunea rezoluției asupra clauzelor particulare;
- iii) acțiunea rezoluției asupra clauzelor care conțin variabile.

Cum ii) și iii) sînt oarecum distincte de i) ne vom ocupa mai întîi de ele. Principial regulile ii) și iii) acționează astfel: din 2 clauze cl_1 și cl_2 , numite clauze părinte, se poate obține o clauză fie cl_{12} , numită rezolvantă a lui cl_1 și cl_2 , obținută prin disjuncția lui cl_1 și cl_2 , $cl_1 \vee cl_2$, din care se elimină perechile complementare de forma f și $\neg f$ de rbd-uri.

În cazul ii) putem defini o serie de posibili rezolvenți din compuneri de clauze părinte:

a) Din clauzele părinte:

$$cl_1 \text{ și } \neg cl_1 \vee cl_2 \text{ rezultă rezolvanta } cl_2 \iff (cl_1 \Rightarrow cl_2);$$

b) Din clauzele părinte:

$$cl_1 \vee cl_2 \text{ și } \neg cl_1 \vee cl_2 \text{ rezultă rezolvanta: } cl_2;$$

c) Din clauzele părinte: $cl_1 \vee cl_2$ și $\neg cl_1 \vee \neg cl_2$ rezultă rezolvanta:

$$cl_1 \vee \neg cl_1 \text{ și } cl_2 \vee \neg cl_2, \text{ ambele fiind tautologii.}$$

(numim tautologie în domeniul propozițiilor $\{P_i\}$ o funcțională φ cu $\varphi(P_i) = 1$ pentru P_i adevărată; $\varphi(P_i) = 0$ pentru P_i falsă);

d) Din $\neg cl_1$ și cl_1 rezultă rezolvanta NIL, numită și clauza vidă.

e) Din $\neg cl_1 \vee cl_2$ și $cl_2 \vee cl_3$ sau $(cl_1 \Rightarrow cl_2)$ și $(cl_2 \Rightarrow cl_3)$ rezultă $cl_1 \Rightarrow cl_3$ sau $\neg cl_1 \vee cl_3$ rezolvantă obținută prin tranzitivitate.

Despre aplicarea clauzelor ce conțin variabile, deci a lui (iii) vom discuta pe larg în secțiunea dedicată procesului de unificare.

Să introducem în continuare o noțiune și anume aceea de funcțională Skolem.

Să considerăm următoarea regulă rbd:

$$(\forall)x, (\exists)y f(x,y).$$

care se poate interpreta ca: "Pentru orice x , există un y (dependent posibil de x) pentru care are loc $f(x,y)$. Deci y -ul găsit depinde de x , el putînd fi scris:

$$y = S(x).$$

Să considerăm aceluși funcțională S definită pe mulțimea tuturor x -ilor

pentru care există imaginea $S(x)$, deci $S:\{x\} \rightarrow \text{Im}S$.

Numim aplicația S funcțională Skolem iar scopul ei constă în eliminarea cuantificatorului existențial (\exists). Atunci avem echivalența:

$$(\forall x)(\exists y)f(x,y) \Leftrightarrow (\forall x)(f(x,S(x))).$$

Să arătăm în continuare i) sau cum orice rbd poate fi convertit într-o mulțime de clauze.

Procesul constă în următorii pași logici:

1) Se elimină simbolurile de implicație, " \Rightarrow ". Aceasta, este posibil, după cum am arătat, folosind echivalența " $f_1 \Rightarrow f_2 \Leftrightarrow \neg f_1 \vee f_2$ ".

2) Reducerea simbolurilor " \neg ". Procesul constă în folosirea legilor calculului propozițional precedente astfel ca simbolul " \neg " să apară cel mult o dată în fiecare formulă.

De exemplu o formulă de forma:

$$\neg(\forall x)(\neg f_1(x,y) \vee f_2(y)) \text{ devine:}$$

$$(\exists x)(f_1(x,y) \wedge \neg f_2(y))$$

cu ajutorul formulei De Morgan $\neg(f \vee g) = (\neg f) \wedge (\neg g)$ și a observației că $\neg(\forall) = (\exists)$; $\neg(\exists) = (\forall)$.

3) Standardizarea variabilelor. Acest procedeu constă în redenumirea fiecărei variabile sub un cuantificator (\forall) sau (\exists) astfel ca fiecare cuantificator să aibă propria sa variabilă formală fără ca acest proces să afecteze valoarea de adevăr a rbd-ului.

De exemplu avem echivalența:

$$(\forall x)(f_1(x) \Rightarrow \exists(x)f_2(x)) \Leftrightarrow (\forall x)(P(x) \Rightarrow (\exists y)f_2(y))$$

$$\text{sau } (\forall x)(f_1(x)) \wedge (\exists x)(f_2(y,x) \wedge \sim f_1(x))$$

se pot înlocui cu:

$$(\forall x)(f_1(x)) \wedge (\exists u)(f_2(y,u) \wedge \sim f_1(u))$$

4) Eliminarea cuantificatorilor existențiali. Acest pas constă în Skolemizarea rbd-urilor după regula descrisă în definiția funcționalelor Skolem, cu observația că simbolurile folosite în funcțiile Skolem nu trebuie să fi apărut anterior în nici un rbd.

Dacă variabila existentă după cuantificatorul existențial nu este precedată de nici o variabilă definită universal, în sensul că nu avem rbd ce începe cu " $(\forall x)\exists y$ " ci rbd ce începe cu " $(\exists x)$ ", atunci funcționala Skolem nu are nici un argument ci se identifică cu o constantă, ca de exemplu: " $(\exists x)f(x,y)$ " devine $f(A,y)$ cu A simbol constant.

5) Conversie în forma matriceală (prenex). Cum în această fază nu mai există cuantificatori existențiali, iar fiecare cuantificator universal își are propria sa variabilă în numerotare literală distinctă, toți cuantificatorii universali se scot în afara regulilor rezultând astfel o formulă ce are ca prefix mulțimea cuantificatorilor urmată de o formulă fără cuantificatori denumită formal matrice.

Astfel, o relație de forma:

$$(\forall x)(\neg f_1(x) \vee ((\forall y)\neg f(y) \vee f_1(x,y)))$$

devine:

$$(\forall x)(\forall y) \{ \neg f_1(x) \vee (\neg f(y) \vee f_1(x,y)) \}$$

6) Scrierea "matricei" sub formă conjunctivă. Procesul constă în scrierea "matricei" ca o conjuncție de disjuncții de literalii (rbd-uri și \neg rbd-uri), forma obținută fiind cunoscută ca forma normală conjunctivă. De exemplu: $(f_1 \vee f_2) \wedge (f_1 \vee f \vee \neg f_3) \wedge f_2$ este o formă normală conjunctivă sau o conjuncție de disjuncții. În scrierea acestei forme se folosesc regulile

Elemente de calculul predicatelor în inteligența artificială

distribuției profesionale: de exemplu $(f_1 \vee (f_2 \wedge f_3))$ se înlocuiește cu $(f_1 \vee f_2) \wedge (f_1 \vee f_3)$.

7) Eliminarea cuantificatorilor universali.

Eliminarea este imediată din 6) ținând cont că ordinea de eliminare nu are importanță.

8) Eliminarea simbolurilor " \wedge ".

Putem astfel înlocui $(f_1 \wedge f_2)$ cu mulțimea $\{f_1, f_2\}$ obținând în final o mulțime finită de rbd-uri ce sînt conjuncții de disjuncții de literali, rbd-uri numite clauze.

9) Redenumirea variabilelor

Putem renumera sau redefini variabilele astfel ca nici un simbol variabil cu o anumită relație să apară în mai mult de o clauză, proces numit adeseori standardizarea separată a variabilelor.



**Limbaje de programare
în designul sistemelor de operare și al aplicațiilor.
Limbajul C (I).**

Cum, când, ce C!

Ne propunem un curs de învățare și aplicare a celui mai utilizat limbaj de programare a generației a IV-a de calculatoare.

Prezentarea se va face gradat prin exemple de sine stătătoare încercînd să surprindă treptat toate domeniile de aplicabilitate ale acestui limbaj în arta programării.

1. Istoric

Limbajul de programare C a fost proiectat de către Dennis Ritchie pentru a pune la dispoziția sistemului de operare UNIX un limbaj de programare eficient pentru scrierea software-ului de sistem. Prima implementare a fost efectuată în 1972 pe PDP-11 la Bell Laboratories.

C prezintă următoarele caracteristici:

- simplitate în utilizare;
- eficiența codului generat este controlabilă de către utilizator;
- este un limbaj structurat: conține mecanisme suficient de bogate pentru a permite codificare și folosirea metodei programării structurate;
- conciziune: permite o exprimare compactă și cu puține cuvinte a programelor.

Ca precursori ai lui C pot fi considerate următoarele limbaje de programare:

• CPL (Combined Programming Language) propus în 1963 la Cambridge - London;

• BCPL (Basic Combined Programming Language) definit pentru a realiza software de sistem portabil de către Richards în 1967 la Cambridge (există o implementare și pe FELIX);

• B, limbaj utilizat de către proiectantul s.o. UNIX, Ken Thompson împreună cu limbajul de asamblare pentru dezvoltarea acestuia;

• C, construit pentru a elimina deficiențele lui B.

Dacă considerăm următorul mod de ierarhizare a limbajelor de programare:

concret	1. limbaj mașină (hard);
(trecut)	2. limbaj de asamblare;
"	3. limbaj orientat mașină;
"	< "C";
"	4. limbaj orientat pe problemă;
"	.
"	.
"	1. limbaj neprocedural;
"	.
"	.
"	.
abstract	n. dialog liber (natural)
(viitor)	

atunci putem considera locul lui C între 3 și 4.

Prin această poziție în ierarhie, C prezintă o dualitate:

- posedă mecanisme de nivel scăzut prin care se poate face referire la elemente ale mașinii (registre, biți, cuvinte, pointeri) dar suficient de abstracte pentru a asigura o independență relativă față de o mașină dată;
- posedă mecanisme de nivel relativ înalt specificate limbajelor orientate pe probleme.

De exemplu, s.o. UNIX V7 a fost descris în 10000 linii sursă în C și mai puțin de 1000 în limbaj de asamblare.

2. Concepte de bază

Conceptul de bază în C este cel de funcție. Funcția este o entitate compilabilă separat. Poate accepta parametrii de intrare la apel și poate returna o valoare după execuție.

Programarea în C este adecvată pentru stilul programării modulare. Funcțiile referite într-un modul pot fi definite în textul sursă al programului sau pot fi luate dintr-o bibliotecă.

Invocarea funcției se face prin numele ei însoțit de o eventuală listă de parametrii și urmat, eventual, de ::

```
nume_funcție(p1,p2,...,pn);
```

Dacă funcția returnează o valoare atunci numele său poate fi citat ca operand în expresii aritmetice și/sau logice. Parametrii funcției pot fi constante, variabile, funcții și expresii. Un program în C este o funcție care poate apela alte funcții (scrise eventual în alt limbaj) care are numele standard main().

Structura generală a unei funcții este:

```
nume_funcție(p1,p2,...,pn)
```

-

- lista de declarații de date
- (parametrii de apel ai funcției)

-

{

-

- instrucțiuni specifice funcției reprezentate de:
 - instrucțiuni de declarare a datelor locale funcției;
 - instrucțiuni executabile (inclusiv apeluri de funcții);

}

- { } includ sau definesc un bloc de instrucțiuni sau o secvență.

În C, la apelul unei funcții, datele se transmit prin stivă. Funcția primește o copie a valorilor datelor necesare. În unele cazuri datele pot fi furnizate și prin adrese.

Funcțiile în C sînt recursive. Dacă funcția nu are parametrii atunci se definește prin nume_funcție().

Structura programului este:

```
main()
```

{

-

- instrucțiuni

-

}

Cum, cînd, ce C!

Orice program în C operează cu trei periferice standard (influență UNIX) și anume:

- intrare (stdin), redirectabil cu <spec_fis;
- ieșire (stdout), redirectabil cu >spec_fis;
- erori (stderr).

Prin specificare >>spec_fis se va realiza adăugarea la sfîrșitul fișierului identificat de spec_fis.

Pentru operațiile de I/O C nu conține instrucțiuni specifice ci un set de funcții standard pentru efectuarea acestora. Aceste funcții sînt singurele care se adaptează la specificul unui calculator.

Pentru rularea unui program se parcurg pașii:

- crearea, cu ajutorul unui editor de texte, a textului sursă al programului care este preferabil să aibă terminația .C;
- compilarea programului;
- link-editare;
- rulare efectivă.

De exemplu, dacă dorim să aifșăm, la ieșirea standard a calculatorului (în general ecranul unui video terminal) mesajul "De CRACIUN ne-am luat rația de LIBERTATE!" vom putea scrie următorul program :

```
#include <stdio.h>
main()
{
    printf ("De CRACIUN ne-am luat rația de LIBERTATE!");
}
```

În această secvență de instrucțiuni funcția printf() este o funcție standard a limbajului C care realizează afișarea liberă sau, conform unui format specificat, a textelor introduse ca literal și/sau a conținutului variabilelor sau rezultatului evaluării funcțiilor specificate ca argument, pe ieșirea standard.

Prin #include <stdio.h> se specifică faptul că la compilarea programului se vor include specificațiile date în fișierul <stdio.h>. Acest fișier conține declarații, efectuate în limbajul C, prin care se specifică anumite valori ale unor variabile care au rolul de a emula programul pe un anumit terminal sau sistem de calcul (calculator).

Deoarece la noi avem o răspîndire puternică a calculatoarelor din gama INDEPENDENT și CORAL vom da, în continuare, modul de rulare a unui program scris în limbajul C sub sistemele de operare MIX sau RSX.

Introducerea programului sursă se va efectua cu ajutorul editoarelor standard, EDI sau EDT, ale acestora.

Presupunem că numele fișierului va fi AFIȘEAZA.C.

În aceste condiții vom avea următoarea secvență de operații:

```
>XCC AFIȘEAZA.C ⇒ AFIȘEAZA.S
>; realizează compilarea programului
>XAS -l -d AFIȘEAZA ⇒ AFIȘEAZA.OBJ, AFIȘEAZA.LST
>; realizează asamblarea cu ștergerea fișierului .S(-d)
>; și generarea fișierului de listare .LST(-l)
>TKB AFIȘEAZA/CP=AFIȘEAZA, LB:[1,1]C/LB
>; ediția de legături
>RUN AFIȘEAZA
```

De CRACIUN ne-am luat rația de LIBERTATE!

Cum, cînd, ce C!

Pentru calculatoarele personale de tip IBM PC XT sau AT cu produsul DESMET se vor parcurge pașii:

1. Introducerea textului sursă al programului cu ajutorul editorului EDIT sau cu WordStar;

2. Compilarea programului cu comanda

```
C88 <nume_fișier> [<opțiune>]
```

unde

<nume_fișier> este numele fișierului sursă creat cu editorul de texte (extensia implicită este .C);

<opțiuni> pot fi precedate de "_" (influență UNIX) și sînt:

A - produce la ieșire un program sursă assembler în locul unui fișier obiect (cu extensia .A);

C - compilatorul va produce informații necesare programului de depanare (debugger);

M - produce un format obiect INTEL;

Dnume - numele discului în care sînt stocate componentele "GEN" și "ASM88";

Inume - numele directorului implicit din care vor fi încărcate fișierele citate în directivele #include <nume_fișier>;

Onume - specifică producerea unui fișier obiect cu numele "nume". Dacă "nume" nu conține și extensii atunci va primi implicit extensia .O;

Td - se admite utilizarea fișierelor temporare pentru compilator (fișiere de lucru în cazul programelor mari).

3. Asamblarea programelor de tip .A:

```
ASM88 <nume_fișier> [<opțiuni>]
```

unde

<nume_fișier> este numele fișierului care este supus analizei asamblorului;

<opțiuni>:

Lnume - produce un fișier listing cu numele "nume";

M - produce un fișier obiect format INTEL;

Onume - produce un fișier obiect cu numele "nume";

Td - utilizarea fișierelor temporare la asamblare;

Wnn - lungimea paginii de listing ($60 \leq nn \leq 132$).

Dacă programul obținut în fază 2 este de tip obiect (.O) atunci faza 3 nu mai este necesară.

4. Obținerea codului executabil:

```
BIND <nume_fișier1><nume_fișier2>... [<opțiuni>]
```

Această comandă solicită existența bibliotecii standard DESMET CSTDIO.S.

<nume_fișier> este numele fișierelor obiect care vor fi link-editate pentru a forma un program executabil;

<opțiuni>:

A - în lista de fișiere avem și fișiere obiect obținute prin asamblare;

C - crearea unui fișier de tip .CHK necesar activității de depanare (cu debuggerul D88);

Fnume - numele fișierelor destinate lui BIND pot fi introduse într-un fișier sursă specificat prin "nume";

Lnume - listarea discului (sau directorului) în care se află biblioteca CSTDIO.S;

Mn - este admisă acoperirea (overlay);

Onume - "nume"le sub care va fi construit programul executabil;

P[numel] - o listă sortată cu numele programelor de tip .S declarate "public" și

Cum, cînd, ce C!

"offset";
Shhh - dimensiunea stivei;
Vn - dimensiune acoperire (overlay);
_ (underscore) - suprimarea numelor care încep cu __ (underscore).

Pentru compilatorul MSC (Microsoft C) compilarea și link-editarea se va efectua conform menu-urilor acestuia.

Sub sistemul de operare UNIX se vor parcurge următorii pași:

1. Compilare:

a) faza 1:

MC1 [=stivă][>fișier_listă] fișier [<opțiuni>]

=stivă - dimensiunea zonei de recursivitate (implicit 2048 bytes);
fișier_listă - fișierul pe care se direcționează mesajele;
fișier - specificatorul fișierului sursă;
opțiuni - se dau cu litere mici precedate de -:
-a - abandonează optimizările pentru pointeri;
-b - forțează alinierea la byte;
-c - comentariile se prelucrează în stivă
-d - cauzează includerea începutului informațiilor în fișierul "quad" (numărul liniei inclus în fișierul quad);
-id - citește toate #include... din discul "d";
-od - crează fișierul de ieșire (quad) pe discul "d";
-x - schimbă clasa de memorare pentru declarațiile externe (external).

b) faza 2:

MC2 nume_fis [<opțiuni>]

opțiuni:

-od - crează fișierul de ieșire (.OBJ) pe discul "d";

nume_fis - fișierul de intrare (.Q).

2. Obținerea codului executabil:

LINK C.OBJ + nume_fis1.OBJ + ... + MC.LIB

Se obține un program executabil cu numele nume_fis1.EXE.

3. Executarea programului:

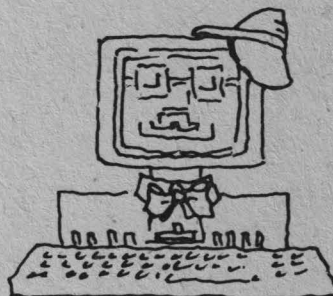
nume_prog [=stivă][<fiș.in>][<fis.ies>[<args>]

=stivă - toate obiectele declarate cu clasa "auto" vor fi alocate pe stivă;

<fis.in - numele fișierului care va substitui intrarea standard;

>fis.ies - fișierul pe care se va redirecționa ieșirea standard; fis.ies va fi deschis pentru adăugare;

<args> - sînt argumente destinate funcției main() (citate în program prin argc și argv) care vor fi transmise acesteia ca doi pointeri (argc dă numărul de elemente pe care le conține argv).



Baze de date relaționale și distribuite.
Programare, utilizare și analiză comparativă de sistem.
Limbajul SQL pentru pachetele ORACLE și RDB (I).

Designul și exploatarea foilor de calcul electronice (I)

De dată mai recentă, pachetele de programe din seria "foaie de calcul" - worksheet din engleză - permit obținerea rapidă și comodă, pornind de la datele inițiale ale unor rapoarte, rapoarte ușor de modificat ulterior în mod interactiv; pornind de la forma finală a raportului se obține, pe de o parte, forma portabilă a acestuia (fișier, listare la imprimantă) iar pe de altă parte se actualizează datele inițiale la noile valori.

Sigur, este greu să definești în puține cuvinte utilitatea unor pachete de programe a căror documentație conține sute de pagini. De aceea vom trece la o scurtă trecere în revistă a acestora la sfârșitul cărora orice cititor va avea o primă imagine asupra utilității și utilizării acestor tipuri de pachete de programare.

Vom începe cu pachetul de programe 1-2-3, produs al firmei Lotus Development Corporation, larg răspândit pe calculatoarele personale.

Foaia de calcul

Foaia de calcul este o matrice 8192x256. Liniile sînt numerotate de la 1 la 8192; coloanele sînt marcate cu litere de la A-Z apoi AA-AZ, BA-BZ, ..., IA-IV.

Elementele matricei se numesc celule; ele sînt unic identificate prin litera coloanei în care se află urmată de numărul liniei (de exemplu A1 pentru elementul din poziția 1,1; IV 8192 pentru elementul din poziția 8192,256). Fiecare celulă poate conține informația care poate fi modificată interactiv. La un moment dat utilizatorul nu poate selecta decît o singură celulă în care se află cursorul și care se numește celula curentă. Conținutul celulei curente poate fi modificat interactiv.

Foaia de calcul se poate găsi în patru moduri diferite: READY, POINT, MENU, HELP pe care le vom detalia pe parcurs.

Deasupra foii de calcul se găsesc trei linii cu informații care alcătuiesc împreună un panou de control.

Prima linie a panoului conține informații despre celula curentă incluzînd adresa celulei, conținutul celulei și, la cerere, prin opțiunile selectate: lățimea coloanei, formatul acesteia, protecția aferentă.

A doua linie afișează intrarea curentă la introducerea sau editarea datei din celula curentă. La testare în modul READY se trece în modul MENU apărînd, pe această linie, opțiunile posibile.

A treia linie afișează, fie sub menu-ul selectat opțiunile din acesta (pentru modulul MENU), fie o scurtă descriere a opțiunii selectate din menu.

Indicatorul de mod (la partea dreaptă sus a panoului) indică starea în care se găsește 1-2-3 la momentul respectiv.

Indicatorul de stare (la partea inferioară dreaptă a ecranului) indică informații referitoare la anumite taste sau particularități ale unui program.

Indicatorul dată și oră (la partea inferioară stîngă a ecranului) indică data și ora.

Deplasarea în foaia de calcul

Cursorul poate fi deplasat cu o celulă în foaia de calcul cu tastele UP, DOWN, LEFT, RIGHT sau cu un ecran cu tastele PAGE UP, PAGE DOWN, BIG RIGHT, BIG LEFT. Cu tasta HOME cursorul poate fi mutat în celula A1. Utilizând combinații ale tastei END cu tastele HOME, UP, DOWN, RIGHT, LEFT cursorul poate fi mutat în colțul drept al zonei ocupate cu date din foaie, respectiv ultima poziție ocupată pe direcția specificată.

Introducerea și editarea datelor de la tastatură

În celula selectată de cursor se pot introduce și/sau edita date de orice fel de la tastatură. Data introdusă sau editată (modificată) apare în celulă după tastare RETURN sau apăsarea pe o tastă de deplasare a cursorului.

Fiecare caracter tastat apare pe o a doua linie a panoului.

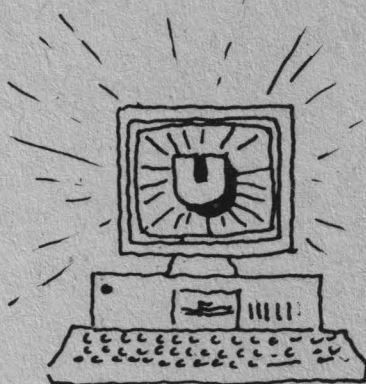
La validarea datei, 1-2-3 efectuează acțiunile:

- verifică existența erorilor; dacă există se emite un semnal sonor, se deplasează cursorul în celula respectivă și se trece automat în modul EDIT;
- dacă nu este găsită nici o eroare, data se introduce în celulă curentă; data anterioară dispare și nu mai poate fi recuperată;
- în funcție de setarea aleasă, 1-2-3 recalculează toate formulele din foaie (se va detalia ulterior);
- 1-2-3 trece în modul READY.

Data introdusă poate fi editată pe măsura tastării ei; de asemenea, poate fi editată o dată anterior introdusă. Pentru a edita o dată se selectează modul EDIT prin tastarea tastei corespunzătoare. Se reamintește că la încercarea de validare a unei date eronate, trecerea în modul EDIT o face automat 1-2-3.

Pentru a edita o dată se procedează astfel:

- pentru o dată deja introdusă se selectează ca celulă curentă celula care o conține. Pentru o dată în curs de introducere se pornește cu pasul următor;
- se apasă tasta EDIT. Indicatorul de mod trece în modul EDIT. Utilizând tastele direcționale cursorul se poziționează corespunzător;
- se inserează sau se șterg caractere în poziția selectată de cursor;
- pentru validare se apasă tasta RETURN.



Baze de date distribuite (I)

1. Definiție și obiective

1.1. Motivație

Mediul economico-social este într-un continuu proces de schimbare. Acest proces se materializează și prin apariția de noi întreprinderi (firme, corporații, etc.), dispariția altora, diviziunea unora sau reunirea în conglomerate cu piese componente de diverse orientări și structuri organizatorico-funcționale.

Indiferent care este modul lor de evoluție (exceptând dispariția "fizică") ele cooperează tot timpul printr-un masiv flux informațional.

Spațiul geografic nu mai reprezintă un impediment al acestor restructurări iar accesibilitatea tehnicii de calcul, ca suport al fluxului informațional, reprezintă o problemă de opțiune.

Obținerea unei structuri eterogene a unei întreprinderi impune o politică economică globală de acțiune și în acest context un cadru informațional, ca suport al deciziei, formal. În acest cadru nu este impusă o politică economică rigidă în sensul că decizia locală este luată în contextul existenței unui flux informațional local. Nucleul modern al oricărui flux informațional este reprezentat de una sau mai multe baze de date, pe care le vom numi locale (la nivel de element decizional al structurii).

Inițial politica de informatizare a unei întreprinderi, distribuită geografic sau nu, propunea construirea unor baze de date centralizate (pe domenii de activitate sau integrate) la care, prin diverse canale (teleprelucrare, rețele de calculatoare, etc.), aveau acces, diferențiat, toate structurile întreprinderii.

Această soluție reprezintă pe o parte un factor de risc, dat de o rată înaltă a centralizării, iar pe de altă parte o investiție costisitoare și incertă. Incertitudinea este dată de procesul rapid al schimbării structurilor. (În general, o bază de date, pentru a fi eficientă ca investiție, trebuie să funcționeze circa 10 ani).

Diseminarea informaticii prin explozia microcalculatoarelor și necesitatea unei cât mai bune modelări și adaptabilități la viața reală a dus la apariția bazelor de date distribuite.

În suita de articole, pe această temă, vom prezenta caracteristicile și arhitectura bazelor de date distribuite precum și modalitățile de realizare a proiectării lor. La sfârșitul expunerii vom prezenta două studii de caz privind proiectarea bazelor de date distribuite.

1.2. Definiție

O bază de date distribuită (BDD) este o bază de date (BD), logic integrată, dar fizic distribuită pe mai multe sisteme de calcul distincte, interconectate între ele.

Logic integrată: din punct de vedere al utilizatorului BDD reprezintă o singură BD. El interacționează cu o BDD în același mod cum interacționa cu o bază de date centralizată. Utilizatorul nu trebuie să cunoască modul de partiționare, replicare și distribuire a datelor. Singurul lucru pe care îl cunoaște este schema conceptuală a bazei de date, schemă denumită schemă globală. Prelucrările inițiate

la un nod pe care se află partiții (părți) ale bazei de date antrenează, în general, prelucrarea informațiilor aflate în alt nod.

De exemplu, bazele de date locale BDL1, BDL2 și BDL3 stocate în nodurile unei rețele notate (corespondent) N1, N2 și, respectiv, N3 se integrează în baza de date distribuită prin intermediul unei scheme globale (SG) la care va face referire orice utilizator global.

Un **utilizator local** reprezintă un utilizator (aparținând oricăruia dintre tipurile de utilizatori ai bazelor de date) care are acces și exploatează o bază locală. El cunoaște și manipulează numai schema aceleiași baze de date locale (sau o subschemă a acesteia). Manipularea bazei de date locale include oricare din bazele de date din rețea (chiar dacă se efectuează cu terminal virtual utilizând numai facilitățile puse la dispoziție de software-ul de rețea).

Un **utilizator global** aparține oricăruia din tipurile de utilizatori ai bazelor de date cu singura diferență că el cunoaște și are acces la schema (sau o subschemă globală a) bazei de date globale. Utilizatorul global exploatează schema globală, conform autorizărilor și drepturilor sale de acces, de aceeași manieră în care ar lucra cu o bază de date locală (sau centralizată).

Fizic distribuită pe mai multe sisteme de calcul distincte: baza de date este partiționată iar partițiile respective sînt pe calculatoare diferite (se admit copii ale fragmentelor memorate în noduri diferite).

Fiecare fragment este văzut, în nodul în care există, ca o bază de date centralizată, care poate fi exploatată și administrată local.

Baza de date distribuită (o mulțime de colecții de date și legăturile dintre ele) este fragmentată conform principiilor:

- plasarea datelor memorate în nodul de producere și utilizare frecventă a lor;

- minimizarea transportului de date prin rețeaua de calculatoare.

Pentru a răspunde acestor principii fragmentarea se realizează la două nivele:

- partiționarea mulțimii de colecții în submulțimi de colecții de date;

- partiționarea unei colecții de date în fragmente.

Partiționarea unei colecții de date poate fi realizată în două moduri:

- **orizontal** - fragmentele obținute au aceeași schemă conceptuală (structură) ca și colecția, dar diferă între ele prin datele pe care le conțin;

- **vertical** - fragmentele (fiecare în parte) conțin doar o parte din schema conceptuală a colecției.

Este admisă și combinarea acestor două moduri.

Fragmentele rezultate constituie elementele de distribuire a datelor. Totalitatea fragmentelor unei baze de date distribuite, memorate pe un nod al rețelei formează o bază de date locală.

1.3. Obiective

În concordanță cu definiția un sistem de baze de date distribuite trebuie să asigure cel puțin următoarele obiective:

- 1° **transmiterea datelor la utilizatorii lor:** indiferent care este nodul în care se află un utilizator și indiferent unde sînt stocate datele, acesta trebuie să-și poată obține informațiile de care are nevoie (cu condiția să aibă dreptul și autoritatea de a le accesa);

- 2° **evitarea unei foarte înalte centralizări a resurselor,** centralizare care cauzează foarte mult eficienței și eficacității sistemului: o rată înaltă a

centralizării cauzează un cost ridicat de prelucrare și transmitere a datelor la utilizatori;

3° să sporească durabilitatea sistemului: pot fi introduse în orice moment structuri de baze de date (pot fi integrate noi baze de date cu scheme conceptuale diferite de oricare din cele existente) prin includerea schemei lor în schema conceptuală globală fără a afecta aplicațiile existente;

4° să facă mai ușoare operațiile de menținere și restructurarea a bazei de date cu menținerea unei rate înalte a disponibilității: deoarece cea ce vede un utilizator global reprezintă o schemă globală, care are drept corespondent diverse scheme locale, operația de restructurare a unei scheme locale nu afectează cu nimic utilizatorul global;

5° să permită proiectarea structurii organizatorice și funcționale a sistemului analizat: prin faptul că, în general, bazele de date locale se află în locurile în care se produc informațiile pe care le conțin, o arhitectură distribuită permite o emulare mai puternică, a sistemului informațional, conformă structurii organizatorice și funcționale;

6° să mărească gradul de utilizare a sistemului: prin emularea cadrului organizatoric și funcțional și prin disponibilitatea datelor se obține creșterea numărului de utilizatori ai sistemului.

Realizarea acestor obiective atrage după sine considerarea și rezolvarea unor probleme tehnice ca:

- **posibilitatea accesului la BDD privită ca sistem integrat:** baza de date distribuită trebuie să permită să fie văzută de utilizator ca o bază de date centralizată;

- **asigurarea transparenței alocării fizice a datelor față de utilizator:** exceptând utilizatorul (utilizatorii) special (i), administratorul bazei de date, ceilalți utilizatori nu trebuie să cunoască locul în care sînt alocate datele pentru a-și formula întrebările adresate bazei de date;

- **portabilitatea software-ului:** deoarece o bază de date distribuită poate avea în componență baze de date locale gestionate de diverse tipuri de calculatoare (chiar și diverse tipuri de sisteme de gestiune a bazelor de date și sisteme de operare) este necesară asigurarea portabilității software-ului de gestiune a bazei de date distribuite;

- **asigurarea unui sistem eficient de catalogare:** programele de aplicație trebuie să fie disponibile în toate nodurile rețelei pentru a realiza o exploatare eficientă a aplicațiilor globale. De asemenea este posibil ca prin definirea unor subsisteme globale, alocate diverselor grupuri de utilizatori, să fie necesară construirea unor programe de asigurare a corespondenței schemei locale cu schema globală, programe a căror prezență este obligatorie în toate nodurile rețelei;

- **asigurarea independenței logice, fizice și distributive a datelor:** independența logică și fizică trebuie asigurată similar ca bazele de date centralizate. Independența distributivă se referă la faptul că dacă schimbăm nodul în care este stocată o partiție a bazei de date distribuite acest lucru nu trebuie să influențeze aplicațiile și schema conceptuală globală.

Față de bazele de date centralizate bazele de date distribuite ridică probleme noi ca:

- prevenirea creșterii redunanței sau a inconsistenței datelor la dezvoltarea de noi aplicații;

- definirea unor noi instrucțiuni pentru standarde de definire și utilizare a datelor;

- administrarea eficientă a cererilor;
- utilizarea eficientă a resurselor de telecomunicație (rețelei de calculatoare); etc.

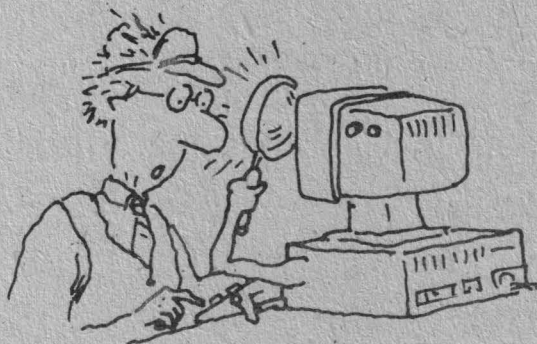
1.4. Bază de date relațională distribuită

Definiția bazei de date distribuită se adaptează modelului relațional astfel:

"O bază de date relațională distribuită (BDDR) constă dintr-o colecție de tabele (relații) fiecare din ele putând fi stocată într-un singur nod al rețelei sau poate fi răspândită într-o rețea de calculatoare. Fiecare relație distribuită poate fi fragmentată orizontal sau vertical în acord cu un criteriu de distribuire (în general un predicat de distribuire)".

O tabelă (relație) este locală dacă este stocată în întregime într-un singur nod și globală dacă fragmentele sale sînt stocate în diverse noduri.

Avantajul utilizării bazelor de date distribuite este dat de faptul că sistemul de gestiune a bazelor de date distribuite (software-ul care gestionează baza de date distribuită) funcționează ca un sistem centralizat în timp ce fizic se adaptează repartiției componentelor unei întreprinderi sau administrări.



**Teme pentru informaticieni.
Prelucrarea listelor în C și PASCAL.**

Prelucrarea listelor în C și PASCAL (I)

În cele ce urmează vom analiza posibilitatea mînuirii structurilor de date dinamice în Limbajele C și Pascal, date care, spre deosebire de variabilele statice (de tip vector sau matrice, declarate în secțiunile de date și alocate într-un spațiu de memorie destinat) cer rutine speciale de gestiune a memoriei care este alocată sau eliberată după nevoie. În general variabilele dinamice nu au nume, referirea lor făcîndu-se indirect prin intermediul unor variabile de tip pointer. Să reamintim că un pointer are drept valoare adresa unei alte variabile, statice sau dinamice, funcție de utilizarea sa. Avem deci nevoie de definirea unei structuri de date cu un membru care să fie o referință de tip pointer la o structură de același tip cu cea definită primar, permițînd astfel înlănțuirea unui număr nespecificat de structuri de același tip. Vom numi un astfel de lanț de structuri înlănțuite listă de date, fiecare element al listei avînd deci aceeași formă declarată. Definim o listă dacă definim forma elementelor sale. Astfel, definirea unei liste formată din numere întregi presupune definirea fiecărui element astfel:

În C:

```
struct listă {
    int    număr;
    struct listă *adresa_următorului_element;
} LIS;
```

În Pascal aceeași structură admite definirea:

```
Type listă = ^element
      element = record
          număr: int;
          adresa_următorului_element: listă
      end;
```

Var Lis: listă;

Nu insistăm asupra tipurilor de declarație a pointerilor sau a înregistrărilor (structure, record) în C și Pascal, acestea fiind referite pe larg în cursuri de programare. Obiectivul nostru este studiul paralel a posibilităților de manipulare și utilizare a listelor ca structuri de date dinamice în aceste limbaje. Declarația structurii <listă> poate fi acomodată în două cuvinte de memorie, unul pentru întregul <număr> și celălalt pentru legătura de tip pointer <adresa_următorului_element>. Această variabilă de legătură între elementele listei, <adresa_următorului_element>, conține adresa de memorie a locației elementului imediat următor din listă ori valorile speciale NULL (în C) sau NIL (în Pascal) ce marchează sfîrșitul listei. Să definim acum două liste, fiecare formată dintr-un singur element după care să încercăm înlănțuirea lor:

- În C:

```
struct listă L1, L2;
```

- Atribuirea de valori o facem prin:

```
L1.număr = 1989;
L2.număr = 1990;
```

Prelucrarea listelor în C și PASCAL

L1.adresa_următorului_element = L2.adresa_următorului_element = NULL;

- Înlănțuirea celor două liste ne dă:

L1.adresa_următorului_element = &L2

- schematic avem:

```

L1                L2
-----
| 1989 | .----|---->| 1990 | NULL |
-----
```

- tehnicile de găsire a anumitor valori în listă sînt:

L1.număr - are valoarea 1989

în timp ce:

L2.adresa_următorului_element -> număr are valoarea 1990.

În Pascal:

- declarația celor 2 liste este:

```
var L1,L2 : listă;
```

```
begin
```

```
  L1↑.număr = 1989;
```

```
  L2↑.număr = 1990;
```

```
L1↑.adresa_următorului_element=L2↑.adresa_următorului_element:=nil;
```

```
end.
```

- legătura celor două liste se face prin:

```
L1↑.adresa_următorului_element: = L2;
```

Să trecem în continuare la definirea unei liste ca o înșiruire secvențială de elemente de același tip, cu un prim element adresat de un pointer de început (sau cap de listă), fiecare alt element conținând adresa succesivului său, cu ultimul element avînd adresa de legătură valoarea NULL(C) sau NIL(Pascal).

Definirea unei astfel de liste în C are forma:

```
#define NULL 0
typedef int ÎNTREG /*definim o listă de întregi*/
struct lista {
  ÎNTREG număr;
  lista *următorul /*definesc *următorul ca fiind adresa elementului
                    succesiv din listă */
};
typedef struct lista ENTITATE;
typedef ENTITATE *LIS;
```

Alocarea spațiului pentru definirea de mai sus, se poate face dinamic în funcție de nevoia de elemente noi în listă.

Limbajul C prin funcția <malloc()>, apelată prin:

```
malloc (spațiu)
```

întoarce un pointer pentru un obiect cu un număr de <spațiu> octeți (parametrul <spațiu> fiind de tip întreg). Astfel:

```
cap_listă= (Lis)malloc(sizeof(ENTITATE));
```

acordă pointerului <cap_listă> adresa zonei de memorie la care se află un element al listei.

Să creăm în continuare o listă care să conțină multimea {1,7,3}:

```
cap_listă = (Lis)malloc(sizeof(ENTITATE));
```

```
cap_listă -> număr = 1;
```



```
cap_listă -> următorul = (Lis)malloc(sizeof(ENTITATE));
cap_listă -> următorul -> număr = 7;
cap_listă -> următorul -> următorul=(Lis malloc(sizeof(ENTITATE)));
cap_listă -> următorul -> următorul->număr = 3;
cap_listă -> următorul -> următorul -> următorul = NULL;
```

Evident se poate defini o funcție care să execute recursiv pașii de mai sus, generînd un modul sursă mai compact și mai ușor de manevrat. Este ceea ce vom realiza în Pascal. Să scriem deci codul sursă de mai sus beneficiind de facilitățile limbajului Pascal care conține instrucțiunea <new> pentru alocarea de spațiu unui nou element al listei.

Să definim astfel o procedură pentru crearea și înserarea de valori într-o listă:

```
procedure cit_listă(var cap_listă:Lis);
var element_nou : Lis;
begin
    new(element_nou);
    with element_nou↑ do
        begin
            readln(număr);
            următorul:=cap_listă
        end;
    cap_listă:= element_nou /*modific poziția din vârful listei*/
end; /*procedură*/
```

Pornind în programul principal cu cap_listă:=nil, apelul cit_listă (cap_listă) permite introducerea de elemente în listă pînă la o condiție de oprire semnalată de utilizator în program.

Putem procesa și operația de scriere a elementelor unei liste create prin procedura de mai sus, impunînd condiția de oprire la întîlnirea ultimului element cu adresa de legătură NIL.

```
procedure scriu_listă(var cap_listă: Lis);
var element_nou:Lis;
begin
    element_nou:=cap_listă;
    while element_nou< >nil do
        with element_nou↑ do
            begin
                writeln(număr);
                element_nou:=următorul;
            end;
        end; /*procedura*/
```

Recursivitatea și iterația joacă un rol important în prelucrarea datelor. Să prezentăm astfel, în limbajul C, două rutine de generare a unei liste dintr-un șir de întregi, una realizată prin recursie, alta prin iterație. Amîndouă funcțiile generate întorc un pointer în cadrul listei rezultate:

```
- rutina prin recursie
Lis șir_în_listă (l)
integer l[];
```

Prelucrarea listelor în C și PASCAL

```

    Lis cap_listă;
if (l[0] == '\0')
    return (NULL) /*sfârșit de listă*/
else {
    cap_listă = (Lis) malloc (sizeof(ENTITATE));
    cap_listă -> număr = l[0];
    cap_listă -> următorul = șir_în_listă (l+1);
    return (cap_listă);
}
}

```

Următoarea rutină folosește generarea listei prin punerea în evidență a capului de listă și a cozii sale (restul elementelor) procedeu utilizat ca strategie de bază în programarea logică.

- rutina prin iterație (l)

Lis șir_în_listă (l)

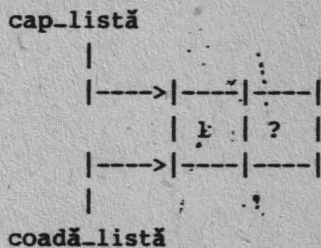
```

integer l;
{
    Lis cap_listă = NULL, coadă_listă;
    int k;
    if (l[0] != '\0' ){
        cap_listă = (Lis) malloc (sizeof(ENTITATE));
        cap_listă -> număr = l[0];
        coadă_listă = cap_listă;
        for (k = 1; l[k]!='\0'; ++k){
            coadă_listă -> următorul=(Lis)malloc(sizeof(ENTITATE));
            coadă_listă = coadă_listă -> următorul;
            coadă_listă -> număr = l[k];
        }
        coadă_listă -> următorul = NULL;
    }
    return (cap_listă);
}
}

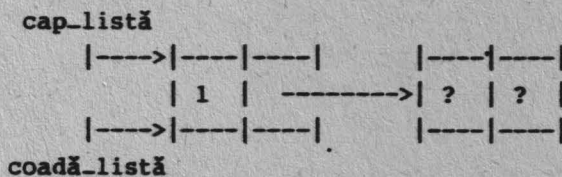
```

Grafic, să creăm lista [1,2] din șirul l[0]=1; l[1]=2; l[2]='\0' (terminatorul de șir).

Astfel, cap_listă -> număr = l[0] arată:



Bucloa "for" executată cu i = 1, crează un nou element:




```
while L↑NIL do
  with L↑ do
    begin
      writeln(număr);
      L:=următorul
    end;
  end.
```

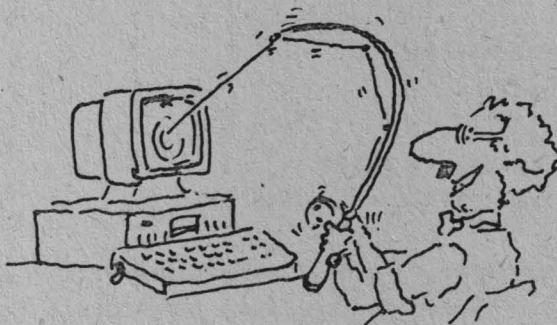
3. Concatenarea a două liste:

În C:

```
concatenez (L1,L2)
Lis L1,L2;
{
  if (L1->următorul ==NULL)
    L1->următorul =L2;
  else
    concatenez (L1->următorul, L2);
}
```

În Pascal:

```
function concatenez (L1,L2:Lis);
begin
  if (L1↑.următorul ==NIL)then
    L1↑.următorul:=L2
  else
    concatenez (L1↑.următorul, L2);
end.
```



Inedit, recenzii, noutăți, istoric în informatică

Din istoria calculatoarelor personale (I)

Era sistemelor cu divizarea timpului s-a încheiat. Sistemele cu divizarea timpului au luat naștere din necesitatea de a pune, în același timp, capacitatea unui calculator conversațional la dispoziția mai multor utilizatori, aceasta atunci când aceste calculatoare erau prea scumpe ca să fie utilizate de un singur om.

Dezvoltarea tehnologiei hardware a deschis noi căi. Nivelul investițiilor, care în zilele noastre face accesibil pentru fiecare utilizator cuantele din sistemul de divizare a timpului prin terminal cu tub catodic, va asigura la mijlocul anilor optzeci, pentru aceiași utilizatori, mașini proprii mai performante decât microprocesoarele zilelor noastre, cu instrumente de I/E grafice și sonore. Dezvoltarea va asigura ocolirea compromisurilor și limitelor impuse de sistemele cu divizarea timpului. Tehnologiile de rețea noi, de viteză mare vor păstra avantajele sistemelor cu divizare a timpului: informații distribuite, comunicații între utilizatori, partajarea perifericelor scumpe. Vom păși în lumea calculatoarelor personale multilaterale și lumea aceasta diferă de lumea sistemelor cu divizarea timpului așa cum și sistemele cu divizare a timpului diferă de prelucrările batch.

Referințe: CMU SPICE Comitee: Proposal for a joint Effort in Personal Scientific Computing, Tehnical Report, Carnegie - Melon University, 1979.

Calculatorul personal este prima unealtă individuală din istorie care permite să se mărească vizibil eficiența muncii intelectuale. Distanța, pe scara productivității profesionale, între oameni apropiați ca posibilități, dintre care unul este înarmat cu mijloace corespunzătoare iar altul este neînarmat, se va mări, evident, rapid. Prin aceasta se și explică ritmurile ridicate de creștere a vânzărilor de calculatoare personale din ultimii 10 ani. În 1976 s-au vândut 20000 de calculatoare personale. În 1982 numărul de calculatoare personale instalate atingea cca 5 milioane iar către 1984 - 10 milioane.

O evaluare indirectă, dar foarte convingătoare, a eficacității deosebite a calculatoarelor personale în activitatea profesională a milioane de oameni care lucrează în sfera informațională a economiei naționale trebuie să se considere faptul următor: circa jumătate din întregul tiraj al calculatoarelor personale este cumpărată de profesioniști de nivel mediu și superior ai mecanismului economic din SUA din banii proprii "obținuți prin sudoare". Pentru acest contingent de cumpărători prețul calculatorului personal constituie aproximativ salariul pe o lună. În 1990 prețul unui calculator s-a redus la 1/3 dintr-un salariu lunar.

Alfabetizarea computerizată în masă și tirajele de multe milioane în producția de calculatoare personale - mașinile unelte ale celei de-a doua revoluții industriale - imprimă un curs puternic dezvoltării forțelor de producție. Întârzierea în dezvoltarea industriei calculatoarelor personale - acolo unde ea va exista - va necesita, deja la sfârșitul deceniului nostru, eforturi costisitoare în masă pentru a preîntâmpina numai cele mai evidente urmări legate de influența calculatoarelor personale asupra ritmurilor de creștere a

Din istoria calculatoarelor personale

productivității muncii în sfera informațională a economiei naționale. Iar, în ansamblu, apariția în viața noastră a fenomenului calculatorului personal poate provoca probleme care sînt comparative ca amploare cu mișcarea cunoscută din istorie pentru lichidarea analfabetismului.

Referințe: G. Gromov: ZNANIE - SILA (URSS), 1985.

Cu răspîndirea calculatoarelor din generația a treia a pornit și un fenomen de alienare. Între programatori și mașini s-a pus un zid din ce în ce mai înalt și din ce în ce mai gros. Programarea și-a pierdut un caracter - acum deja știm - foarte important. S-a pierdut caracterul personal al programării. Atunci nu era evident dar astăzi știm că programe cu adevărat bune nu se nasc decît în contactul direct și nestingherit dintre omul creator și mașină.

.....

Evoluția a închis un ciclu în spirală, cu calculatoarele personale ne întorcem la un nivel mai înalt la utilizarea, în apropierea omului, a mașinilor din generația unu și doi. Mașinile vechi se utilizau în mod "personal" din cauza capacității lor reduse, în schimb microcalculatoare pot fi aduse în apropierea omului datorită prețului lor scăzut.

Cîteva date din istoria calculatorului personal

- 1959 - Anul de naștere al microelectronicii, primul circuit integrat (TEXAS INSTRUMENTS).
- 1971 - Primul microprocesor cu 4 biți, 4004-Intel.
- 1974 - Microprocesorul cu 8 biți Intel 8080.
- 1975 - Microprocesorul 6502 la prețul de 25 \$.
Cluburi de construire a microcalculatoarelor.
Apariția revistei pentru microcalculatoare Byte.
- 1976 - Primul calculator personal Apple II.
- 1978 - Pachetele Wordstar, Visicalc, dBase II.
- 1981 - Apariția calculatorului profesional IBM PC.
- 1983 - Pachetul integrat Lotus 1-2-3.
- 1984 - Microprocesoarele din familia Intel 80816, 80286 pe 16 biți și Motorola 36000.
- 1985 - Baza de date Oracle cu limbajul de manipulare SQL.
Sistemele AutoCAD, dBase III, IV, cu compilatorul CLIPPBOARD, pachete de birotică Smart, Framework, limbaje evaluate C, Pascal, Assembler.
- 1986 - Apariția tehnicilor în inteligența artificială Microlisp, Prolog, procesoare de limbaj natural.
- 1987 - Explozia tehnologică în arhitectura hardware-calculatoare cu discuri hard și Streamer de 30, 60 MB.
- 1988 - Apariția sistemului de operare OS/2 pentru lucrul multitasking pe microcalculatoare, rețele de comunicație micro-micro, micro-mainframe.
- 1989 - Discuri de masă rigide de 100, 200 MB precum și tehnologia de procesare paralelă.
Introducerea discurilor optice read-write cu capacitate peste 1 MO.
- 1990 - Consolidarea și dezvoltarea noilor tendințe astfel:
 1. Utilizarea de noi tipuri de microprocesoare care permit modul de lucru protejat (80286, 80386 și 80486);
 2. Mărirea capacității memoriei interne operative (16MO - 4GO);

Din istoria calculatoarelor personale

3. Creșterea vitezei de prelucrare;
4. Mărirea vitezei de transfer la operațiile de I/O prin creșterea volumului de date transferate (5MO/S - 33MO/S);
5. Mărirea capacității de stocare a discului hard (1,2 GO);
6. Extinderea posibilităților de lucru în mod grafic;
7. Oferirea unor mecanisme care să permită integrarea facilă a micro în rețele de calculatoare eterogene.

Construcția micro este orientată conform uneia din noile standarde arhitecturale care s-au impus și anume:

a) ISA (Industry Standard Architecture) propusă și dezvoltată de IBM la construirea lui PS/2. Prin acest standard debitul pentru operațiile de I/O este de 5MO/S iar memoria activă gestionată este de 16MO. Placa de bază a calculatorului integrează controlerul grafic de tip VGA iar cuplarea în rețea se poate realiza în arhitecturi de tip SNA;

b) EISA (Extended Industry Standard Architecture) propusă și dezvoltată de Compaq. Prin acest standard debitul pentru operațiile de I/O este de 33MO/S iar dimensiunea memoriei active gestionate este de 4GO.

Din punct de vedere software se degajă următoarele tendințe:

1. Oferirea unor mecanisme de utilizare a memoriei virtuale;
2. Exploatarea modului de lucru protejat al noilor tipuri de microprocesoare pentru a permite utilizarea micro în regim de multiprogramare, multitasking și multiconsola;
3. Link-editare dinamică a programelor;
4. Mărirea dimensiunii programelor (de la 64KO la 16MO la care se poate adăuga încă 1GO dacă se utilizează memoria virtuală);
5. Integrarea în rețele de calculatoare eterogene;
6. Încorporarea unui SGBD relațional în software-ul de bază (limbajul utilizat de SGBD este SQL).

Categoriile de calculatoare personale

În cei 10 ani de la apariția lor, calculatoarele personale s-au diversificat atât din punct de vedere al capacității cât și al utilizării:

- calculatoare familiale;
- calculatoare semiprofesionale;
- calculatoare profesionale;
- calculatoare personale de capacitate mare;
- calculatoare personale portabile.

Calculatorul familial (home computer)

Sînt calculatoare construite pe un microprocesor de 8 biți (Intel 8080, Z80, 6502), au o memorie dinamică de pînă la 48 Kocteți. De obicei se livrează în forma unei tastaturi care încorporează atât unitatea centrală cât și circuitele auxiliare. Perifericele acestei categorii de calculatoare sînt aparate electronice care se găsesc în jurul casei: televizorul, cu rolul de afișare, și casetofonul (eventual magnetofonul) pentru înregistrarea programelor și a datelor.

Tastatura, de obicei, este tastatură de tip folie.

Din istoria calculatoarelor personale

Au generator sonor (adesea pînă la 5 octave).

Interpretorul BASIC este "ars" în memorie.

Se utilizează în mare parte la jocuri (inclusiv șah). Poate fi programat în BASIC, astfel poate executa calcule cu un volum de date mai redus (ex.: bugetul familial).

Reprezentative sînt calculatoarele firmelor Sinclair și Commodore:

Sinclair ZX - 81 (2K) 12000 Fr (1985)

1490 OS (1984)

ZX - Spectrum (48K) 20000 Fr (1986)

5990 OS (1984)

Commodore - 16 300 DM (1986)

În Ungaria: ABC - 80, PRIMO, Aircomp - 16, HO 1080 Z.

Programe pentru calculatoare familiale:

- jocuri 300-900 Fr (1985)

- programe de șah 40-70 DM (1985)

- diverse interpretoare (MPROLOG, FORTH etc.).

Calculatoare personale semiprofesionale (sau calculatoare CP/M)

Sînt calculatoare realizate cu microprocesoare de 8 biți. Ceea ce le deosebește, ca arhitectură, de calculatoarele familiale este existența perifericelor mai pretențioase (unitățile de disc FLOPPY, imprimante, eventual cuplor de teletransmisie). Arhitectura rămîne totuși închisă, de tip "folosește-l și aruncă-l" (Shrow-away).

Modelele reprezentative sînt:

Commodore - 64:

- apărut în 1982;
- realizat cu microprocesor Z80;
- 50000 Fr (1986).

Apple II, Apple IIe, Apple IIc:

- realizat cu microprocesorul 6502;
- Apple IIe (64K, imprimantă, FLOPPY): 41140 OS.

Alte modele: TRS - 80, Atari, HP - 80.

În Ungaria: IANUS, LABSYS, VARYTER, M08X, PROPER - 8, COMPUT - 80, TZ80, Minicomp - C - 1, Transmic 8.

Aceste calculatoare nu mai sînt mașini BASIC ci au sistem de operare cel mai răspîndit fiind sistemul de operare CP/M. Aplicațiile generice cele mai uzuale: VISICALC, WORDSTAR, dBASE etc.

Aplicațiile concrete pentru C64 (întocmit pe baza unor anunțuri din MIKROSZAMITOGEP MAGAZIN):

- evidența personalului (1000 de persoane cu cîte 19 date fiecare);
- evidența comenzilor și a stocurilor;
- repartizarea costurilor;
- evidența mijloacelor fixe;
- evidența mijloacelor de transport (pentru 20 de unități și 300 de mașini);
- calculul salariilor (pentru 850 muncitori);
- evidența materialelor;
- facturare.

Din istoria calculatoarelor personale

Calculatoare personale profesionale

Caracteristici:

- sînt realizate cu microprocesoare de 16 sau 32 biți;
- au memorie dinamică pînă la 1 M;
- unitățile de disc Winchester (pînă la 2x20 Mbiți);
- arhitectura deschisă: se pot atașa noi plachete, configurația se poate dezvolta ulterior.

Modele reprezentative:

IBM PC:

- apărut în 1981;
 - microprocesor de 16 biți (Intel 8086);
 - o configurație cu display, tastatură, imprimantă, Winchester de 10MB, soft de bază, FLOPPY:
 - 2,1 milioane Fr;
 - 5866 \$.
- (1985)

Tipuri de IBM PC:

	Memorie internă	Floppy	Winchester (disc de masă)
IBM PC	256K	2x320K	--
IBM PC XT	512K	1x360K	10M
IBM PC AT	1M	1x1,2M	20M

MacIntosh (firma Apple)

- apărut în 1984;
- microprocesor de 32 biți (MOTOROLA);
- utilizare cu "icoane" și "șoricel" (mouse) pentru comanda cursorului;
- soft de bază (128+64K, FLOPPY, soft de bază):
63655 OS (1984).

Situația vânzărilor în domeniul calculatoarelor profesionale (1984):

IBM	35%
Apple	12%
IBM + compatibile IBM	aprox. 50%

Apariția modelelor IBM PC, XT, AT pe piață a produs o avalanșă atît în creația software-ului pentru aceste modele cît și fabricarea unor modele "compatibil IBM PC".

Modele (compatibile IBM PC):

- HP 150 II:
 - memorie pînă la 640K;
 - Winchester de 4,8 sau 15M;
 - configurația de bază (256K + FLOPPY) 65500 OS (1984)
 - Winchester de 15M 125000 OS (1984)
 - software WordStar 9771 OS (1984)
 - " dBase II 11237 OS (1984)
 - " Cobol 14667 OS (1984)

În Ungaria: PROPER - 16;
HT 6800 X;
VT - 16 (Videofon).

Aplicații: birotică, gestiune, calcule științifice, CAD, CAM etc.

Din istoria calculatoarelor personale

Aplicații în gestiunea economică (pe baza unor anunțuri din MIKROSZAMTIOGEP MAGAZIN):

- gestiunea stocurilor;
- evidența conturilor;
- cartea mare a întreprinderii;
- evidența materialelor;
- repartizarea costurilor;
- evidența mijloacelor fixe;
- calculul impozitelor;
- evidența comenzilor;
- calculul salariilor.

Calculatoare personale portabile

Sînt realizate foarte compact, de obicei cu afișare cu cristale lichide. Sînt echipate cu model acustic pentru o cuplare ușoară la rețeaua telefonică (cuplarea = formarea numărului + așezarea receptorului pe modem).

Modelele mai vechi sînt mai voluminoase (gen valiză):

- OSBORNE;
- IBM PC;
- COMPAQ.

Modelele mai noi sînt realizate în cutii tip geantă diplomat:

- TRS 100 (Tandy);
- HP 110;
- Epson Geneva.

Calculatoare personale de capacitate mare

Caracteristicile acestui tip de calculator au fost formulate în 1979 la Carnegie-Mellon University în raportul: Proposal; for a Joint Effort in personal Scientific Computing. Acest raport a fost destinat pentru stimularea fabricanților de calculatoare ca aceștia să dezvolte sisteme comercializabile (în jur de 10000 \$) și care să satisfacă criteriile impuse în raport. Această mașină fictivă se numește SPICE (Scientific Personal Integrated Computing Environment).

Din specificația hardware:

- 1000000 instrucțiuni/s;
- posibilitatea microprogramării, posibilitatea emulării oricărui tip de microprocesor;
- adresare virtuală pînă la 2^{30} . . . 2^{32} octeți;
- memorie centrală: minimum 1M;
- memorie externă: 100M;
- ecranul grafic, color;
- cuplor de rețea (10M biți/s).

Din specificația software:

- calcule științifice și tehnice, sisteme de proiectare;
- prelucrare de texte, redactare de documente (text + grafică), verificarea ortografiei;
- comunicații și servicii de rețea (gestiunea de bază de date, imprimare

Din istoria calculatoarelor personale

centralizată, corespondență).

Protocolul de comunicație de tip Xerox și ARPA-INTERNET.

Cercetarea legată de calculatoare personale de capacitate mare a luat amploare în S.U.A. pe la mijlocul anilor '70 și sistemele sînt prezente pe piață din 1980. În acest domeniu centrul de cercetare din Palo Alto a firmei Xerox deține un rol conducător. A apărut cu principii de sistem noi: limbaj de programare de nivel înalt sprijinit de o arhitectură adecvată, rețea locală de viteză mare, "șoricelul" (mouse), icoanele, principiul ferestrei în prelucrările de documente etc.

Modele:

- Alto (1973) - Xerox;
- Dorado (1977) - Xerox;
- Domain (Distributed Operating Multi - Acces Interactive Network) - Apollo;
- MicroVAX - Digital Equipment Corporation;
- Symbolics 3600 - mașina LISP.

Software și aplicații pentru microcalculatoare profesionale și semiprofesionale

În Ungaria este un obicei ca un institut, în primul rînd, să cumpere sau să închirieze un calculator și, abia după aceea, să se întrebese la ce l-ar putea folosi. Cîțiva angajați sînt instruiți să opereze calculatorul și să programeze în BASIC și se așteaptă ca el să scrie programele necesare.

Acesta este drumul eșecului. Nu aduce decît dezamăgire, reîntoarcerea la metodele tradiționale și descoperirea informaticii.

Ordinea corectă este următoarea:

- a decide care sînt problemele parțial sau integral programabile;
- trebuie însărcinat un specialist în marketing pentru a găsi programul sau pachetul de programe adecvat pentru scopul propus;
- angajații sînt instruiți în programarea structurată, independentă de mașină;
- în același timp, cu un specialist din afară, se alege calculatorul adecvat pentru pachetul de programe;
- se cumpără pachetul de programe și mașina;
- specialiștii, instruiți în programare structurată, vor învăța utilizarea mașinii și a pachetului de programe.

Pentru prima dată, în 1983 vânzările de software pentru microcalculatoare au depășit un miliard de dolari.

În milioane de \$

	În milioane de \$
1. IBM	110
2. Tandy (Radio Shack)	110
3. Apple	68
4. Microsoft (MS - DOS, Xenix)	68
5. Visicorp (Visicalc)	52
6. Micropro (WordStar)	50
7. Digital Research (CP/M)	44
8. Lotus Development (1-2-3)	38

Din istoria calculatoarelor personale

Produsele software pentru microcalculatoare s-au dezvoltat în trei direcții principale. Ascensiunea comercială a celor trei direcții corespunde, în mare, ordinii de enumerare:

1) Produse pentru dezvoltare software și sisteme de operare (limbaje de programare și utilitare).

CP/M sistem de operare pentru microcalculatoare de 8 biți.

MS - DOS și PC - DOS sisteme de operare pentru microcalculatoare IBM - PC și compatibile cu IBM - PC.

Sisteme de operare pentru microcalculatoare cu mai mulți utilizatori: UNIX, Xenix, OS/2.

Interpretoare și compilatoare: BASIC, FORTH, PASCAL, C, LISP, microPROLOG.

2) Aplicații cu utilitate directă:

- pachete de programe pentru rezolvarea uneia sau unor grupe de funcții în practica comercială și economică (facturare, calcul salarii, înregistrarea comenzilor, urmărirea conturilor etc.);

- pachete de programe pentru satisfacerea necesităților de calcul și arhivare dintr-un anumit domeniu (asigurări, vânzări de locuințe, industria hotelieră, probleme de notariat, vânzări de ziare), desene animate, planificarea instalației de încălzire centrală pentru o locuință, șah, educație, proiectarea circuitelor, proiectarea plachetelor imprimante).

3) Produse pentru dezvoltare de aplicații, aplicații generice:

- pachete pentru gestiunea bazelor de date;

- "electronic spreadsheet" pentru luarea deciziilor asistată de calculator;

- grafică economică;

- pachete pentru prelucrarea documentelor (word processor);

- pachete pentru comunicații între microcalculatoare și între calculatoare personale și calculatoare mari;

- pachete integrate pentru luarea deciziilor asistate de calculator: reunesc într-un singur pachet parțial sau integral funcțiile descrise mai sus.

Implementarea APLICAȚIEI CONCRETE nu se face prin programare clasică ci prin definirea problemei în termenii Generalizați ai aplicației generice.

Pachete pentru gestiunea bazelor de date

Au apărut pe piață în formă de generatoare de aplicații. Reunesc următoarele funcții:

- definiția bazei de date;

- interogarea bazei de date;

- introducerea de date;

- actualizarea datelor din baza de date;

- generarea rapoartelor.

În limbajele de comandă de tip interactiv sau vizual.

Deoarece, cu aceste generatoare de aplicații, chiar și aplicațiile complicate și mari pot fi implementate într-un timp de zece ori mai scurt și deoarece în aceste implementări nu trebuie utilizați programatori și analiști, aceste sisteme deschid o nouă direcție de utilizare a microcalculatoarelor.

Pachetul tipic este dBASE II și varianta lui îmbunătățită dBASE III sau

Din istoria calculatoarelor personale

IV. Se utilizează sub sistemul de operare CP/M pe calculatoarele cu 8 biți și sub "S-DOS pe cele de 16 biți.

Alte pachete: Personal Pearl, Condor, Visifill, Infostar, Knowledge Man.

O altă categorie de generatoare de aplicații:

Oracle (firma Oracle) și Focus (firma Information Builders).

Acestea pot fi utilizate în rețele de calculatoare personale și minicalculatoare pentru că sînt implementate pe ambele categorii de calculatoare. Un exemplu: conducerea la fața locului a unor activități de construcții printr-o aplicație realizată în Oracle pe calculator IBM PC într-o rețea coordonată de o aplicație centrală scrisă tot în Oracle pe un megamini VAX.

Prețul unui pachet de tip dBASE aproximativ 500 \$ (1986).

Foaie de calcul pentru luarea deciziilor asistată de calculator

Pachetele tipice: Visicalc (1979), Supercalc, Visitrend.

Permite gestionarea unor tabele economice, comerciale cu relații de calcul între anumite coloane sau între anumite linii și efectuarea mai multor iterații - pe un asemenea tabel - pentru aflarea optimumului.

Foaia de calcul se formează în memoria microcalculatorului. Dacă foaia este mare pe ecran se vede doar o porțiune a sa însă ecranul poate fi "plimbat" foarte comod pe întreaga foaie. Se definesc relațiile dintre celulele tabelului și poate începe completarea datelor. Simultan cu introducerea sau modificarea datelor sînt calculate sau recalculat toate datele din celulele ale căror valori se obțin prin relații de calcul.

Aplicații: vânzări, rabat comercial, prognoză comercială, în planificare la aplicații de genul "ce se întîmplă dacă", evidență etc.

Visicalc: a apărut în 1979. Pînă în 1985 s-au vîndut 850 mii buc., utilizabil sub CP/M și pe Apple.

În 1984 pe 66% din microcalculatoarele utilizate în întreprinderi se lucra cu foaie de calcul electronică.

Pachete pentru prelucrarea documentelor

Documentul este un ansamblu redactat din texte și ilustrații. Pachetele reunesc parțial sau integral următoarele funcții:

a) introducerea și prelucrarea textelor (introducerea textului, despărțirea automată în silabe, deplasarea unor cuvinte în rîndul următor, aranjarea în pagină, verificarea ortografică, realizarea cuprinsului, a indexului, a unor note etc.). Pachete tipice: WordStar, Microsoft Word, Multimate, Macwrite (Apple), Easy Script, Tex.

b) realizarea graficelor. Pachete tipice: Visiplot, MacPAINT (Apple), Ventura, Autocad.

În aplicații economice pot realiza pe bază de tabele, grafice economice.

c) redactarea documentelor (plasarea graficelor în documente, rearanjarea părților din document înainte de tipărire).

Pachete de comunicații

Calculatoarele personale profesionale pot fi utilizate ca stații de lucru în rețele sau ca terminale conversaționale.

Pentru aceasta sînt dotate cu cuploare (interfețe) de transmisie și emulatoare de terminal (software). Emulatorul cel mai răspîndit este emulatorul de terminal IBM 3270.

Pachete integrate pentru luarea deciziilor asistate de calculator

Cea mai tipică aplicație a calculatorului în birotică: pe baza unor date extrase dintr-o bază de date se întocmește un raport, cu ajutorul unui program de prelucrare de texte, iar în acest raport anumite părți sînt redade sub formă de tabele sau grafice.

Pentru asemenea prelucrări există pachete de programe integrate. Cel mai popular și cu un succes comercial remarcabil este LOTUS 1-2-3.

Denumirea de 1-2-3 arată că s-a reușit reunirea a trei funcții într-o singură aplicație generică:

- foaie de calcul electronică 265x2048 celule;
- gestiunea bazei de date;
- grafică comercială.

LOTUS 1-2-3 a apărut în octombrie 1982 și pînă în vara anului 1985 s-au vîndut 1 milion de bucăți, neluînd în considerare copiile ilegale.

Pachetul SYMPHONY (tot firma LOTUS 1984) adaugă încă două funcții la posibilitățile lui 1-2-3 și anume: prelucrarea de texte și comunicații de date.

Alte pachete: Frame work (Ashton-Tate 1984), Golden-Gate (Cullinet 1984), Smart.

Pachetele integrate pot fi utilizate numai pe calculatoare profesionale (16 biți) IBM PC sau compatibile și MacIntosh (Apple) fiind reprezentative.

Rețele locale

Aplicațiile pe microcalculatoare, care pînă acum funcționau izolate, se transformă în sisteme informatice distribuite, reunite prin rețele de comunicație.

Însuși calculatorul personal se transformă în stație personală de calcul.

Sistemul informatic distribuit, reunit prin rețea de comunicații, este un mediu care diferă de cel precedent bazat pe hîrtie, prin dinamism:

- dinamism la nivelul stației de lucru la înregistrarea, prelucrarea, afișarea informațiilor (să ne gîndim la desenarea unei figuri sau la modificarea lui cu pachet de grafică);

- dinamism în schimbul de informații între stațiile de lucru;

- dinamism în "comportamentul" sistemului informatic (utilizatorul este cel care definește sau redefinește sistemul de decizii, baza de date, relațiile dintre punctele de lucru etc.).

Î asemenea rețea (denumită și rețea locală - local area network) se compune din:

- stații de lucru constituite din calculatoare personale;

- puncte de activități specifice pentru descrierea rețelei, cu activități

Din istoria calculatoarelor personale

care sînt partajabile între utilizatori:

- imprimare (printer server);
- comunicare (communication server);
- baza de date (file server).

Aceste puncte sînt deservite de calculatoare specializate.
- rețeaua de comunicații.

Din "Proposal for a joint Effort in Personal Scientific Computing".

Servicii de rețea

Într-un proiect mare, de obicei, iau parte mai mulți. Colegii lucrează în clădiri diferite, pe probleme descompuse, pe calculatorul lor personal. Din cînd în cînd trebuie să se consulte despre problemele apărute. Oricare din ei poate iniția apelul unuia sau a mai multor colegi, prin rețea. Cei chemați pot accepta sau pot refuza apelul. Teleconferința se desfășoară prin periferice de I/E de sunet iar transmisia prin rețea compactat, digitizat (realizat: voice switching - Xerox - 1984). În cursul discuției anumite dialoguri pot fi înregistrate într-o bază proprie pentru a fi reascultate mai tîrziu. La terminarea conferinței chiar întregul text poate fi expedit, pe baza unei liste, celor interesați.

Există și cerința ca lucrătorii să definească lista temelor și persoanelor de care se lasă "deranjați" în timpul unei activități urgente și importante (de ex.: de secretară, cafeaua este gata). Orice alt apel este reținut și poate fi rechemat mai tîrziu. Desigur, toate acestea pot fi realizate și de o secretară bună dar o secretară bună este rară, scumpă și este disponibilă doar în timpul programului.

Gestionarea apelurilor telefonice

Funcționarul este departe de locul de muncă dar așteaptă un apel telefonic important. În acest caz, o mașină dedicată acestui scop primește apelul, anunță că apelatul lipsește și solicită numele și adresa apelantului. Un program de recunoaștere a vorbirii poate înscrie apelul pe lista celor care trebuie rechemați.

Servicii de informare

A sosit un coleg nou în institut. Cînd se anunță de prima dată în sistem acesta îi solicită informații biografice și cu un SPICE dotat cu o cameră de luat vederi este înregistrat portretul lui. În ziua următoare sistemul prezintă noul coleg celorlalți colegi și afișează portretul lui pe ecran. Din această clipă oricine poate să se intereseze, de ex., de John Smith (în baza de date pot fi regăsite portretul și datele biografice).

Software și sisteme de operare

Pînă la începutul anului 1988 sistemele de operare implementate pe marea majoritate a microcalculatoarelor personale au fost MS-DOS, PC-DOS, Xenix, Unix orientate pe microprocesoarele Intel 8086-8088 sau Motorola 68000 ce alcătuiau "inima" oricărui PC. Odată cu apariția microprocesoarelor 80286 și 80386 din familia Intel ce beneficiază de modul protejat s-a simțit nevoia dezvoltării unui nou sistem de operare care, în același timp, să poată elimina și limitările fizice și logice ale celor vechi.

Astfel, firma IBM anunță, în februarie 1988, lansarea pe piață a sistemului de operare OS/2 implementabil pe calculatoarele personale ce încorporează procesoarele 80286 și 80386. Ca și MS-DOS-ul, sistemul OS/2 este de tip "single-user" fiind livrat pe modelele 50, 60 și 80 ale microcalculatoarelor IBM PS/2 (personal system/2) precum și pe cele din familia IBM PC AT și IBM PC XT ce sînt prevăzute cu un procesor 80286.

Pentru a putea fi unanim acceptat, sistemul OS/2 beneficiază de o multitudine de particularități, dintre care:

- 1) Poate adresa o memorie fizică de pînă la 16M bytes;
- 2) Permite execuția de aplicații concurente;
- 3) Asigură protecția între aplicații simultane independente;
- 4) Permite multitasking;
- 5) Permite comunicarea și sincronizarea între programe;
- 6) Beneficiază de o interfață de aplicație programabilă (API);
- 7) Compatibilitatea cu aplicațiile scrise sub MS-DOS sau PC-DOS;
- 8) Permite scrierea și asigurarea la sistem a driver-erelor utilizator;
- 9) Conține un program de instalare și dezvoltare de sistem.

Odată cu creșterea puterii sistemelor de calculatoare personale gestionarea unei baze de date precum și dezvoltarea complexității structurilor impune utilizarea unor limbaje de interogare și organizare din ce în ce mai complexe. Din această cauză, în iunie 1988, sistemul OS/2 a fost dezvoltat rezultatul fiind un nou sistem denumit OS/2 Extended Edition ce conține o puternică bază de date relațională precum și un administrator de sistem avînd la bază limbajul SQL (Structured Query Language).

Întreaga bază de date a sistemului OS/2 EE, pe lîngă limbajul SQL, este organizată pe modelul relațional inventat de E.F. Codd la Centrul de cercetări din San Jose. Interesant este aici și posibilitatea execuției sub OS/2 a mai multor procese concurente permițînd accesul la aceeași bază din procese OS/2 distincte, cu păstrarea consistenței lor, asigurînd și suportul complet al tranzacției (orice citire sau scriere în bază este făcută în cadrul unei tranzacții).

De asemenea, remarcabilă este realizarea, în cadrul sistemelor OS/2, a programelor suport pentru interfața Multipat/2 ce se constituie ca submicro-calculator permițînd comunicarea full-duplex în sisteme ce suportă diverse protocoale, sincron sau asincron, în timp real. Cea mai recentă componentă a sistemului OS/2 este pachetul de tipărire complexă (AFP) ce permite mixtarea textului, imaginilor și graficelor în procesul de afișare sau tipărire.

Informatică în învățămîntul liceal și universitar.

Ghid de probleme rezolvate pentru elevi și studenți

Această secțiune își propune să prezinte soluțiile sub formă de programe sursă a unei mulțimi de probleme diverse ce apar în activitatea curentă de programare. Programele, scrise în diverse limbaje de programare moderne, caută să răspundă atît cerințelor didactice în privința învățării strategiilor de formalizare algoritmică cît și dilemelor curente ce intervin în scrierea aplicațiilor de către analiști consacrați. Sub motivația enunțurilor, programele prezentate fac uz de modele și proceduri care, fără îndoială, cercetate în detaliu, oferă cheia unor algoritmi cu o largă întrebuințare în probleme curente de programare.

1. Să se calculeze cel mai mic multiplu comun a două numere x și y , folosind descompunerea lor în factori primi.

R: Programul scris în Turbo Pascal (v. 3) citește numerele x, y de la tastatură și, în urma descompunerii celor două numere în factori primi, calculează cmmmc prin produsul factorilor primi comuni și necomuni la puterea cea mai mare. Se scrie la imprimantă atît descompunerea celor două numere cît și cmmmmc.

```
program cmmmmc_prin_calculul_divizorilor_primi;
label contoar1,contoar2,ciclu;
var lst:test;
type
  tablou=array[1..100,1..2] of integer;
  var x,y,s,i,j,k,max,mc,maxdiv1,maxdiv2,t:integer;
  var a,b,z:tablou;
  function p(a,b:integer):integer;
  var i,s:integer;
begin
  s:=1;
  for i:=1 to b do
    s:=s*a;
    p:=s;
end;
function prim(x:integer):boolean;
var d:integer;
begin
  d:=2;
  while (d<=trunc(sqrt(x))) and (x mod d<>0) do
    if d=2 then
      d:=3
    else
      d:=d+2;
    prim:=d>trunc(sqrt(x));
  end;
begin
  assign(lst,'lpt1');rewrite(lst);
```

Ghid de probleme rezolvate pentru elevi și studenți

```

k:=1;
write(lst,"Introduceți numărul x,x= ");
readln(x);
writeln(lst,x);
write(lst,"Introduceți numărul y,y= ");
readln(y);
writeln(lst,y);
for i:=1 to x do
begin
  if ((x mod i=0) and prim(i)) then
  begin
    a[k,1]:=1;
    k:=k+1;
  end
end;
maxdiv1:=k-1;
k:=1;
for i:=1 to y do
begin
  if ((y mod i=0) and prim(i)) then
  begin
    b[k,1]:=i;
    k:=k+1;
  end
end;
maxdiv2:=k-1;
write(lst,"Divizorii primi ai lui x la puterea 1-sînt:");
for i:=1 to maxdiv1 do
writeln(lst,'a[' ,i,' ,1]=' ,a[i,1]);
writeln(lst,"Divizorii primi ai lui y la puterea 1 sînt:");
for i:=1 to maxdiv2 do
writeln(lst,'b[' ,i,' ,1]=' ,b[i,1]);
a[1,2]:=1;
for i:=2 to maxdiv1 do
begin
  j:=1;
  s:=a[i,1];
contoarl: t:=x mod s;
  if (t=0) then
  begin
    s:=s*a[i,1];
    j:=j+1;
    go to contoarl;
  end
  else
    a[i,2]:=j-1
  end;
  writeln(lst,"Puterile la care apar divizorii 'lui x sînt:");
  for i:=1 to maxdiv1 do
    writeln(lst,a[i,1],'la puterea ',a[i,2]);
  b[1,2]:=1;

```

Ghid de probleme rezolvate pentru elevi și studenți

```

for i:=2 to maxdiv2 do
begin
j:=1;
s:=b[i,1];
contoar2: t:y mod s;
if (t=0) then
begin
s:=s*b[i,1];
j:=j+1;
go to contoar2;
end
else
b[i,2]:=j-1;
end;
writeln(lst,"Puterile la care apar divizorii lui y sînt:");
for i:=1 to maxdiv2 do
writeln(lst,b[i,1],'la puterea ',b[i,2]);
for i:=1 to maxdiv1 do
begin
z[i,1]:=a[i,1];
z[i,2]:=a[i,2];
end;
k:=maxdiv1+1;
for i:=1 to maxdiv2 do
begin
j:=1;
ciclu: if(z[j,1]<>b[i,1]) then
begin
if(j=maxdiv1) then
begin
z[k,1]:=b[i,1];
z[k,2]:=b[i,2];
k:=k+1;
end
else
begin
j:=j+1;
go to ciclu;
end;
end
else
begin
if(z[j,2]<b[i,2]) then
z[j,2]:=b[i,2];
end
end
max:=k-1;
for i:=1 to max do
writeln(lst,'z[' ,i ,',1]=' ,z[i,1] ,;z[' ,i ,',2]=' ,z[i,2]);
mc:=1;
for i:=1 to max do

```

```

mc:=mc*p(z[i,1],z[i,2]);
writeln(lst,'C.m.m.m.c. al numerelor x=',x,',y=',y,'este mic',mc)
end.
Introduceți numărul x,x=36
Introduceți numărul y,y=21
Divizorii primi ai lui x la puterea 1 sînt:
a[1,1]=1
a[2,1]=2
a[3,1]=3
Divizorii primi ai lui y la puterea 1 sînt:
b[1,1]=1
b[2,1]=3
b[3,1]=7
Puterile la care apar divizorii lui x sînt:
1 la puterea 1
2 la puterea 2
3 la puterea 3
Puterile la care apar divizorii lui y sînt:
1 la puterea 1
3 la puterea 1
7 la puterea 1
z[1,1]=1;z[1,2]=2
z[2,1]=2;z[2,2]=2
z[3,1]=3;z[3,2]=2
C.m.m.m.c. al numerelor x=36;y=21 este mc=252

```

2. Să se calculeze recurent o primitivă a funcției:

$$f(t) = \frac{1}{\sin^n(t)}$$

pe intervalul $[0, x]$.

R: Programul scris în Turbo Pascal (v. 3) realizează calculul recurent folosind facilitățile subrutinelor recursive de tip <function> cu scrierea la imprimantă a calculului efectuat pentru valoarea specificată a lui n și a lui x.

```

program integrală_recurentă;
var lst:text;
var n:integer;x:real;
function cp(n:integer;x:real):real;
begin
if n=0 then
cp:=1
else
cp:=cos(x)*cp(n-1,x);
end;
function tg(x:real):real;
begin
tg:=sin(x)/cos(x);
end;
function int(n:integer;x:real):real;
begin
if n=1 then
int:=ln(tg(x/2+pi/4))

```

```

else
  begin
    if n=2 then
      int:=tg(x)
    else
      int:=((n-2)/(n-1))*int(n-2,x)+(1/(n-1))*sin(x)/cp(n-1,x);
    end;
  end;
begin
  Assign(lst,'lpt1');rewrite(lst);
  write("introduceți n=");
  write(lst,"introduceți n=");
  readln(n);
  writeln(lst,n);
  write("introduceți x=");
  write("introduceți x=");
  readln(x);
  writeln(lst,x);
  writeln(lst,'valoarea integralei este I(',n,')=',int(n,x));
  writeln('I(',n,')=',int"n,x));
  writeln(lst);
end.

```

3. Să se determine elementul minimal al unui șir de caractere, media aritmetică a valorilor din șir și pozițiile elementelor din șir a căror valoare este mai mare decât valoarea primului element al șirului.

R: Programul de mai jos, scris în Turbo Pascal, rezolvă problema propusă scriind la imprimantă rezultatele cerute.

```

program vector;
var lst:text;
type
matrice=array[1..2,1..3] of integer;
vector=array[1..10] of integer;
var s,p:vector;
    m:matrice;
    i,j,k,min,mm,mp,pivot:integer;
    Ma:real;
begin
assign(lst,'lpt1');rewrite(lst);
for i:=1 to 10 do
begin
write(lst,'s[' ,i,']=');
readln(s[i]);
writeln(lst,ls[i]);
end;
for i:=1 to 10 do
begin
write('s[' ,i,']=',s[i]);
writeln;
end;
min:=s[1];

```

Ghid de probleme rezolvate pentru elevi și studenți

```

for i:=2 to 10 do
begin
    if (s[i]≤min) then
        min:=s[i]
    end;
writeln(lst,'minimul este=',min);
Ma:=0;
for i:=1 to 10 do
ma:=ma + s[i];
ma:=ma/10;
writeln(lst,'Ma=',ma);
pivot:=-s[1];
k:=1;
    for i:=2 to 10 do
    begin
        if (s[i]>pivot) then
begin
p[k]:=i;
            k:=k+1;
            end
        end;
        mp:=k-1;
        writeln(lst,"Pozițiile din șir mai mari ca primul termen sînt:");
        for i:=1 to mp do
            writeln(lst,',' ,p[i]);
        writeln(lst,"Valorile mai mari ca pivotul sînt:");
        for I:=1 to mp do
            writeln(lst,'s[' ,p[i] ,']=s[p[i]]);
end.
s[1]=5
s[2]=2
s[3]=3
s[4]=1
s[5]=1
s[6]=0
s[7]=7
s[8]=9
s[9]=8
s[10]=9
minimul este=0
Ma=4.5000000000E+00
Pozițiile din șir mai mari ca primul termen sînt:
;7
;8
;9
;10
Valorile mai mari ca pivotul sînt:
s[7]=7
s[8]=9
s[9]=8
s[10]=9

```

Ghid de probleme rezolvate pentru elevi și studenți

4. Să se scrie, în Basic, un program care să utilizeze fișiere random.

R: a) Un prim program este:

```
10 OPEN "R", #1, "A:DATE", 33
20 FIELD #1, 15 AS N$, 6 AS T$, 6 AS R$, 6 AS C$
30 INPUT "NR.ÎNREG."; COD%: IF COD% > 100 THEN GO TO 30
40 IF COD% = 0 THEN CLOSE 1: END
50 GET #1, COD%: PRINT N$; T$; R$; C$
60 INPUT "NUME:"; X$
70 INPUT "VAL1:"; Y$
80 INPUT "VAL2:"; Z$
90 L = LEN(X$): IF L <> 0 THEN LSET N$ = X$
100 L = LEN(Y$): IF L <> 0 THEN LSET T$ = Y$
110 L = LEN(Z$): IF L <> 0 THEN LSET R$ = Z$
120 T1 = VAL(T$): R1 = VAL(R$)
130 R2 = T1: IF R1 > 0 THEN R2 = R1
140 IF R2 < 1000 THEN C1 = C2 * 0.3 / 100
150 IF R2 > 999 THEN C1 = R2 * 0.7 / 100: F% = C1
160 C2$ = STR$(F%): LSET C$ = C2$
170 PUT #1, COD%: PRINT N$; T$; R$; C$
180 GO TO 30
190 END
```

Înregistrarea fișierului este formată din patru câmpuri:

- N\$ (15 caractere) nume;
- T\$ (6 caractere) valoarea1;
- R\$ (6 caractere) valoare2;
- C\$ (6 caractere) valoare3.

Câmpurile nume, valoarea1 și valoare2 pot fi actualizate. Tastatura lui Enter lasă câmpul neschimbat.

Câmpul valoare3 se calculează folosind câmpul valoarea1 sau câmpul valoare2 (când acesta este strict pozitiv).

Se pot crea maxim 100 înregistrări.

La sfârșitul prelucrării se tastează 0 pentru NR.ÎNREG.

b) O altă variantă de utilizare a unui fișier random este:

```
10 OPEN "R", #1, "A:DATE", 33
20 FIELD #1, 15 AS N$, 6 AS T$, 6 AS R$, 6 AS C$
30 TOTAL = 0: INPUT "NR.TOTAL ÎNREGISTRARI:"; NRP: IF NRP > 100 THEN GO TO 30
40 FOR COD% = 1 TO NRP
50 GET #1, COD%
60 V = VAL(C$): TOTAL = TOTAL + V
70 PRINT TAB(1); COD%; TAB(6); N$; TAB(23); T$; TAB(32); R$; TAB(41); V
80 IF COD% = 20 OR COD% = 40 OR COD% = 60 OR COD% = 80 THEN STOP
90 NEXT COD%
100 PRINT "TOTAL:"; TOTAL: CLOSE 1
110 END
```

Înregistrarea fișierului este formată din patru câmpuri:

- N\$ (15 caractere) nume;
- T\$ (6 caractere) valoarea1;
- R\$ (6 caractere) valoare2;
- C\$ (6 caractere) valoare3.

Pentru un număr de înregistrări indicat (≤ 100) se face totalul pentru

valorile din câmpul de valoare3 și se afișează înregistrările în grupe de câte 20.

c) În continuare prezentăm un program ce utilizează două fișiere random.

```

10 DIM A$(20),B(4),C(4)
20 OPEN "R", #1,"DATE1"
30 OPEN "R", #2,"DATE2"
40 FIELD #1,10 AS A$(1),10 AS A$(2)
50 FIELD #2,10 AS A$(3),10 AS A$(4)
60 FOR I=1 TO 4
70 B(I)=I:X$=STR$(B(I)):LSET A$(I)=X$
80 NEXT I
90 PUT #1,1
100 PUT #2,1
110 GET #1,1
120 GET #2,1
130 FOR I=1 TO 4
140 C(I)=VAL(A$(I)):PRINT C(I);
150 NEXT I
160 CLOSE 1:CLOSE 2
    
```

În program intervin două fișiere random. În descrierea câmpurilor sînt folosite componente vectoriale.

5. Să se scrie un program care să rezolve metode numerice de calcul al integralelor prin metodele trapezelor, dreptunghiului și Simson.

R: Programul următor, scris în GW-Basic, rezolvă problema propusă.

```

2 REM PROGRAM DE CALCUL A INTEGRALELOR CU METODA
3 REM DREPTUNGIURILOR,TRAPEZELOR ȘI SIMSON
4 REM
5 REM
6 REM
7 DIM X(100),S(100)
9 DEFINT I-N
10 DEF FNF(X);X*X
20 DEF FNF1(X)=2*X
100 REM RUTINA DE DEFINIRE A FUNCȚIEI
101 REM ȘI A DERIVATEI
102 CLS
110 PRINT "AȚI DEFINIT FUNCȚIA? <D/N>"
120 INPUT Q$
130 IF Q$="D" THEN GO TO 1000
140 PRINT "TASTAȚI LA TERMINAL LINIA:"
150 PRINT "10 DEF FNF(X)=<FUNCȚIA DORITA>"
160 PRINT "CALCULAȚI VA ROG DERIVATA FUNCȚIEI"
170 PRINT "TASTAȚI APOI LINIA:"
180 PRINT "20 DEF FNF1(X)=<FUNCȚIA DERIVATA>"
190 PRINT "CEREȚI ACUM EXECUȚIA PROGRAMULUI"
200 STOP
1000 REM RUTINA DE DEFINIRE CONDIȚII
1001 CLS
1002 INPUT "AȚI DEFINIT INTERVALUL ȘI DIVZIUNILE? <D/N>";V$
1003 IF V$="D" THEN GO TO 1050
    
```



```

1010 INPUT "CAPAT STÎNGA INTERVAL";A
1020 INPUT "CAPAT DREAPTA INTERVAL";B
1030 INPUT "NUMAR DE DIVIZIUNI";N
1050 LET X(1)=A
1060 LET E=0
1061 LET N1=N+1
1070 FOR I=1 TO N1
1080 LET X(I)=A+(I-1)*(B-A)/N:NEXT I
1082 CLS
1090 PRINT "ALEGEȚI METODA DE CALCUL A INTEGRALEI DINTRE METODELE:"
1100 PRINT "TRAPEZ(T),DREPTUNGHI(D),SIMSON(S)"
1110 LET R=FNF1(X(1))
1120 INPUT "T,D,S";C$
1130 IF C$="T" THEN GO TO 2000
1140 IF C$="S" THEN GO TO 3000
1150 REM METODA DREPTUNGHIULUI
1151 COLOR 1,7
1152 LOCATE 10,10:PRINT "METODA DREPTUNGHIULUI"
1160 FOR I=2 TO N1
1161 LET Z=(X(I)+X(I-1))/2
1170 LET S(I)=FNF(Z)
1171 LET E=E+S(I)
1172 NEXT I
1173 LET P=E*(B-A)/N
1175 FOR I=2 TO N1
1176 IF R>=FNF1(X(I)) THEN GO TO 1178
1177 LET R=FNF1(X(I))
1178 NEXT I
1179 LET SU=(B-A)*R/(4*N)
1180 CLS:LOCATE 3,3:COLOR 3,6
1190 PRINT "VALOAREA INTEGRALEI PE INTERVALUL |";A;B;"| ESTE:"
1200 PRINT "|(F(X))=";P
1210 PRINT "EROAREA DE CALCUL ESTE <=";SU
1230 INPUT "DORIȚI SA CONTINUĂȚI? <D/N>";C$
1240 IF C$="D" THEN GO TO 1000
1250 STOP
2000 REM METODA TRAPEZELOR
2010 CLS
2020 PRINT "METODA TRAPEZELOR"
2021 LET E=0
2030 FOR I=2 TO N
2031 LET E=E+FNF(X(I))
2040 NEXT I
2050 LET P=E+(FNF(X(1))+FNF(N1))/2
2060 LET P1=P*(B-A)/N
2070 PRINT "VALOAREA INTEGRALEI"
2080 PRINT "PE INTERVALUL |";A;" ";B;"| ESTE:"
2090 PRINT "|(F(X))=";P1
2100 INPUT "DORIȚI SA CUNOAȘTEȚI EROAREA? <D/N>";C$
2110 IF C$="D" THEN GO TO 2200

```

Ghid de probleme rezolvate pentru elevi și studenți

```

2120 INPUT "DORIȚI SA CONTINUAȚI? <D/N>";L$
2130 IF L$="D" THEN GO TO 1000
2140 STOP
2200 PRINT "CALCULAȚI VA ROG DERIVATA A DOUA"
2210 INPUT "DERIVATA A DOUA ESTE:";V$
2230 DEF FNF2(X)=V$
2240 LET M=FNF2(X(1))
2250 FOR I=2 TO N1
2260 IF M>=FNF2(X(I)) THEN GO TO 2280
2270 LET M=FNF2(X(I))
2280 NEXT I
2290 LET SU1=((B-A)*(B-A)*(B-A)*M)/(12*N*N)
2300 GO TO 1230
3000 REM METODA LUI SIMSON
3010 CLS
3020 PRINT "METODA LUI SIMSON"
3030 LET E1=FNF(X(I))+FNF(X(N1))
3031 LET E2=0
3032 FOR I=2 TO N
3033 LET E=E+2*FNF(X(I))
3034 LET E2=E2+4*FNF((X(I)+X(I-1))/2)
3035 NEXT I
3036 LET E2=E2+4*FNF(X(N1)+X(N))/2
3037 LET SI=(B-A)*(E1+E2+E)/(6*N)
3060 PRINT "VALOAREA INTEGRALEI"
3070 PRINT "PE INTERVALUL [";A;B;"] ESTE"
3080 PRINT "(F(X))=";SI
3090 INPUT G$
3100 IF G$="D" THEN GO TO 3200
3110 GO TO 1230
3200 PRINT "CALCULAȚI VA ROG DERIVATA A DOUA ȘI A TREIA"
3210 INPUT "RENUNȚAȚI? <D/N>";L$
3230 IF L$="D" THEN GO TO 3110
3240 INPUT "DERIVATA A TREIA ESTE:";Z$
3250 DEF FNF3(X)=Z$
3260 LET R=FNF3(X(1))
3270 FOR I=2 TO N1
3280 IF R>=FNF3(X(I)) THEN GO TO 3300
3290 LET R=FNF3(X(I))
3300 NEXT I
3310 LET SU2=((B-A)*5)*R/(64*N*N)
3320 PRINT "EROAREA DE CALCUL ESTE<=";SU2
3330 GO TO 1230

```

6. Să se scrie un program care concatenează fișierele specificate de utilizatori la un fișier specificat.

R: O soluție este prezentată în programul următor și este valabilă pentru fișiere secvențiale de tip text.

```

#include <stdio.h>
main()

```

```

{
FILE*fi,*fl; /*declararea pointerilor care vor fi asociați fișierelor
             implicate*/
int c;
static char rezfis[65]=""; /*zonă inițializată cu spații care va conține
                           numele fișierului de intrare*/

char file[65],cc;
fl=fopen(rezfis,"au"); /*deschiderea fișierului rezultat ca fișier
                       nedefinit ("u") și în adăugare ("a")*/
if(fl==NULL) /*fișierul de ieșire nu există?*/
{printf("\n Introduceți numele fișierului rezultat:");
scanf("%s",rezfis); /*citește numele fișierului*/
if(fl=fopen(rezfis,"au")==NULL)
{
printf("\n Nu pot să deschid...%s...abandon!",rezfis);
exit(-1);
}
}
while(file[0]!=EOF)
{ printf("\n Introduceți numele fișierului de concatenat:");
scanf("%s",file);
fi=fopen(file,"ru"); /*deschidere în citire*/
if(fi==NULL);
{
printf("\n Nu pot să deschid fișierul...%s...",
"încercați din nou",&file);
continue; /*ca un comentariu*/
}
while((c=getc(fi))!=EOF) /*citește un caracter*/
{
cc=c;
putc(cc,fl); /*scrie în fișier*/
}fclose(fi); /*închide fișierul de intrare*/
}
fclose(fl); /*închide fișierul de ieșire*/
printf("\n Operația s-a terminat O.K.!");
}

```

7. Să se scrie un program care transformă ecranul unui terminal PC în ecran grafic de rezoluție medie (320,200) puncte în care să se deseneze un disc format din razele sale.

R: Programul următor, scris în Turbo Pascal, veziunea 3, satisface cerințele problemei culorile razelor ce umplu discul fiind schimbate prin ultimul parametru (i+j) al directivei grafice <draw>.

```

program LINIE;
var
i,j:integer;
begin
graphmode;
Palette(14);

```

```

for i:=0 to 15 do
  for j:=0 to 9 do
    draw(20*i,20*j,319-20*i,199-20*j,i+j);
  write(chr(7)) {beep}
end.

```

8. Să se definească conceptual scheletul unui program expert.

R: Acest program este o machetă de sistem expert care interoghează utilizatorul în privința tipurilor de viețuitoare ce se pot încadra în anumite criterii variabile care, satisfăcute, dau sistemului posibilitatea de a-și îmbogăți baza de cunoștințe, inițial definită doar de operatori asociativi. (Sursa program este scrisă în Turbo Prolog.)

```

*/
database
  xpozitiv(symbol,symbol)
  xnegativ(symbol,symbol)
predicates
  run
  vietate_este(symbol)
  are_atribut(symbol)
  pozitiv(symbol,symbol)
  negativ(symbol,symbol)
  șterge_date_bază
  verific_istoric(symbol,symbol,symbol)
  întreb_sistem_util(symbol,symbol)
goal
  run.
clauses
  run:-
    vietate_este(X),!,
    write("\n Animalul poate fi o (un)",X),
    nl,nl,șterge_date_bază.
run:-
  write("\n Incapabil să determin; am nevoie să actualizez baza!"),
  write("Voi reveni.\n\n"),șterge_date_bază.
pozitiv(X,Y) if xpozitiv(X,Y),!.
pozitiv(X,Y) if not(negativ(X,Y)),! and întreb_sistem_util(X,Y).
negativ(X,Y) if xnegativ(X,Y),!.
întreb_sistem_util(X,Y):-
  write(X,"el (ea)",Y,"\n"),
  readln(Replica),
  verific_istoric(X,Y,Replica).
verif_istoric(X,Y,da):-
  asserta(xpozitiv(X,Y)).
verif_istoric(X,Y,nu):-
  asserta(negativ(X,Y)),
  fail.
șterge_date_bază:-
  retract(xpozitiv(_,_)),fail.
șterge_date_baza:-

```

```

    retract(xnegativ(_,_)),fail.
șterge_date_bază:-
    write("\n\n Apăsați tasta spațiu pentru ieșire"),
    readchar(_).
vietate_este(leopard) if
    are_tribut(mamifer),
    are_tribut(carnivor),
    pozitiv(are,culori_simple),
    pozitiv(are,puncte_negre),!.
vietate_este(tigru) if
    are_tribut(mamifer) and
    are_tribut(carnivor) and
    pozitiv(are,culori_simple) and
    pozitiv(are,dungi_negre),!.
vietate_este(girafa) if
    are_tribut(cu_unghii) and
    pozitiv(are,gît_lung) and
    pozitiv(are,picioare_lungi) and
    pozitiv(are,pete_îchise),!.
vietate_este(zebra) if
    are_tribut(cu_unghii) and
    pozitiv(are,dungi_negre),!.
vietate_este(stîrc) if
    are_tribut(pasăre) and
    not(pozitiv(acțiune,zboară)) and
    pozitiv(are,gît_lung) and
    pozitiv(are,picioare_lungi),!.
vietate_este(pinguin) if
    are_tribut(pasăre) and
    not(pozitiv(acțiune,zboară)) and
    pozitiv(acțiune,înoată) and
    pozitiv(are,culoare_alb_negru),!.
vietate_este(albatros) if
    are_tribut(pasăre) and
    pozitiv(acțiune,zboară),
    pozitiv(acțiune,zboară_bine),!.
are_tribut(mamifer) if
    pozitiv(are,păr),
    pozitiv(acțiune,dă_lapte),!.
are_tribut(carnivor) if
    are_tribut(mamifer),
    pozitiv(acțiune,mănîncă_carne),
    pozitiv(are,dinți_ascuțiți),
    pozitiv(are,gheare),!.
are_tribut(cu_unghii) if
    are_tribut(mamifer),
    pozitiv(are,țipăt),
    pozitiv(acțiune,ierbivor),!.
are_tribut(pasăre) if
    not(pozitiv(are,păr)),

```

```
not(pozitiv(acțiune,dă_lapte)),
pozitiv(are,pene),
pozitiv(acțiune,dep_ouă),!.
```

9. Să se scrie pe ecran grafic un mesaj.

R: Următorul program, scris în Microsoft assembler (masm) (v 5,0), scrie la adresa video B800h mesajul specificat în segmentul de date DS.

```
dosseg
.model small
.stack 100h
.data
cd db "adriana"
.code
start:mov ax, data
      mov ds,ax
      mov ax,B800h
      mov es,ax
      lea bx,cd
      xor di,di
ciclu:
      cmp[bx],"o"
      je gata
      mov es:[di],[bx]
      inc di
      inc di
      inc bx
      jmp short ciclu
gata:xor ax,ax
      mov ax,4C00h
      int 21h
      end start
```

Probleme propuse

Rezolvarea următoarelor probleme se poate face în orice limbaj de programare și pentru orice sistem hardware. Este indicată folosirea unor tehnici de programare care să permită implementarea lor pe o gamă cât mai largă de calculatoare.

1. Se dau două segmente, S_1 și S_2 , cunoscute prin coordonatele extremităților. Știind că distanța dintre S_1 și S_2 este dată de $d(S_1, S_2) = \inf\{d(x, y), x \in S_1, y \in S_2\}$ să se elaboreze un algoritm care să determine punctele $x_0 \in S_1$, $y_0 \in S_2$ cu proprietatea că $d(S_1, S_2) = d(x_0, y_0)$ și să se calculeze $d(x_0, y_0)$. Să se scrie programul corespunzător în limbajul FORTRAN IV.

Rezolvare (model):

Fie $(x(1), y(1))$ și $(x(2), y(2))$ coordonatele vîrfurilor segmentului S_1 iar $(x(3), y(3))$ și $(x(4), y(4))$ cele ale vîrfurilor lui S_2 . Ecuațiile dreptelor generate de segmentele S_1 și S_2 le scriem sub forma $y = mx + n$ iar pentru calcularea distanței de la punctul (x, y) la dreapta de ecuație $y = mx + n$ folosim formula:

Probleme propuse

$$d = \frac{y - mx - n}{\sqrt{1 + m^2}}$$

Algoritmul este următorul:

Pasul 1: Se citesc datele de intrare $X(I), Y(I), I=1,4$ și se calculează $M_1 = (Y(2)-Y(1))/(X(2)-X(1))$, $M_2 = (Y(4)-Y(3))/(X(4)-X(3))$, $N_1 = Y(1)-M_1 \cdot X(1)$, $N_2 = Y(2)-M_2 \cdot X(2)$; M_1 și N_1 corespund segmentului S_1 iar M_2 și N_2 lui S_2 . Se merge la pasul 2.

Pasul 2: Dacă $M_1 \neq M_2$ sau $N_1 \neq N_2$ se merge la pasul 3. În caz contrar se calculează $XMAX = \max(X(1), X(2))$, $YMAX = \max(Y(1), Y(2))$, $XMIN = \min(X(3), X(4))$, $YMIN = \min(Y(3), Y(4))$. Dacă $XMIN \leq XMAX$ cele două segmente au puncte comune și deci distanța este $D = 0$; în caz contrar $D = XMIN - XMAX$. Se scrie mesajul "segmentele date fac parte din drepte confundate iar distanța dintre ele este $D =$ " și se merge la pasul 1 pentru o nouă citire.

Pasul 3: Dacă $M_1 = M_2$ se merge la pasul 4. În caz contrar se determină punctul de intersecție (XI, YI) al dreptelor generate de segmentele S_1 și S_2 . Se calculează $K_1 = (XI-X(1)) \cdot (XI-X(2))$ și $K_2 = (XI-X(3)) \cdot (XI-X(4))$. În cazul când $K_1, K_2 \leq 0$, cele două segmente se intersectează în punctul (XI, YI) , se scriu coordonatele punctului (XI, YI) și valoarea distanței ($D=0$) apoi se merge la pasul 1.

Pasul 4: Folosind subrutina DIST se determină distanțele $D(I), 1 \leq I \leq 4$ de la fiecare vîrf al unui segment la cel de-al doilea segment. Apoi se calculează distanțele $D(J), 5 \leq J \leq 8$ de la vîrfurile lui S_1 la cele ale lui S_2 . Folosind subrutina MIN se determină $VD = \min\{D(I), 1 \leq I \leq 8\}$ și indicele K pentru care se realizează acest minim apoi se merge la pasul 5.

Pasul 5: Se afișează valoarea distanței dintre cele două segmente și punctele între care se realizează apoi se merge la pasul 1.

Observație: Algoritmul se oprește cînd s-a detectat EOF la pasul 1.

Subrutina $DIST(X, Y, I_1, I_2, I_3, RM, RN, DC, XC, YC)$ calculează distanța de la PUNCTUL $(X(I_1), Y(I_1))$ la dreapta $y = RM \cdot x + RN$ determinată de punctele $(X(I_2), Y(I_2)), (X(I_3), Y(I_3))$. XC și YC sînt coordonatele punctului de pe dreapta $y = RM \cdot x + RN$ situat la distanța DC de punctul $(X(I_1), Y(I_1))$. Algoritmul corespunzător acestui subprogram este următorul:

Pasul 1: Se calculează distanța de la $(X(I_1), Y(I_1))$ la dreapta $y = RM \cdot x + RN$. Dacă $RM \neq 0$ se merge la pasul 2. În caz contrar se fac atribuirile $XC = X(I_1)$, $YC = Y(I_2)$ și se merge la pasul 3.

Pasul 2: Se determină coordonatele XC, YC ale intersecției dreptei $y = RM \cdot x + RN$ cu perpendiculara coborîită din $(X(I_1), Y(I_1))$ pe ea apoi se merge la pasul 3.

Pasul 3: Dacă $(XC-X(I_2)) \cdot (XDC-X(I_3)) > 0$ se face $DC = 0$ deoarece punctul (XC, YC) este exterior segmentului de vîrfuri $(X(I_2), Y(I_2)), (X(I_3), Y(I_3))$.

Subprogramul $MIN(D, VD, K)$ calculează $VD = \min\{D(I) \neq 0, 1 \leq I \leq 8\}$ și indicele K ce realizează acest minim.

Algoritmul este următorul:

Pasul 1: $VD = 0, I = 2$.

Pasul 2: Dacă $D(I) = 0$ se merge la pasul 5, în caz contrar se merge la pasul 3.

Pasul 3: Dacă $VD = 0$ se face $VD = D(I)$, $K = I$ și se merge la pasul 5, în caz contrar se merge la pasul 4.

Pasul 4: Dacă $VD < D(I)$ se merge la pasul 5, în caz contrar se face $VD = D(I)$, $K = I$ și se merge la pasul 5.

Probleme propuse

Pasul 5: Dacă $I < 8$ se face $I = I+1$ și se merge la pasul 2, în caz contrar RETURN.

2. Algoritmul de mai jos calculează valorile unei funcții $F(X,M)$. Care este această funcție? Să se scrie, în limbajul FORTRAN IV, programul corespunzător algoritmului următor:

Pasul 1: Se citesc variabilele M (de tip întreg) și X (de tip real) apoi se inițializează $I = 1$.

Pasul 2: Dacă $I \leq X$ se face $I = I+1$ și se merge la pasul 2, în caz contrar $K = I-1$ și se merge la pasul 3.

Pasul 3: $I = K+M-1$, $RM = M \cdot X$ apoi se merge la pasul 4.

Pasul 4: Dacă $RM \geq I$ se calculează $R = I-M \cdot K$ și STOP, în caz contrar se merge la pasul 4.

Rezolvare:

Pasul 2 determină cel mai mare întreg mai mic decât X ; deci $K = [X]$. Apoi $X = [X] + \{X\}$ și deci $M \cdot X = M[X] + M\{X\}$. Dar $\{X\} < 1$; rezultă $M\{X\} \leq M-1$ și deci $M \cdot X \leq M[X] + M-1$. Așadar, $[M \cdot X] \leq M[X] + M-1 = M \cdot K + M-1$. Rezultă că pasul 3 reține cea mai mare valoare posibilă a lui $[M \cdot X]$ și calculează produsul $M \cdot X$. Pasul 4 determină cel mai mare întreg I mai mic decât $M \cdot X$ și calculează $R = I - M \cdot K = [M \cdot X] - M[X]$. Deci $F(X,M) = [M \cdot X] - M[X]$.

Următoarele probleme sînt propuse spre rezolvare, fără indicații.

3. Să se determine o strategie de rezolvare a problemei celor 8 dame (pe o tablă de șah să se plaseze 8 dame care să nu se atace reciproc) atît într-un limbaj convențional cît și în limbajul Prolog.

4. Să se rezolve dielma:

Un sătean are de trecut de pe un mal al unui rîu pe celălalt un lup, o capră și o varză. El are la dispoziție o barcă în care nu au loc decît două ființe sau o ființă și un obiect. Cum poate reuși săteanul a transporta în siguranță pe malul celălalt lupul, capra, varza știind că lupul nu poate rămîne singur cu capra iar capra nu poate rămîne singură cu varza.

5. Să se determine o strategie de simulare a cubului Rubik.

6. Pentru utilizatorii de PC-uri:

Să se scrie o rutină care să redirecțeze o întrerupere (este indicată de INT 5) către un program rezident care să permită tipărirea unui ecran grafic.

7. Concepeți o rutină care să genereze toate submulțimile cu k elemente ale unei mulțimi cu n elemente.

8. Elaborați un algoritm care să permită mărirea sau micșorarea unui desen grafic cu păstrarea cotelor punctuale (principiul măririi fotografice).

9. Să se elaboreze un program de rezolvare a șirurilor recurente liniare de forma:

$$a \cdot x_{n+1} + b \cdot x_n + c \cdot x_{n-1} = 0$$

unde a, b, c sînt constante reale iar $x_0 = i, x_1 = j, i, j$ valori particulare date.

La granița dintre informatică și matematică Jocuri, probleme distractive, strategii.

Lexicul informaticii. Paradigma calculatoarelor.

Deosebitul dinamism care caracterizează informatica se reflectă în terminologia specifică domeniului atât prin creșterea în volum cât și prin mutații semantice și relaționale în cadrul acesteia. Limitînd investigația numai la nivelul sferei semantice "calculator (electronic)" se poate constata că această noțiune a generat, prin extensie semantică, un vast cîmp lexico-semantic incluzînd totalitatea tipologică a sistemelor de prelucrare a datelor cunoscute actualmente. Acest domeniu poate fi analizat paradigmatic ca un ansamblu lexico-semantic cu caracter taxonomic. Cardinalitatea apreciabilă a mulțimii elementelor acestei paradigme lexico-semantice își are sorginea atât la palierul lexical cât și, în special, la cel frazeologic (sintagmatic).

La nivel lexical numărul elementelor este relativ restrîns impunîndu-se constatarea unei sinonimii totale:

a) atât a elementelor simple: calculator (forma scurtă a sintagmei "calculator electronic"), computer (din limba engleză) și ordinator (din limba franceză);

b) cât și a elementelor compuse/derivate (serii de triplete) formate cu acestea prin utilizarea prefixelor micro, mini și super caracterizate prin frecvență mare (microcalculator, microordinator, microcomputer, minicalculator etc.) sau maxi, midi, macro, para, ultra caracterizate prin frecvență mică (macrocalculator, paracomputer etc.);

c) precum și a celor formate prin dublă prefixare (super-microordinator, super-minicalculator, mega-minicalculator etc.) printre care unele sînt net contrastante (mega-minicalculator). Situația este generată de etimologiile diferite ale celor trei termeni de bază (de menționat că termenul ordinator este adaptarea termenului francez ordinateur (care a fost creat special pentru a se evita folosirea celui din engleză!) fiind de relevat următoarele aspecte:

a) specialiștii domeniului, informaticienii, utilizează în exclusivitate, de la începuturile informaticii românești, forma calculator (primul calculator românesc, realizat în 1957, se numea CIFA - Calculatorul IFA; alt exemplu timpuriu: CET - calculator electronic tranzistorizat);

b) formele elementare computer și ordinator, precum și cele compuse constituite pe baza acestora, sînt utilizate în exclusivitate de non-informaticieni (ziariști, publiciști) în special în periodice dar și în alte publicații, în mod abuziv, uneori exclusiv (deci nu din considerente de natură stilistică - evitarea repetării unui termen), sub influența limbii surselor de documentare;

c) dicționarul lingvistic cel mai răspîndit - DFX - tratează deficitar cei trei termeni, în ambele ediții, lăsînd impresia că au semantică parțial diferită (calculator... s.n. - mașină sau instalație cu ajutorul căreia se efectuează automat operații matematice și logice...; computer s.n. - calculator electronic, ordinator...; ordinator s.n. - calculator numeric universal...) ceea ce poate induce confuzie alimentînd predilecția publiciștilor de a utiliza

xenisme; inconsecvența DEX-ului este eliminată în Dicționarul de neologisme (ediția 1978) prin includerea sintagmei calculator electronic la computer și ordinator făcându-se trimitere la aceasta (principiul circularității sinonimelor);

d) sinonimia de tip sinecdocă (parte pentru întreg) prin utilizarea termenului procesor și a derivatelor obținute prin prefixare (monoprocessor, biprocessor, multiprocessor).

La nivel frazeologic (sintagmatic) se constată existența sinonimiei cvasitotale lexico-frazeologice cu noțiunea în studiu: sintagma uzuală sistem (electronic) de calcul și formele derivate prin prefixare cu mini și micro, sintagma "arhaică" mașină electronică de calcul și sintagma metaforă "creier electronic". Trebuie relevată utilizarea inadecvată a termenului sintagmă calculator (de buzunar) pentru a desemna minusculele mașini de calculat pentru care în limba engleză se folosește un termen special: calculator - eroare comună devenită regulă!

Elementele sferei semantice analizate constituie un dicționar-tezaur structurat taxonomic ceea ce facilitează reliefația relațiilor semantice de tip sinonimie, antonimie, polisemie și hponimie/hiperonimie (relație de incluziune/apartenență în structuri lexico-semantice).

Deoarece terminologia domeniului se află în plin proces de formare atît ca urmare a progresului tehnic, care conduce la creșterea performanțelor și la caracteristici tehnico-funcționale tot mai ridicate ale echipamentelor, semantica multor noțiuni are caracter dinamic-glisant. Cîteva exemple elocvente:

a) calculatorul de capacitate medie-mare la nivelul 1980, FELIX C-512/1024, cîteva ani mai tîrziu este considerat de capacitate medie;

b) minicalculatorul se definea ca un calculator de capacitate mică și preț redus pentru ca acum să desemneze mașini cu performanțe care odinioară erau considerate specifice calculatoarelor de capacitate medie-mare;

c) microcalculatorul desemna inițial un sistem de calcul de capacitate mică iar acum se fabrică microcalculatoare cu putere de calcul echivalentă calculatoarelor de capacitate medie-mare.

Din aceleași considerente taxonimia domeniului este foarte controversată în literatura de specialitate, eterogenitatea deosebită a acesteia nepermițînd o clasificare absolut riguroasă.

Dicționarul-tezaur include cvasitotalitatea noțiunilor utilizate în literatura de specialitate și publicații de uz general, în special presa scrisă (cuvîntul-temă calculator este abreviat prin c. iar sinonimul acestuia - sistem - prin s.; bara oblică "/" separă sinonimele sau alternativele, parantezele introduc elemente opționale din sintagme, forma echivalentă din limba engleză sau explicații suplimentare, după caz).

Dicționar-tezaur structurat taxonomic.

1. calculator (c.)/calculator electronic/mașină electronică de calcul (rar)/sistem (electronic) de calcul/sistem de prelucrare a datelor:

a) c. analogic, c. numeric/digital, c. hibrid (analogic-numeric);

b) c. de generația I-a, a 2-a, a 3-a, a 3,5-a, a 4-a, a 5-a;

c) c. de capacitate mică, medie/minicalculator, medie-mare, mare/macrocalculator/maxicalculator, foarte mare (mainframe).

2. minicalculator/minicomputer/microordinator/mini-sistem (de calcul) /sistem minicalculator;

3. microcalculator/microcomputer/microordinator/microsistem (de calcul) /sistem microcalculator;

Lexicul informaticii. Paradigma calculatoarelor.

4. c. personal (personal-computer, denumire improprie, este microcalculator):
 - a) c.p. portabil/individual;
 - b) c.p. familial/domestic (home-computer);
 - c) c.p. semiprofesional;
 - d) c.p. de birou (desk-top computer);
 - e) c.p. profesional/de întreprindere (business computer).
5. c. de buzunar (calculator):
 - a) c.b. neprogramabil/minimal/algebric, c.b. științific (set extins de operații);
 - b) c.b. programabil (C.B.P.) în limbaje neevolute (notația poloneză inversă sau limbaj de tip algebric), în limbaje evolute (BASIC);
 - c) c.b. procesor de texte/cuvinte.
6. c. ceas (brățară) sau ceas-calculator:
 - a) cu funcții identice c. de buzunar;
 - b) specializate: medical/biologic, dicționar bi- sau plurilingv, agendă electronică.
7. c. specialitate funcțional:
 - a) procesor/calculator frontal (front-end procesor)/de teletransmisie/c. satelit;
 - b) procesor/calculator dorsal (back-end processor)/pentru gestiunea datelor/procesor pentru baze de date, procesor pentru gestiunea memoriei interne comune/procesor pentru bănci de memorie;
 - c) procesor/calculator de rezervă (back-up processor) și c. de bază (primary computer) în sistem tolerante la erori/multisisteme redundante/sisteme cu fiabilitate ridicată;
 - d) c. gazdă (host-computer), c. principal/central/"master";
 - e) s. cu prelucrare pe loturi (batch-processing), cu multiacces/cu acces multiplu, cu timp divizat (time-sharing), cu procesoare la distanță (remote processor), cu stații terminale/satelit (remote job entry, remote batch);
 - f) terminal inteligent/terminal programabil/terminal pentru achiziție /prelucrare date (microcalculatoare).
8. c. universal și c. dedicate/specializate (aplicativ)/"la cheie":
 - a) c. de proces/pentru conducerea proceselor;
 - b) procesoare de texte/cuvinte specializate sau semispecializate (word-processor);
 - c) c. de bord (auto, spațiale, aviație);
 - d) c. naval;
 - e) c. radar;
 - f) automate de șah;
 - g) c. muzical.
9. s. mono/uni-calculator, mono- sau multiprocesor:
 - a) s. monocalculator/simplex/uniprocessor/unicalculator;
 - b) s. biprocessor sau bi-microprocesor, cu procesoare identice sau diferite/complementare;
 - c) s. monocalculator multiprocesor sau multi-microprocesor.
10. s. mono/multi-flux de instrucțiuni/date:
 - a) s. cu un singur flux de instrucțiuni și un singur flux de date (Single Instruction Single Date stream - SISD)/sistem serial (calculatoare uzuale);
 - b) s. cu mult flux de instrucțiuni și un singur flux de date (Multiple

Instruction Single Data stream - MISD)/supercalculator/s. pseudo-multiprocesor (monoprocesor multi-UAL cu prelucrări paralele diferite); tipuri: conductă (pipeline) cu UAL înseriate și uni/multifuncțional cu multi-unități funcționale de tip static/dinamic;

c) s. cu un singur flux de instrucțiuni și multiflux de date (Single Instruction Multiple Data stream - SIMD)/supercalculator/s. cvasi-multiprocesor/s. (cu prelucrări/procesoare) paralele, cu mono-unitate de comandă și multi(procesor+memorie); tipuri: procesor matricial, procesor (cu memorie) asociativ(ă), complet paralel, serial (pe bit, octet, cuvânt, bloc), matricial-asociativ, orgofonal;

d) s. cu multiflux de instrucțiuni și multiflux de date (Multiple Instruction Multiple Data stream - MIMD)/s. multiprocesor propriu-zis/s. strâns cuplate; tipuri: cu magistrală unică, paracomputer/ultracomputer/cu memorie comună, cu comutator încrucișat, cu memorii multipoartă, s. multitask și date multiple (Multiple Task Multiple Data - MTMD)/sistem multiprocesor orientat pe taskuri.

11. supercalculator/supercomputer/superordinator/supersistem (de calcul):

a) sisteme tip SIMD, MISD, MIMD (vezi 10.b, c, d);

b) super-minicalculator, super-microordinator, mega-minicalculator, mega-microcalculator, supercalculator personal (generația a 5-a), supercalculator inferențial (generația a 5-a).

12. transputer/microcalculator integrat (computer-on-a-chip).

13. s. multicalculator/multisisteme/multi-minisisteme:

a) s. bicalculator/duplex, cuplate slab, fie direct (c. principal - c. frontal) fie prin acces la discuri magnetice comune;

b) s. cu configurație extinsă/completă formate dintr-un c. principal, un procesor frontal, un procesor dorsal și un procesor de rezervă;

c) s. cuplate moderat/s. distribuite local;

d) s. cuplate slab în rețele de calculatoare cu tipurile topologice: inel, complet interconectate, cu memorie centrală, cu magistrală globală, stea, magistrală cu comutator central, rețea regulată, rețea neregulată, rețea ierarhică.

14. s. de prelucrare a cunoștințelor (knowledge information processing systems)/c. de generația a 5-a:

a) componente principale: mașină inferențială paralelă pentru rezolvarea problemelor, mașină pentru baza de cunoștințe, mașină de interfață inteligentă;

b) mașini funcționale pentru: comunicații, conducerea proceselor, service, asistare utilizator, simulare, calcule numerice ultrarapide, date de tip abstract, BD relaționale, manipulare simbolică, euristică;

c) supercalculator inferențial, mașină personală inferențială;

d) mașină de tip flux de date (data flow), sistem sistolic (rețea sincronă de procesoare paralele).

15. c. la nivel teoretic/conceptual, de perspectivă:

a) c. abstract/mașină abstractă/mașină Turing;

b) c. cu arhitectură tip von Neumann, cu program memorat, tip non-von Neumann/post-von Neumann, tip Harvard;

c) c. optic;

d) c. biologic/molecular.

16. alte tipuri:

a) c. cu o adresă, c. cu mai multe adrese.

O definire matematică a careurilor de cuvinte încrucișate

Se consideră o mulțime $A \neq \emptyset$ finită numită alfabet și $\pi \in A$ un element numit punct negru. Definim:

$$A^p = \{(a_1, a_2, \dots, a_p) \mid a_i \in A \forall i=1, 2, \dots, p\}, \text{ iar}$$

$V = \bigcup_{p \geq 1} A^p$ îl vom numi vocabular cu elemente $w \in V$ numite cuvinte.

Se notează $w = (w_1, w_2, \dots, w_p) = w_1 w_2 \dots w_p$.

Dacă $w = w_1 w_2 \dots w_p \in V$ atunci $|w| = p$ reprezintă lungimea lui w .

Fie $D \subset V$, $D \neq \emptyset$ o mulțime finită numită dicționar. Fie D_T, D_N două submulțimi $C D$ cu proprietățile $D_T \cup D_N = D$ și $D_T \cap D_N = \emptyset$, mulțimi numite dicționar tematic, respectiv dicționar matematic.

Fie $u \in \mathbb{N}^*$, $I_n = \{1, 2, \dots, n\}$.

Definiție:

$K^1, K^2 \in \mathcal{K}(AU\{\pi\})$; $K = (K^1, K^2)$ cu proprietatea $(K^1)^t = K^2$ se numește careu de dimensiune n .

În cele ce urmează vom considera K un careu de dimensiune n .

Definiție:

$w = w_1 w_2 \dots w_p$ este inclus în K ($w \in K$) dacă:

$$\exists t \in \{1, 2\}, \exists i, j, l \in \{1, 2, \dots, n\} = I_n$$

astfel încît:

a) $i = 1$ sau ($i \neq 1$ și $K^t_{1, i-1} = \pi$);

b) $j = n$ sau ($j \neq n$ și $K^t_{1, j+1} = \pi$);

c) $\forall q = 1, 2, \dots, p$, $w_q = K^t_{i+q-1}$.

Definiție:

K are licența dacă $\exists w \in K$ astfel încît $w \in D$.

Definiție:

Fie $w, v \in D$, $m = v$ dacă $|w| = |v| = p$ și $\forall r = 1, 2, \dots, p$, $w_r = v_r$.

Fie \sim o relație de echivalență în D numită relație de familiaritate iar D/\sim mulțimea claselor de echivalență.

Definiție:

K are cuvinte din aceeași familie dacă:

$$\exists w_1, w_2 \in K, w_1 \neq w_2 \text{ și } w_1 = w_2.$$

Definiție:

K are baré dacă: $\exists t \in \{1, 2\}$; $\exists i, j \in I_n$ cu $K^t_{ij} = \pi$, astfel încît $K^t_{ij} = \pi$ sau $K^t_{ij} = \pi$ unde $i \in \{i-1, i+1\} \cap I_n$ și $j \in \{i-1, j+1\} \cap I_n$.

Fie $d_T, d_P \in [0, 1]$ reprezentînd densitatea tematică respectiv densitatea de puncte negre a careului K .

Definim funcția: $\varphi_T : D \rightarrow \mathbb{N}$,

$$\varphi(w) = \begin{cases} |w|, & w \in D_T \\ 0, & w \notin D_T \end{cases}$$

O definiție matematică a careurilor de cuvinte încrucișate

iar $N_T = \sum_{w \in K} \varphi_T(w)$ reprezintă numărul de litere tematice a lui K .

Definiție:

K are cel puțin d_T densitate tematică dacă $N_T \geq 2d_T(1-d_p)n^2$.

Definim funcția: $\varphi_p : A \cup \{\pi\} \rightarrow \{0,1\}$, cu:

$$\varphi_p(a) = \begin{cases} 1 & a = \pi \\ 0 & a \neq p \end{cases}$$

iar $N_p = \frac{1}{2} \sum_{a \in K} \varphi_p(a)$ reprezintă numărul de puncte negre a lui K .

Definiție:

K are cel mult d_p densitate de puncte negre dacă $N_p \leq d_p n^2$.

Definiție:

Fie: $(i,j), (k,l) \in I_n * I_n$

$(i,j) \neq (k,l)$ sînt vecine dacă:

a) $\{i,j\} \cap \{1,n\} \neq \emptyset$ și $\{k,l\} \cap \{1,n\} \neq \emptyset$

sau

b) $i \in \{k-1, k, k+1\} \cap I_n$ și $j \in \{l-1, l, l+1\} \cap I_n$.

Definiție:

K are semiînchideri dacă:

$\exists t \in \{1,2\}, \exists s > 2, \exists (i_1, j_1), \dots, (i_s, j_s) \in I_n \times I_n$ astfel încît:

a) $\forall r = 1, \dots, s-1, K_{i_r, j_r}^t$ vecin cu $K_{i_{r+1}, j_{r+1}}^t$ și K_{i_s, j_s}^t vecin cu K_{i_1, j_1}^t ;

b) $r-1 \leq \text{card} \{K_{i_r, j_r}^t \mid K_{i_r, j_r}^t = \pi\} \leq r$.

Fie $d_T, d_p \in \{0,1\}$.

Definiție:

K este careu corect încrucișat dacă și numai dacă:

L) K nu are licențe;

B) K nu are bare;

F) K nu are cuvinte din aceeași familie;

T) K are cel puțin d_T densitate tematică;

N) K are cel mult d_p densitate de puncte negre;

S) K nu are semiînchideri.

Fie $(S_x)_{x \in D}$ o familie de mulțimi unde $\forall x \in D, S_x$ reprezintă mulțimea semnificațiilor (definițiilor) lui x .

Fie $D' \subset D$. Definim $S_{D'} = \cup \{y \mid y \in S_x\}$ un sistem de definiții pentru D' .

Evident $\cup_{w \in K} w \subset D$.

Notăm $S = \cup_{w \in K} w$ un sistem de definiții pentru K .

Definiție:

$\Sigma = (K, S)$ se numește un careu de cuvinte încrucișate dacă:

a) K este corect încrucișat;

b) S reprezintă un sistem de definiții pentru K .

O problemă de ordonare în teoria grafurilor

O pereche (E, V) se numește graf dacă E este o mulțime de puncte și $V \in P_2(E)$ unde $P_2(E) = \{(a, b) | a, b \in E \text{ și } a \neq b\}$.

Fiind date $x, y \in E$ numim drum de la x la y o succesiune $\{x, b_1\}, \{a_1, b_2\}, \dots, \{a_n, y\} \in V$ dacă $\forall i = \{1, \dots, n\} b_i = a_{i+1}$. Un graf $G = (E, V)$ este conex dacă: $\forall x, y \in E$ există un drum de la x la y . $G' = (E', V')$ este subgraf a lui $G = (E, V)$ dacă $E' \subseteq E$ și $V' \subseteq V$ și cel puțin una din incluziuni este strictă (scriem $G' \subset G$). $G' \subset G$ se numește componentă conexă a lui G dacă G' este conex și este maximal cu această proprietate.

Un graf G este un arbore dacă este conex și graful G' obținut din G prin suprimarea unei muchii nu mai este conex. Un graf G este o pădure dacă orice componentă conexă a lui G este un arbore. Un graf G este marcat dacă nodurilor sale li s-a dat un nume.

În continuare introducem pe mulțimea arborilor care formează o pădure relația notată \sim . Fie $A_1 = (V_1, E_1)$ și $A_2 = (V_2, E_2)$ doi arbori ai unei păduri cu rădăcinile respectiv a_1 și a_2 de nume numel și nume2.

Spunem că $A_1 \sim A_2$ dacă și numai dacă prin definiție $|V_1| > |V_2|$ sau dacă $|V_1| = |V_2|$ și numel $>$ nume2, unde $>$ este relația de mai mare pe mulțimea numerelor naturale și $>$ este relația de ordine lexicografică naturală.

Definiție: Două structuri de date prezentate sub formă de păduri sînt echivalente dacă, pentru o specificație corectă, elementele desemnate sînt aceleași.

Pentru a avea elementele definite fără ambiguități facem următoarele ipoteze:

1. Rădăcinile tuturor arborilor care formează pădurea sînt distincte;
2. Pentru orice nod subarborile care are drept rădăcină nodul respectiv nu conține un alt nod cu același nume;
3. Pentru orice nod neterminal fiii săi au nume distincte.

Fie A_1, \dots, A_n arbori și $A_i = (V_i, E_i)$, $\forall i = \{1, \dots, n\}$ și fie pădurea P formată cu acești arbori $P = (\bigcup_{i=1}^n V_i, \bigcup_{i=1}^n E_i)$

Propoziția 1: Relația \sim este o relație de ordine totală.

Demonstrație: Presupunem $A_i \sim A_j$ și $A_j \sim A_k$ $i \neq j \neq k, i \neq k$.

Putem avea următoarele cazuri:

(i) $|V_i| > |V_j|$ și $|V_j| > |V_k|$ și din tranzitivitatea relației de ordine pe N rezultă: $|V_i| > |V_k|$, deci $A_i \sim A_k$;

(ii) $|V_i| = |V_j|$ și $|V_j| > |V_k| \Rightarrow |V_i| > |V_k| \Rightarrow A_i \sim A_k$;

(iii) $|V_i| > |V_j|$ și $|V_j| = |V_k| \Rightarrow |V_i| > |V_k| \Rightarrow A_i \sim A_k$;

(iv) $|V_i| = |V_j|$ și $|V_j| = |V_k| \Rightarrow |V_i| = |V_k|$ *.

Deoarece $A_i \sim A_j \Rightarrow \text{nume}(A_i) > \text{nume}(A_j)$

$A_j \sim A_k \Rightarrow \text{nume}(A_j) > \text{nume}(A_k)$

Din tranzitivitatea relației de ordine lexicografică rezultă că avem: $\text{nume}(A_i) > \text{nume}(A_k)$ **.

Din relațiile * și ** rezultă $A_i \sim A_{i,k}$.

În mod analog dacă $A_i \sim A_{k,j}$ rezultă că nu putem avea și $A_j \sim A_i$. Din

O problemă de ordonare în teoria grafurilor

această propoziție și folosind ipoteza 1 rezultă că putem face o renumerotare a arborilor astfel încât să avem:

$$A_1 \sim A_2 \sim \dots \sim A_n.$$

Fie un arbore A . Graful obținut prin suprimarea nodului rădăcină și a muchiilor adiacente cu aceasta formează o pădure de componente B_1, B_2, \dots, B_k .

Conform rezultatelor anterioare și ipotezei 3 putem face o renumerotare a componentelor B_j astfel încât $B_1 \sim B_2 \sim \dots \sim B_k$.

Propoziția 2: Fie o pădure P de componente A_1, A_2, \dots, A_n . Fie pădurea P' de componente A'_1, A'_2, \dots, A'_n obținută prin aplicarea pașilor următori:

I. Pentru fiecare $i, 1 \leq i \leq n$; pentru fiecare nod neterminal x se face ordonarea subarborilor care au drept rădăcină fiii nodului x ordonarea făcându-se conform relației \sim ;

II. Se ordonează componentele pădurii P conform relației \sim .

Dacă P verifică ipotezele 1., 2. și 3. atunci cele două păduri sînt echivalente.

Demonstrație: Fie o specificație Y_1 of Y_2 of \dots of Y_k pe care o presupunem corectă. Din definirea specificației rezultă că există și este unic un $l, 1 \leq l \leq n$ astfel încât $Y_k \in A_l$.

Fie G_k subarborile lui A_l de rădăcină Y_k . Deoarece P verifică dacă rădăcina arborelui A_l este rădăcină a arborelui $A' \pi_k(l)$ în P' deci există un unic $l', l' = \pi_k(l)$ astfel încât $Y_k \in A'_{l'}$.

Fie G'_k subarborile lui $A'_{l'}$ de rădăcină Y_k .

Presupunem că am determinat în mod unic Y_i în P și în P' și arătăm că Y_{i-1} este determinat în mod unic, $1 \leq i \leq k$ deci și G_i și G'_i .

Fie T pădurea care se obține din G_i prin suprimarea lui Y_i și a muchiilor adiacente cu el și fie T' pădurea care se obține din G'_i în mod analog.

Presupunem T de componente B_1, B_2, \dots, B_s și T' de componente B'_1, B'_2, \dots, B'_s . Deoarece specificația este corectă rezultă că există și este unic $l, 1 \leq l \leq s$, astfel încât $Y_{i-1} \in B_l$.

Dacă T satisface ipotezele 1., 2. și 3. atunci există o permutare π_{i-1} astfel încât rădăcina arborelui B_l este rădăcină a arborelui $B' \pi_{i-1}(l)$ deci există și este unic $l', l' = \pi_{i-1}(l)$ astfel încât $Y_{i-1} \in B'_{l'}$.

Fie G_{i-1} subarborile lui B_l de rădăcină Y_{i-1} și G'_{i-1} subarborile lui $B'_{l'}$ de rădăcină Y_{i-1} .

Această inducție inversă demonstrează propoziția dacă arătăm că B_1, B_2, \dots, B_s satisfac ipotezele 1., 2. și 3.

Deoarece ipotezele 2. și 3. se referă la un nod din P și orice nod din T este un nod din P rezultă că ele sînt verificate și pentru T . Ipoteza 1. este verificată deoarece, conform ipotezei 3., pentru orice nod fiii săi au nume distincte.

În particular, Y_i este un nod al unui arbore deci fiii săi au nume distincte. Deoarece fiii nodului Y_i sînt rădăcinile arborilor B_1, B_2, \dots, B_s și sînt distincți, rezultă că T verifică și ipoteza 1. Acest lucru completează demonstrația propoziției.

Curier editorial

Așadar, să inaugurăm prin intermediul revistei primul nostru contact cu cititorii. Cum nu putem aprecia în acest stadiu de început opiniile dorințele sau sugestiile dumneavoastră vă vom veni noi în întâmpinare prin câteva propuneri și informații utile.

Astfel, să intitulăm prima rubrică a curierului editorial:

1. Sugestii și servicii software

În cadrul acestei secțiuni așteptăm de la cititori un contact de propuneri și oferte de colaborare sau, de ce nu a unei rubrici de mică publicitate informatică.

Astfel anunțăm cititorii că la sediul redacției pot primi informații, sfaturi și ajutor în obținerea sau cunoașterea unor pachete software sau cărți privind lucrul pe microcalculatoarele personale.

- Utilitare: Laplink III (pachet de comunicație); Lotus Magelan; Mirror; Sideways; Freelance; Xtree; Norton Utilities; C-Toolbox;
- Software integrat: Lotus; Symphony; Framework III;
- Procesare de text: Word Perfect 4.1. și 5.1.; WordStar 2000 Plus 3.0.;
- Pachete de grafică: Vermont Vision 1.0; Design CAD 3-D;
- Baze de date: Oracle 5.0 și 6.0; dBASE III, IVPlus; SBT; SQL SERVER.
- Compilatoare: Basic; GW - Basic; C (Turbo, Microsoft, Lattice); Turbo Pascal 3.0., 4.0., 5.0.; Turbo Prolog; Cobol; Fortran 77; Assembler;
- Rețele locale: Novell; 3+Comm; Token Ring; Crosstalk Lan.
- Pachete contabil-finnciare: Sun Account; SBT.

O altă rubrică s-ar intitula:

2. Biblioteca revistei

Aici, redacția va pune în timp la dispoziția cititorilor o bibliotecă cu cărți și reviste de specialitate sub forma lor fizică sau pe suport magnetic.

Sperăm ca și revista să poată fi obținută pe suport magnetic, permițând celor ce n-au citit-o sau vor să primească numai anumite articole să închirieze suportul magnetic cu problemele revistei. Propunem aici și contribuții personale ale cititorilor pentru îmbogățirea bibliotecii cu documentație și software.

O altă rubrică o vom numi:

3. Corespondențe și schimburi

În care vom da posibilitatea cititorilor să propună colaborări între ei și schimburi sau închirieri de suporturi magnetice, echipamente hardware sau software.

Astfel o primă propunere vine din partea redacției în atenția micilor cititori de a schimba diskete 5 1/4 cu jocuri pentru calculatoare PC sub sistemul de operare MS-DOS.

Și, în sfârșit, o ultimă rubrică este:

4. Sfaturi și servicii hardware

Unde, problemele și sugestiile privind instalarea sau depanarea hard a sistemelor de microcalculatoare poate fi soluționată prin sfaturi ale

Curier editorial

specialiștilor noștri sau prin sugestiile dumneavoastră.

În numerele următoare ale revistei vom perfecta toate detaliile privind participarea dumneavoastră în paginile curierului editorial. Oricum, abonații vor beneficia de facilități de acces, în baza carnetului de abonat ce va fi oferit de revistă.

Începînd cu nr.2, revista va putea fi obținută pe diskete 5 1/4 sau 3 1/2 la sediul redacției împreună cu programe rezolvate și compilatoare pe diskete ce pot permite cititorului să-și experimenteze singur programele propuse.

Toate condițiile de beneficiu a acestor servicii se vor perfecta începînd cu numărul 2 al revistei.

C U P R I N S

	<u>Pagina</u>
1. La început de drum..Argument	1
2. Fenomenul "Home-Computer. Calculatoare personale profesionale de la proiectare la realizare (I)	4
3. Fundamente de matematică ale programării logice în inteligența artificială (I).	17
4. Limbaje de programare în designul sistemelor de operare și al aplicațiilor. Limbajul C (I)	26
5. Baze de date relaționale și distribuite. Programare, utilizare și analiză comparativă de sistem. Limbajul SQL pentru pachetele ORACLE și Rdb (I)	31
Baze de date distribuite (I)	33
6. Teme pentru informaticieni. Prelucrarea listelor în C și Pascal (I).	37
7. Inedit, recenzii, noutăți, istoric în informatică. Din istoria calculatoarelor personale (I).	43
8. Informatică în învățămîntul liceal și universitar. Ghid de probleme rezolvate pentru elevi și studenți.	55
Probleme propuse	68
9. La granița dintre informatică și matematică. Jocuri, probleme distractive, strategii. Lexicul informaticii. Paradigma calculatoarelor.	71
O definire matematică a careurilor de cuvinte încrucișate.	75
O problemă de ordonare în teoria grafurilor.	77
10. Curier editorial	79

PRESENTATION OF OUR SOCIETY AND SERVICES

OUR SOCIETY

It was founded in January 1990 with the name mentioned above.
The president of the society is Mr. prof. **ADRIAN NEGRU**, mathematician, and with 10 years of experience in the field of microsystems, data bases and artificial intelligence.
The manager of our society is Mr. **BABIN ALEXANDRU**, vicepresident, a dipl. engineer, specialised in hardware/software for micro, mini, mainframe units.

1. STRUCTURE OF THE SOCIETY

Gaining together more than 200 specialists in mathematics, informatics, management and marketing, our society had developed the following sections of its present and future activity:

a). The editorial board. The section has basically developed its activity in two directions:

- editing informatics magazines, books and studies mainly concerning the microsystems (PC) family
- designing and implementation of all kind of facilities regarding fonts, word processing, and editing facilities.
- publishing board. We try to develop a package to control and improve the management of the publishing system.

b). The software service we had organised several groups of specialists in different branches of informatic activity offering the following services:

- installation of microsystems (PC)
- project development at request
- software testing and debugging
- customised software packages and maintenance
- lectures in programming languages, artificial intelligence, different software technics, informatic management, consulting

c). The hardware service. The hardware team consists of specialists with a large hard/soft experience in the following services:

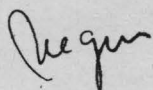
- maintenance and technical assistance for micro and mini systems
- network installation and testing for PC LAN
- consulting and diagnosis for hardware bugs

2. ACTIVITY

Our society has its personal technical resources consisting in PC, printers, documentation and software. Beside we could be found at our department at any our day or night. Even we could not fulfill your demands we could advice you. Condition of payment for us is made by invoice, according to a contract: 20% when signed, 40% during the execution of the contract, and 40% 50 days after execution.

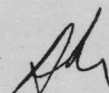
PRESEDINTE

ADRIAN NEGRU

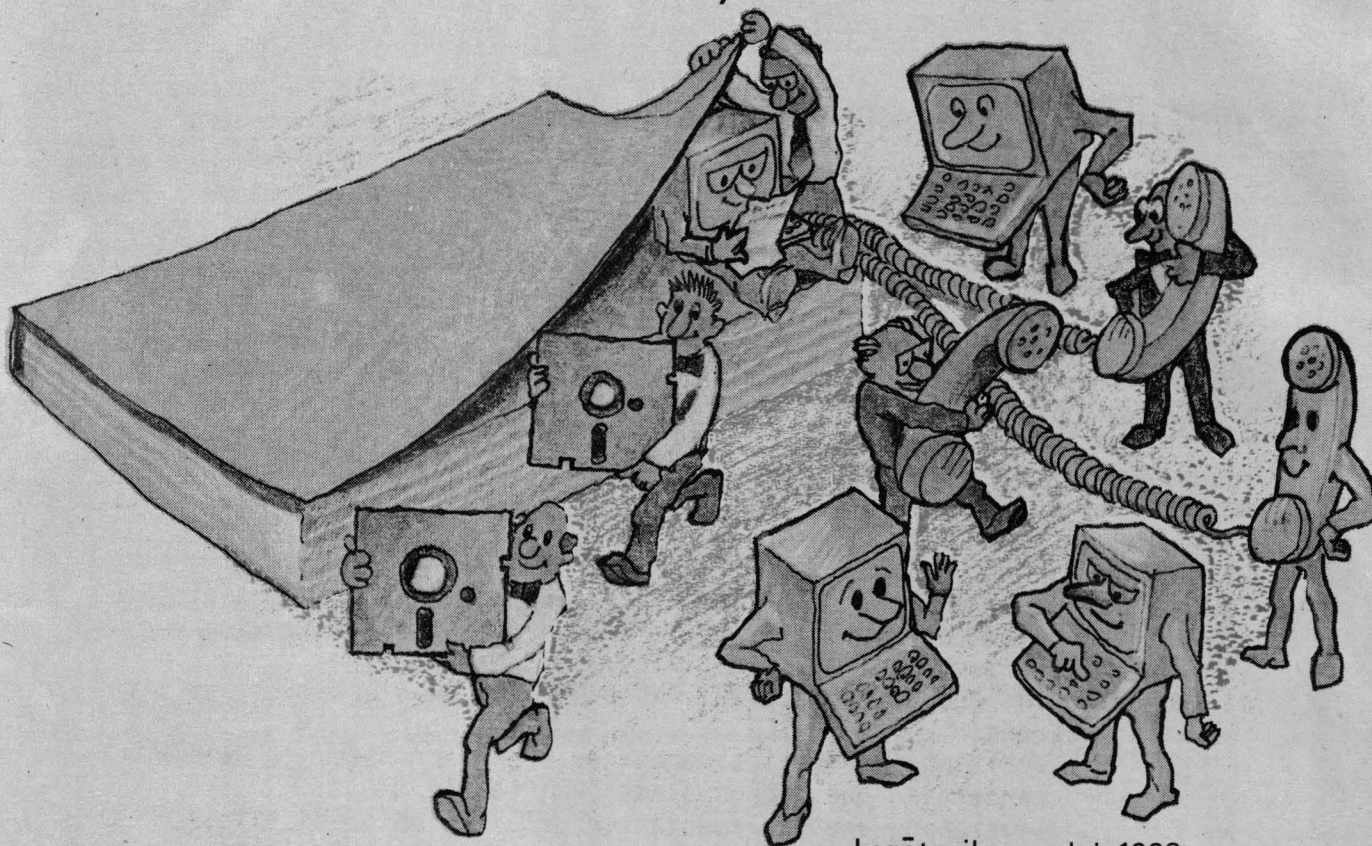


VICEPRESEDINTE

ALEXANDRU BABIN



Imaginea lumii informaționale contemporane



Legăturile anului 1990

IN ACEST NUMAR :

- LA INCEPUT DE DRUM - ARGUMENT
- FENOMENUL HOME - COMPUTER
- PROGRAMAREA LOGICA ȘI IA
- LIMBAJUL C (I)
- LIMBAJUL SQL (I)

- INEDIT - RECENZII, NOUTATI
- INFORMATICA IN INVATAMINT

- JOCURI STRATEGII
- PROBLEME (distractive) nerezolvate
- TEME DE CERCETARE
- ȘI MULTE, MULTE ALTELE

LEI 25

**Adresa noastră: București
str. Turda 114, bl. 35, sc. B, ap. 71
Telefon: 66 04 76**