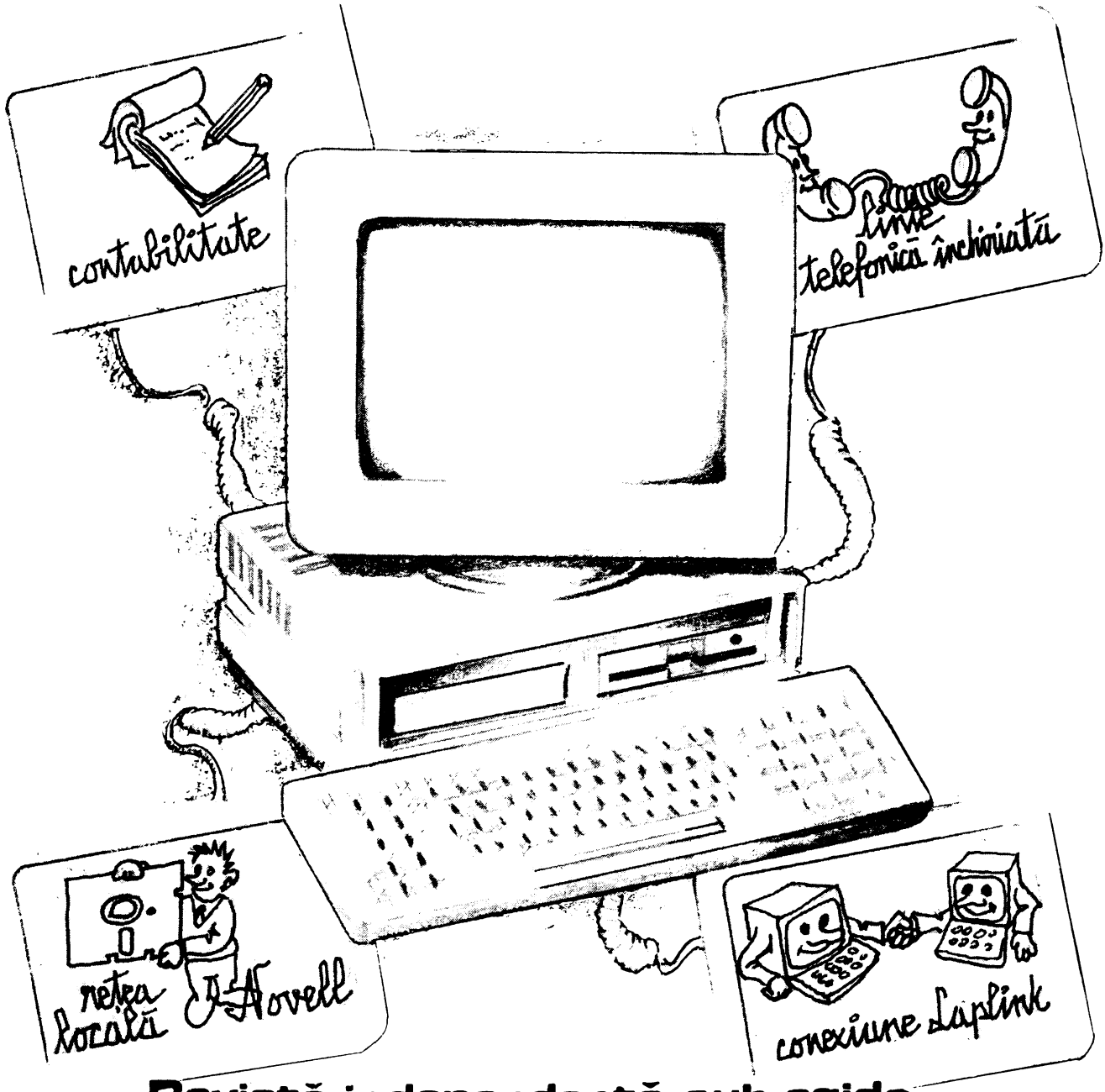
PC-Magazin

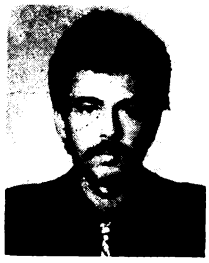
Anul I Nr. 1 Martie 1990



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

Colectivul de redacţie al revistei se prezintă

Redactor şef şi fondator al revistei: Adrian **NEGRU**



Profesia: matematician. Absolvent al Facultăţii de Matematică din cadrul Universităţii Bucureşti

Preferinţe: inteligenţa artificială, linişte sufletească

Vîrsta: 30 ani.

Redactor şef **adjunct şi fondator al revistei:** Alexandru **BABIN**

Profesia: inginer automatist. Absolvent al Institutului Politehnic Bucureşti

Preferinţe: microsisteme PC, onestitate umană

Vîrsta: 34 ani.

Redactor responsabil de număr: Vasile **AVRAM**

Profesia: economist şi cadru didactic asociat la Catedra de Cibernetică Economică din A.S.E. Absolvent al Facultăţii de Cibernetică Economică din A.S.E. Bucureşti

Preferinţe: baze de date, vînătoare sportivă

Vîrsta: 38 ani.

Redactori: Cătălin **VOLOŞENIUC**, 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 Cojocaru; Aurelian Rusu; Viorel Dumitrache; Bodosi Imre; Dragoş Rişcanu; Costel Dinescu; Silviu Bârză; Vasile Vicoleanu; Marius Sturzoiu; Adrian Goicea; Tiberiu Spircu; Magda Negoiţă; Elena Popa; Cristian Groza; Mihai Trandafirescu; Virgil Sultanovici; Garbis Ohanesian.

Dedicaţia primului număr:

"Mamelor noastre **Ioana, Silvia, Constanţa** care prin prinosul lor de durere şi renunţare ne-au dăruit un loc sub soare."

A R G U M E N T

De ce o revistă de informatică? Fără îndoială foarte mulţi o aşteptau de mult, simţeau nevoia unui suport informativ şi orientativ, alţii îşi pun problema utilităţii şi finalităţii imediat augumentative a ei.

Fără îndoială că, printre speranţe, scepticism şi entuziasm, sarcina editării acestei publicaţii este o întreprindere grea, delicată şi mai ales un act de curaj; însă, într-un moment în care informatica se instaurează ca eminenţă cenuşie a oricărei economii moderne, al oricărui proces educaţional şi social, este timpul să scoatem la lumină o ştiinţă ţinută fără logică în frîu, inhibată şi dezgolită de suportul ei aplicativ, software şi hardware, şi care a supravieţuit printr-un efort absolut eroic al adepţilor conştienţi ai beneficiului ei, ingineri, matematicieni, economişti, fizicieni, care fiecare în parte, după putinţă şi căinţă şi-au adus prinosul de cunoştinţe teoretice şi practice în ale existenţei acesteia. Fără îndoială, avem un sumum de utilaj tehnologic, unele calculatoare indigene, altele importate prin eforturi valutare incredibile, licenţe de tehnologie sau altele executate artizanal ori provenite pe diverse căi. În consens însă, marea majoritate a fluxului vital ce animă aceste unităţi metalice fără viaţă, 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 potenţial intelectual informatic al acestei ţări. Putem afirma cu o îndreptăţită mîndrie că ceea ce avem efectiv valoros în momentul de faţă în informatică este bagajul extraordinar de cunoştinţe teoretice a celor ce s-au dedicat acestei ştiinţe, o mare avere obţinută de mulţi 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 noutăţi în documentaţie sau software. Cărţi trecute din mînă în mînă, citite şi apoi comentate, pachete software pe care s-a lucrat aproape orice, reviste sau articole citite şi adaptate conştient propriilor noastre idei şi formaţii ştiinţifice. De asemenea, multe mulţumiri celor ce de-a lungul anilor au putut publica lucrări în ţară în folosul informaticii.

Neexistînd o platformă informatică consecventă în planurile de învăţămînt, dezvoltarea acestei ramuri ştiinţifice s-a făcut, funcţie de posibilităţile de dotare, de profilul economic al cerinţelor aplicative, în centre de calcul ale diverselor ramuri de activitate sau în institute specializate.

Asistăm astfel la un fenomen destul de ciudat şi anume discrepanţa dintre lăcaşurile informatice sub aspectul următor: în centre sau unităţi cu posibilităţi de dotare tehnologică prioritară şi preferenţiată, contactul şi practica cu noutăţile s-a manifestat imediat prin cunoaşterea şi lucrul cu limbaje de programare moderne şi cu pachete software la zi pe piaţa mondială, la fel ca şi practica pe calculatoare moderne de la micro profesionale PC AT pînă la sisteme VAX/VMS; în timp ce în unităţi cu disponibilităţi reduse la investiţie materială, dar cu un potenţial teoretic ridicat, s-au accentuat studiile de mare complexitate matematică sau tehnologică.

Efortul fabricilor de calculatoare pentru utilarea internă a centrelor informatice a fost şi este extraordinar, ţinînd cont de condiţiile în care au putut lucra.

Se impune astfel, acum, corelarea şi interclasarea disponibilităţilor şi cunoştiinţelor din aceste două tipuri de entităţi informatice. Această unire, impune fără îndoială şi coordonare competentă, ce însă, nu trebuie înţeleasă greşit, prin restrîngerea independenţei de acţiune creatoare şi iniţiativă a fiecărui nucleu informatic în parte. Nú trebuie repetată greşala centralismului prin crearea unei Mecca a informaticii care să dea verdictul sub formă de tehnologie şi programe fiecărui centru de calcul sau unitate de învăţămînt, îngrădindu-i astfel întreaga independenţă creativă. Să ne dorim instituirea unei alianţe ştiinţifice a tuturor forţelor informatice, prin programe realizate în comun, un flux uşor de transfer al informaţiilor la nivel naţional şi de-ce nu, a unei pieţe a concurenţei soft şi hard.

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

Apariţia acestei reviste trebuie privită ca unul din multele începuturi ale afirmării informaticii în viaţa noastră şi-şi propune a se institui ca o publicaţie generală pentru tineret, fără dorinţă de monopol publicistic. Salutară ar fi şi apariţia altor reviste de specialitate pe profil ca, de exemplu, economică informatică, statistică informatică, tehnologie informatică, orientate în probleme concrete ale activităţilor pe care le reprezintă.

De asemenea, nu va trebui niciodată considerat aceste apariţii ca o risipă inutilă de hîrtie sau efort uman, pentru că trebuie să nu uităm de acum nici o clipă că apanajul unei societăţi moderne este informaţia şi libera circulaţie a acesteia, putinţa de expresie a ideilor şi aplicare a lor.

De exemplu, Ungaria cu o populaţie echivalentă cu jumătate din a ţării noastre are circa zece publicaţii informatice la nivel naţional.

Mult timp s-a privit programarea ca un simplu proces de codare a informaţiei în limbajul înţeles de un anumit calculator. Această optică este greşită, programarea fiind un act de creaţie, fiecare program înglobînd în el cunoştinţe complexe şi efort intelectual, putîndu-se alinia creaţiilor ce se bucură de drepturi de autor şi de aceea se doreşte stimularea ei liberă. Mai mult, trebuie căutată o concepţie informatică unitară care să suplinească, cel puţin pentru moment, lipsurile tehnologice prin aplicaţii scrise clar, cu surse de program perfectabile şi supuse dezbaterii publice de specialitate.

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

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

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

real folos celor ce-o citesc. De aceea, ca structură, ea se împarte în rubrici de sine stătătoare, articole de număr, probleme propuse şi rezolvate, noutăţi, teme, recenzii, sinteze ca şi o poştă a redacţiei pentru legătura cu publicul larg cititor. Rubricile fixe ale revistei cuprind cursuri de învăţare sau perfecţionare a limbajelor moderne de programare: C, Pascal, Prolog precum şi Basic, pentru că mulţi 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 cunoaşterii în detaliu a microcalculatoarelor personale echipate cu microprocesoare din familia Intel 8086, 80186, 80286, 80386, 80486, cît şi a sistemului VAX/VMS atît sub aspect software cît şi hardware. Există rubrici destinate atît programatorilor începători cît şi avansaţi. Un spaţiu amplu este dedicat unui domeniu de avangardă al informaticii: inteligenţa artificială, prin introducerea în conceptele programării logice şi a tehnicilor ei specifice.

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

Toate expunerile se vor cît mai clare, ele adresîndu-se celor ce fac primii paşi în descifrarea tainelor acestei ştiinţe cît şi specialiştilor mai puţini iniţiaţi cu anumite domenii informatice.

Încercarea noastră este ca paginile revistei să suscite interesul unei mase largi de cititori, făcînd apel ca toţi cei ce au sugestii de îmbunătăţire a formei sau a conţinutului, s-o facă, aşteptînd articole, probleme şi curiozităţi de la toţi cei ce doresc să-şi aducă aportul publicistic şi profesional în paginile acestei publicaţii. De asemenea, poşta redacţiei se va institui şi ca mijloc de comunicare între toţi cei interesaţi în schimburi de idei, software sau documentaţie.

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

Cu speranţa ca în acest an al începutului libertăţii noastre revista să-şi aducă prinosul de bine întru împlinirea noastră, ne punem cu acest număr de avangardă în consensul general de renaştere naţională.

Adrian NEGRU

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

Elemente de arhitectură a unui microsistem profesional

1. Particularităţi constructive ale unui microcalculator

Înţelegem modul de funcţionare a familiei de microcalculatoare personale, profesionale dacă putem investiga arhitectura sa intimă şi mai ales dacă cunoaştem microprocesorul ce acţionează ca un adevărat creier al computerului. A fost aleasă familia microprocesoarelor 8086, care introduce conceptul de calcul pe 16 biţi şi mai nou pe 32 biţi (80386), ceea ce înseamnă că acest calculator poate lucra cu 16 biţi sau 32 biţi în acelaşi timp.

Cele cinci părţi cheie ale unui microcalculator sînt: **procesorul, memoria, sistemul de intrări/ieşiri (I/O), suportul magnetic disc şi programele.** O scurtă imagine introspectivă asupra funcţionării 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 instrucţiunile programelor sistemului. Dacă el este cel care este capabil să efectueze calcule matematice sau operaţii 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 activităţile care se petrec la un moment dat în calculator. Eficienţa unui sistem este dictată atît de calitatea procesorului cît şi de cantitatea de memorie internă pusă la dispoziţie. Majoritatea microcalculatoarelor din familia PC dispun de o memorie internă de 640 KO, modelele constructive mai noi mărindu-şi această capacitate pînă la 1 MO sau 16 MO. Sistemul de intrări/ieşiri este modul în care calculatorul primeşte sau exportă date. De fiecare dată, cînd introducem sau exportăm aceste date, calculatorul apelează la serviciile unităţilor periferice, dintre care: tastatura, imprimanta, linia de comunicaţie asincronă, discurile.

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

Există însă şi alte suporturi magnetice (banda, caseta) dar discul este cel mai eficient şi important. În ultimii ani firmele constructoare de suporturi magnetice disc s-au preocupat tot mai mult de găsirea unor soluţii tehnologice care să mărească capacitatea de stocare a informaţiei 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 convenţionale; în timp ce noile discuri optice exploatate atît în scriere cît şi 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 viaţă. Instrucţiunile 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

aplicaţii şi programele de sistem.

O parte din programele de sistem de care microcalculatorul are nevoie în permanenţă sînt păstrate permanent în memoria calculatorului, definind aşa numitele programe **ROM** (Read-Only-Memory). Aceste tipuri de programe supervizează şi controlează munca calculatorului punînd la dispoziţie 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 secţiune o să trecem în revistă formatul fundamental al datelor pe care le foloseşte un microcalculator. Unitatea fundamentală de stocaj a datelor în calculator este bitul. În majoritatea microcalculatoarelor 8 biţi sînt combinaţi pentru a forma un octet, fiecare bit în octet putînd lua 2 valori, 0 sau 1, permiţînd astfel 256 combinaţii a celor 8 biţi. Astfel, cele 256 simboluri diferite ce pot fi exprimate într-un octet pot reprezenta valoarea unui caracter al setului ASCII sau o valoare întreagă între 0 şi 256. De obicei, sîntem obişnuiţi 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 conţinutul unui octet este exprimat în binar avem nevoie de 8 cifre (0 sau 1), fiecare cifră corespunzînd unui anumit bit, biţii fiind numerotaţi de la 0 la 7. Modul de evaluare al acestor biţi 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 numeşte bit inferior iar bitul 7 - bit superior).

De ce este nevoie de această reprezentare binară? O motivaţie rezidă în faptul că sistemul păstrează informaţii în octeţi de stare în memorie sau regiştri de stare pe cipurile plăcilor sistemului. Anumite informaţii se pot găsi în secvenţe particulare de biţi în cadrul unui octet, astfel, într-un octet de stare al periferiicelor biţii 6 şi 7 dau informaţii asupra numărului logic al imprimantei în timp ce biţii 0 şi 1 specifică numărul logic al unităţii de disc.

Un grup de biţi luaţi 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 precedenta. Utilitatea numerelor hexazecimale stă în faptul că o cifră hexa descrie conţinutul unei jumătăţi de octet; astfel, numărul hexazecimal F6 are echivalentul binar 1111 pentru cifra F şi 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 importanţă deosebită într-un calculator. Multe pachete software integrate ca Lotus, Word Star, Multimate, Tex sînt orientate pentru lucrul pe şiruri de caractere. Fiecare caracter ocupă cei 8 biţi ai unui octet ele fiind standardizate de codul ASCII împărţit în două părţi: codul standard reprezentînd caractere cu valoarea între 0 şi 127 şi codul extins cu caractere în gama de valori 128-255. Nu toate caracterele ASCII sînt tipăribile, o parte a lor avînd exclusiv funcţii de control şi anume cele care poartă codul între 0 şi 31.

O tabelă a acestor caractere este prezentată în capitolul dedicat tastaturii, caracterele în codul 1-4 fiind folosite în probleme de comunicaţie, cele din gama 28-31 fiind marcatori ai limitelor de fişier 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 părţi. Prima parte este numită "Unitate centrală de sistem" în ea fiind încorporate principalele unităţi funcţionale ale calculatorului. La dispoziţia utilizatorului este pusă o unitate fizică numită tastatură, în timp ce dialogul între sistem şi programator se face pe ecranul grafic al unui monitor. Alimentarea sistemului se face convertind curentul alternativ în curent continuu, 4 voltaje diferite 12 volţi; - 12 volţi; + 5 volţi; - 5 volţi. Modelelor mai vechi de microcalculatoare li se atribuia o putere de 65 W ulterior aceasta ajungînd la 130 W pentru modelele XT şi 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 mişcare calculatorul. Astfel, aici apar microprocesorul, ceasul de sistem care acţionează ca un metronom de coordonare a activităţii, cipurile de memorie şi 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 Unităţii de sistem. Urmează apoi conectori opţionali (interfeţe) şi conectorii de bus, ceea ce conferă deschiderea calculatorului spre "lumea" exterioară. Toate cipurile siliconice ca şi celelalte părţi ale calculatorului sînt conectate între ele astfel încît îşi pot transmite reciproc semnale şi date. Constructiv, aceasta se realizează printr-un canal de comunicaţie comun alcătuit dintr-un număr de fire prin care datele trec simultan numit şi bus. Busul unui microcalculator are 62 de linii separate, identificate prin codurile A1-A31 şi B1-B31, conectorii A fiind situaţi 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 circulaţia biţilor 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 şi anume: o adresă de memorie şi una pentru porturile de intrări/ieşiri. Liniile care au rămas sînt utilizate pentru diverse secvenţe 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 comunicaţie serială COM1 şi COM2. Modele constructive mai noi din gama AT mai adaugă busului încă 36 de linii numerotate C1-C18 şi D1-D18, 8 dintre aceste linii sînt afectate datelor, 8 linii de adresă şi 5 serviciilor de întrerupere.

Adaptoarele opţionale ca cipul video şi 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 compatibilităţii cu procesoarele 8088 şi 8086; mai concret, 286 lucrează cu un bus de memorie de 16 biţi ceea ce-l apropie de 8086 şi nu de 8088 care este echipat cu un bus de 8 biţi dar cu care se aseamănă în modul de încărcare şi prelucrare a programelor.

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

Să stabilim ce înseamnă exact, în vorbirea curentă, o frază ca aceea "acest microcalculator foloseşte un ceas de 8 Mhz". Toate acţiunile unui mp sînt gestionate de un ceas ce acţionînd ca un metronom corelează şi coordonează contactul mp cu mediul extern, în sensul măsurării temporale exacte a fiecărei operaţii executate. Măsura unui ceas de sistem o înţelegem prin aceea că un ceas de 8 Mhz "ticăie" de 8 milioane de ori pe secundă. Astfel, cu cît frecvenţa unui ceas este mai mare cu atît mai rapid este un mp în execuţie. Pe un 286 o operaţie de înmulţire durează circa 20 de cicli în timp ce pe 8088 aceeaşi operaţie 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. Şi totul nu se opreşte 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.

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

Protecţia apare ca o noutate şi ca o cheie de siguranţă în arhitectura mp eliminînd situaţiile în care un program cu erori putea bloca sistemul sau putea distruge zone de memorie ce nu-i aparţin.

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

Modul protejat prin posibilitatea de gestionare a unei memorii extinse şi prin mecanismul de memorie virtuală permit instalarea în sistem de pînă la 16 megabytes de memorie şi a unui spaţiu de adresă accesibil unui program de pînă la 1 gigabyte (echivalentul a peste 1 billion de octeţi). 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 îşi poate instala un disc virtual de mărime dorită. Avantajul unei astfel de memorie virtuală constă în aceea că ea se prezintă ca un spaţiu continuu (sau contiguu) de octeţi accesibili direct de un program, adresele fiecărui octet fiind adrese gestionate de program, adică adrese logice şi nicidecum adrese fizice, fixe în spaţiul 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) şi RAMDRIVE.EXE (3.2X..) ce nu sînt altceva decît nişte programe ce simulează ca suport disc o zonă de memorie, de unde şi denumirea de disc virtual dată procedeului de vizualizare.

Zona se recunoaşte printr-un nume dat ei; de obicei D:, E:, numele simbolice A:, B: fiind rezervate pentru unităţile fizice de floppy-discuri, iar C: unităţii de hard-disk sau disk rigid. Dimensionarea şi 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 octeţi de sector.

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

Microprocesorul INTEL 80286 are o adresă de 24 biţi, o interfaţă de memorie de 16 biţi, un set de instrucţiuni extinse, DMA, suport hardware de înmulţire şi împărţire în virgulă fixă, un manager de memorie integrat, 4 nivele de protecţie a memoriei, un gigabyte (1.073.741.824 octeţi) de spaţiu virtual de adresă pentru fiecare task şi 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 biţi. Primii 16 biţi de adresă formează adresa selectorului sau a segmentului de memorie adresat, ultimii 4 biţi ai adresei de segment pe 20 biţi 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 şi pot fi accesate în citire sau scriere. O excepţie (derivă) sau o întrerupere poate avea loc dacă datele sau instrucţiunile unui program depăşesc zona limită de 64 Kbyte.

Dacă informaţia conţinută într-un segment de memorie nu foloseşte toţi 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 biţi format dintr-un selector pe 16 biţi şi componenta de deplasare. Selectorul, în acest caz, desemnează un index într-o tabelă de memorie rezidentă şi nu primii 16 biţi ai unei adrese de memorie reală. Adresa de bază pe 24 biţi a segmentului dorit este obţinută din tabele în memorie.

Deplasamentul de 16 biţi este adăugat adresei de bază a segmentului pentru a forma adresa fizică. Ca observaţii: tabelele de memorie conţin valori pe 8 octeţi numiţi descriptori.

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

Un ciclu de bus pe 16 biţi are nevoie de 3 cicli de ceas şi după fiecare dă 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 biţi care are nevoie de 6 cicli de ceas şi de 4 stări de wait cca 1000 u.s.; iar un ciclu de bus pe 16 biţi 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, I, 2, 3 sînt folosite pentru transferare de date pe 8 biţi, iar canalele 5, 6, 7 procesează transferul de biţi. Canalul 4 este folosit pentru accesul canalelor 0 şi 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 secvenţa ESCAPE, celelalte două moduri fiind comunicarea cu memoria şi folosirea porturilor. Deci memoria este locul unde microprocesorul îşi găseşte instrucţiunile unui program la fel ca şi datele. Atît datele cît şi instrucţiunile sînt stocate în memorie de unde microprocesorul le ia şi le prelucrează intern. Deschiderea acestuia spre exterior se face la nivelul porturilor, acestea

acţionînd 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 foloseşte ca pe un număr de telefon pentru a accesa partea de care are nevoie.

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

Pentru microprocesoarele Intel 80386 şi 80486 aceste registre sînt organizate pe 32 biţi (4 octeţi) şi denumite EAX, EBX, ECX, EDX. Fiecare dintre ele este utilizat de către program ca zonă temporară de stocaj, utilizatorul putînd accesa şi jumătăţi ale lor (cîte un octet) denumite, în cazul lui AX de exemplu, AH (octetul superior, biţii 8-15) şi AL (octetul inferior, biţii 0-7). La fel, BX se împarte în BH şi BL, CX în CH şi CL, DX în DH şi DL. În cazul microprocesoarelor 80386 şi 80486 EAH şi EAL au cîte 16 biţi. Următorul grup de regiştri ajută microprocesorul în adresarea exactă a zonei de memorie cu care se lucrează. Aceştia sînt cunoscuţi ca regiştrii de segment fiecare permiţînd accesul la un segment de memorie ce măsoară 64 k octeţi 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 foloseşte programul, în timp ce registrul EI suplimentează segmentul de date. Registrul de stivă, notat SS, localizează segmentul stivă din memorie. Ultimul grup de 5 regiştri permit localizarea în memorie a anumitor octeţi de date dorite, utilizarea lor împreună cu regiştrii de segment permiţînd localizarea exactă a datelor din memorie.

Pointerul de instrucţiune sau registrul EP acţionează ca un contoar al instrucţiunilor unui program informînd microprocesorul despre instrucţiunea ce se execută la un moment dat.

Pointerul de stivă, notat SP, şi pointerul de bază, notat BP, sînt regiştrii care ţin evidenţa datelor care sînt stocate în stivă. Indexul de sursă, notat SI, şi index destinaţie, notat DI, sînt ultimii doi regiştrii ce permit programelor manipularea unor cantităţi mari de date dintr-un loc în altul. Un ultim registru, numit de semafoare, informează programul prin configuraţia biţilor din el despre starea calculatorului la un moment dat (rezultatul unei operaţii aritmetice, semnalul depăşirii aritmetice, permisul de activare a unei întreruperi, paritate etc.).

Stiva serveşte ca zonă de păstrare a informaţiilor despre activitatea calculatorului. Deci apelul unei subrutine este semnalizat în stivă de informaţii care spun unde a fost întrerupt programul în lucru pentru a fi reluat după execuţia rutinei apelate.

5. Întreruperi

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

Cele hardware sînt iniţiate de componentele fizice ale sistemului propriu de pe placa de sistem sau de la canalul de intrări/ieşiri. Ele sînt cunoscute ca întreruperi bios (apăsarea unei chei de la tastatură, semnal de la imprimantă), ele nefiind corelate cu activităţile software. La semnalarea unei întreruperi valoarea adresei CS:IP este salvată în stivă la fel ca şi registrul de semafoare; după care adresa din memorie a rutinei de întrerupere este încărcată în CS:IP şi i se predă controlul. Rutinele întreruperilor mai sînt cunoscute şi ca HANDLERE de întreruperi, ele terminîndu-se întotdeauna cu instrucţiunea 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 esenţial 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 şi vectori. Fiecare vector are lungimea de 4 octeţi. Primii doi octeţi păstrează valoarea lui IP iar ceilalţi doi a lui CS.

Cei mai de jos 1024 octeţi de memorie conţin 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 şi 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 dispoziţie un nivel de priorităţi al acestuia. Există opt nivele de priorităţi cu excepţia maşinilor 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 regiştrii de un octet care controlează şi dictează cele 8 linii de întrerupere hardware. Registrul de servici 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 informaţie.

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

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

Cele 16 nivele de întreruperi de sistem sînt generate de două controlere de întrerupere 825S A şi de NMI mp. Toate întreruperile pot fi mascate, inclusiv NMI.

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

Subsistemul RAM în cei 16 mega în spaţiul 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 şi un ciclu durează 275 us. Programul de iniţializare a memoriei RAM procesează în felul următor:

- iniţializează canalul 1 ca timer cu o perioadă de 15 microsecunde;
- procesează o operaţie de scriere a memoriei la nivelul oricărei locaţii de memorie.

6. Memoria

Memoria calculatorului este zona unde se stochează informaţia care include atît instrucţiunile de program cît şi datele avînd ca unitate constructivă de stocaj şi de măsură octetul. Orice octet poate lua 256 valori distincte, biţii care îl organizează putînd reprezenta un număr, codul unei litere din alfabet sau o instrucţiune particulară de maşină, funcţie de modul de interpretare a semnificaţiei. Octeţii din memorie se organizează astfel încît să creeze unităţi de informaţie specifice, cum ar fi cuvîntul organizat pe doi octeţi consecutivi sau un şir de caractere reprezentaţi ca o succesiune de octeţi ce interpretează un text. Utilizatorul are acces la memoria calculatorului printr-o adresă ce este un număr asociat fiecărui octet.

Dacă un şir 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 şi formulelor mai mari de doi octeţi şi anume pentru cuvinte duble reprezentate pe 32 biţi.

Dacă numerele sînt reprezentate în format hexazecimal în memorie, reprezentarea se face nu prin inversarea fiecărei cifre hexa ci prin inversarea octeţilor; astfel numărul A0F1 se reprezintă în memorie sub forma F1A0. Modul de adresare al memoriei a fost standardizat prin definirea aşa numitelor segmente, unităţi de 64 k ce pot fi accesate prin aşa numitele adrese de segment care nu sînt altceva decît cuvinte pe 16 biţi. Aceste adrese sînt astfel combinate încît pot accesa 1048576 octeţi de memorie în modul de lucru real. Vom înţelege prin spaţiul 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 biţi din care una reprezintă adresa de segment, iar cealaltă deplasamentul în cadrul segmentului. Să presupunem că adresa începutului

de segment este ABCD şi vrem să aflăm adresa completă a octetului aflat la 1234 de octeţi 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 obţinîndu-se adresa efectivă:

```
  ABCD0
+ 01234
-------
  ACF04
```

Un alt mod de notare condensat a acestei adrese se face: ABCD:1234 ea constituind de fapt şi adresa absolută. Indiferent dacă este vorba despre cod de program sau dacă acestea sînt memorate în cadrul aceluiaşi 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 locaţii de memorie în cadrul segmentului se poate păstra în registrele generale sau în registrele de index. Dacă în primele tipuri de microcalculatoare datorită restricţiilor tehnologice, compilatoarele Basic, Fortran, Pascal impuneau existenţa datelor şi a codului de program (instrucţiuni) în limita strictă a unui segment, 64 k, tipurile constructive mai noi permit organizarea memoriei în aşa numitele modele. Modelul "slave" cuprinde datele şi codul în 64 k, modelul "medium" păstrează datele şi codul în segmente distincte de 64 k, modelul "large" păstrează datele pe un segment şi codul poate să fie mai mare decît un segment în timp ce modelul "huge" permite ca atît datele cît şi codul să depăşească respectiv 64 k. Numai ultimul model permite definirea de şiruri de caractere şi vectori mai mari de 64 k. Fiecare segment este aliniat la o adresă multiplu de 16 numită adresă de paragraf.

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

Memoria RAM este aşa numita memorie volatilă în care sînt încărcate datele şi programele care permit calculatorului să funcţioneze şi care este ştearsă odată cu închiderea sau scoaterea de sub tensiune a calculatorului. Nu tot RAM-ul este la dispoziţia utilizatorului, o parte a sa fiind ocupată de sistemul de operare, servicii I/O, rutine de întrerupere rezidente şi 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 octeţi ce raportează numărul de k de memorie utilizabilă. Pe maşinile AT există un cip care informează în registrele 15H şi 16H asupra cantităţii 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 şi 31H în unităţi de 512 k pînă la 15 MO. Întreruperea BIOS 21H întoarce în AX numărul de KO din sistem pe care îi citeşte din registrele cipului 8255 amintite mai sus.

Pe microcalculatoarele AT funcţia 88H a întreruperii 15H verifică extensia de memorie din afara spaţiului real de adresă, adică existenţa 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 şi funcţionare a memoriei extinse (de peste un MO). Este formată din trei părţi, cipurile de memorie ce se instalează opţional pe placa sistem împreună cu două programe software de gestiune a ei, managerul de memorie extinsă (EMM) şi un program de aplicaţie care utilizează această memorie. La startarea calculatorului EMM este activat şi pregăteşte terenul pentru operaţiile cu memoria extinsă. Sarcina sa principală fiind să găsească în spaţiul de memorie a calculatorului o zonă nefolosită (64 k) în care el depune informaţii despre această memorie suplimentară. În continuare EMM împarte acest cadru de 64 k în 16 ferestre fiind gata de a asigura orice program de aplicaţie care ştie să îl folosească schimbul cu date din memorie în interiorul şi în afara ferestrelor. Limitarea acestei tehnici stă în aceea că schimbul se poate face numai cu date şi 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 atît programele noastre de aplicaţie cît şi programele de sistem care pun calculatorul în funcţiune. Aceste unităţi magnetice ce funcţionează ca unităţi permanente de memorie sînt unităţile de disc. Două probleme se impun a fi cunoscute şi anume tehnologia de înregistrare şi metoda de acces rapid la disc. Tehnologia are la bază înregistrarea magnetică. Aceeaşi metodă de înregistrare a benzilor magnetice sau a casetelor video.

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

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

Capul de citire al discului se mişcă liniar deasupra mesei rotative pentru o disketă fiind necesar 1/6 dintr-o secundă pentru a ajunge la locaţia dorită şi 1/25 dintr-o secundă la hard disc. Combinaţia dintre mişcarea circulară a suprafeţei discului şi cea a capului de disc determină accesul foarte rapid la informaţie numit şi acces random sau nedefinit datorită posibilităţii de găsire în orice moment a oricărei informaţii la disc spre deosebire de bandă unde localizarea unei informaţii presupune desfăşurarea benzii pînă la acel loc (acces secvenţial). Ca organizare, suprafaţa discului este împărţită 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 şi 600 piste fiecare fiind identificată printr-un număr (0 este numărul pistei cea mai exterioară).

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

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 împărţită în sectoare care pot fi în număr de 8 sau 9 în mod normal, 15 pentru HI - densitate şi 17 pentru discurile rigide. La rîndul ei, mărimea unui sector este variabilă între 128 octeţi şi 1024 octeţi 512 octeţi fiind standardizată. Citirea sau scrierea discului se face în unităţi de sector fiecare sector al unei piste fiind numerotat începînd cu cifra unu sectorul numărul zero fiind rezervat pentru identificare şi nu pentru stocaj. Ultima problemă tehnică este cea a numărului de feţe înregistrabile a unui disc, floppy discurile putînd fi înregistrate pe o faţă sau pe amîndouă în timp ce discurile rigide conţin mai multe platane pe un suport, deci pot avea un număr variabil de feţe. Deci, capacitatea de memorare a discului se obţine înmulţind numărul de octeţi dintr-un sector cu numărul de sectoare de pe o pistă şi cu numărul de piste de pe o faţă şi cu numărul de feţe.

Ca organizare constructivă o disketă are 2 orificii unul central care se fixează pe unitatea de disc şi altul ce se constituie ca un punct de referinţă la începutul unei piste. O cămaşă dintr-un material special protejează disketa permiţînd şi rotirea ei în timp ce o deschidere ovală în acest material permite capului de disc citirea sau scrierea pe o discketă. Pe o parte a cămăşii 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 secţiunii vom dedica o rubrică informativă specială asupra noilor tehnologii de discuri şi unităţi de memorare (discuri optice, unităţi laser, acces Bernoulli). Modul de organizare a unei diskete se poate face cu programul Format (sectorizare software) care specifică numărul de părţi, piste, sectoare ce se vor ştanţa pe disc permiţînd, 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 şi 20 MO, 30 MO, existînd în noile tehnologii hard discuri încorporate pe unităţi detaşabile calculatorului, cu capacităţi între 60-400 MO.

Discurile de 10 MO au 305 cilindri (denumirea echivalentă a unei piste) şi 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 înţelegerea tehnicii de lucru şi organizarea discului din perspectiva acestui sistem.

Astfel DOS divide discul în 2 părţi: o zonă în care sînt păstrate informaţiile despre disc şi o zonă de memorare a datelor. Zona de sistem a DOS-ului, care ocupă 2% din suprafaţa totală, se împarte, la rîndul ei, în 3 părţi: înregistrarea de încărcare (boot), zona de alocare a fişierelor (FAT), directorul rădăcină. Zona de boot conţine 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 existenţa pe disc a 2 fişiere IBMBIO.COM şi IBMDOS.COM care informează dacă unitatea în uz este de disc sistem sau nu. De asemenea, programul conţine o tabelă de elemente de organizare a discului precum

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

Pentru managementul spaţiului de date disc sistemul DOS îl divide în unităţi logice numite clustere. Cînd un fişier este înregistrat pe o porţiune de disc acest spaţiu disc este asignat (atribuit) fişierului în aceste clustere. Mărimea unui astfel de cluster variază de la un format de disc la altul, cu lungimi între 1 sector şi întreg spaţiul disc.

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

Alocarea clusterelor este sarcina FAT-ului care ţine evidenţa 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 secvenţială indică şi legătura dintre mai multe clustere dedicate aceleiaşi zone de date fişier.

Valoarea primului număr identificator de cluster este 2 valorile 0 şi 1 fiind rezervate de DOS. Intrările în FAT sînt numere pe 12 biţi putînd acomoda 4 k numere, iar pentru AT acesta este mărit pentru a ţine evidenţa a 10000 de clustere (numere pe 16 biţi). Pentru orice fişier intrarea în directorul său indică numărul primului cluster alocat acestuia fiecare alt cluster conţinînd numărul următorului din lanţul celor alocate fişierului sau marcatorul de sfîrşit de lanţ (sfîrşit de fişier) care este FFF (pentru FAT-uri cu intrare pe 12 biţi) sau FFFF (pentru FAT-uri pe 16 biţi). Porţiunile din disc inutilizabile (sectoare stricate) sînt identificate în FAT prin valorile FF7 (sau FFF7). Codurile numerice între FF0 şi FFF (sau FFF0 şi FFFF) sînt rezervate pentru utilizări viitoare.

Metoda codificării numerelor pe 12 biţi în 3 octeţi 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 conţine identificatorul formatului de disc (codul FE identifică o disketă simplă densitate de 160 k) sistemul DOS ţinînd întotdeauna, pentru siguranţă, două copii ale tabelei FAT.

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

Pentru fiecare fişier există o intrare în director (poziţie) care conţine numele fişierului pe 8 caractere, extensia acestuia pe 3 caractere, mărimea în octeţi a fişierului, precum şi data ultimei modificări (creări) a fişierului. Instrucţiunea DOS ">DIR" listează toate fişierelor înregistrate în director. Mai există încă 2 informaţii înregistrate în director pentru fiecare fişier şi anume: numărul clusterului de început a datelor din fişier şi atributul de fişier (putem avea următoarele categorii de fişiere: fişiere sistem; ascunse; exploatabile doar în citire; normale; arhivate).

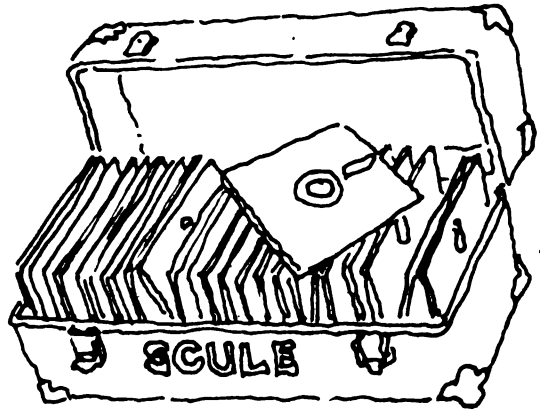
Orice intrare într-un fişier are 32 de biţi mărimea directorului rădăcină variind în funcţie de disc astfel: 4 sectoare pentru disketă simplă densitate, 7 sectoare pentru o disketă dublă densitate, 32 de sectoare pentru un disc de 20 MO (acomodînd, astfel, 512 intrări de fişier).

Subdirectorii sînt o mixtură între fişier şi directorul rădăcină. Astfel,

evidența oricărui subdirector este ținută de FAT în tabela pentru un fișier oarecare. În schimb, atunci cînd 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 atunci 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 inteligenţa artificială (I).

Elemente de calculul predicatelor în inteligenţa artificială

1. Relaţii bine definite

Programarea logică în inteligenţa artificială comporţă un formalism calculatoriu, avînd la bază trei elemente constitutive ce formează aşa numitul **sistem de producţie** al IA(*) şi anume: regulile de producţie sau combinaţionale, o bază globală de date şi un sistem de control.

Aceste trei mari entităţi acţionează global asupra unui produs al IA în sensul că regulile acţionează asupra bazei de date fiecare fiind realizată sau nu în bază, de satisfacerea lor depinzînd condiţiile de aplicabilitate ale regulei respective, condiţii dictate de sistemul de control, care dictează atît începutul cît şi sfîrşitul unei sesiuni de lucru cu sistemul, în funcţie de satisfacerea unor reguli de început şi sfîrşit impuse. Ceea ce este particular oricărui sistem al IA constă în posibilitatea de acces la baza de date a oricărei reguli de producţie, fără restricţii, aceste reguli nefiind nici apeluri tip subrutină la alte baze de date, ele acţionînd strict în cadrul bazei sistemului, adăugarea de fapte la elementele bazei putîndu-se face în cadrul acţiunii regulilor prin aşa numitele "aserţiuni de fapte obţinute prin instantări" ale regulilor.

Astfel rezolvarea unei probleme de IA folosind un sistem de producţie presupune specificarea bazei de date, a regulilor şi 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, informaţiile 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 entităţilor reclamînd capacitatea sistemului de manipulare a unor mulţimi logice de afirmaţii sau fapte.

Astfel, aspectele fundamentale ale formalismului logic, ca reprezentările unor aserţii, deducerea inferenţelor într-o mulţime de expresii, stabilirea de dezvoltări recursive ale elementelor unui limbaj sînt realizate pe două nivele şi anume: i) nivelul individual al programării logice sau calculul predicatelor de ordin unu şi ii) nivelul doi sau al predicatelor de ordin II sau de mulţimi, ce se ocupă cu studiul global al mulţimilor şi proprietăţilor lor, incluzînd calculul lambda şi variabile predicative, ce permite tratarea funcţiilor şi a relaţiilor 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 evoluţia firească a calculului, deci pornind cu analiza predicatelor de ordin unu, precum şi a limbajului şi metodelor logice generate de ele, ocupîndu-ne de folosirea lor în sistemele de producţie 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 înţelege conceptul de inteligenţă artificială.

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

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

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

(1) predicat (termen1, termen2,... termen n)

expresia (1) constituind un **rbd** în calculul predicatelor dacă ei i se dă o interpretare prin stabilirea unei corespondenţe între elementele constitutive ale lui (1), ce sînt şi părţi ale limbajului predicativ şi relaţiile, entităţile, funcţiile 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 aplicaţii ne situăm într-un anumit domeniu al limbajului curent, cunoscînd din logica îmbinării frazelor şi un istoric al entităţilor folosite, deci ştim, 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 entităţilor părţii de cunoaştere afectată aplicaţiei.

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 şi alta din domeniul relaţiilor unui arbore genealogic. De exemplu faptul că există un drum între BUCUREŞTI şi BRAŞOV poate fi reprezentat prin:

(2) drum (BUCUREŞTI, BRAŞOV, 173)

al treilea termen al relaţiei reprezentînd şi distanţa dintre termen 1 = Bucureşti şi termen 2 = Braşov.

Aici atît Bucureşti, Braşov cît şi 173 sînt simboluri constante. Formula (2) poate fi cuantificată şi sub forma:

(2') drum (Oraş, Oraş, Distanţă)

unde Oraş este simbol variabil, iar Distanţă este un întreg variabil sau mai formal:

drum (X,X,D)

cu X = simbol şi 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 urmaşul altei persoane".

Avem posibile construcţii:

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

părinte (Ion, Dan);

părinte (Dan, Monica);

înţelegînd 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).

înţelegînd că predicatul căsătorit (X,Y) are sensul: "X este căsătorit cu Y" şi la fel "Y este căsătorit cu X", ambele constituindu-se ca **rbd** în domeniul discursului. Se impune în ambele predicate, distincţia $X \neq Y$, altfel asistăm la ambiguităţi logice, formulele părinte (X,X) sau căsătorit (X,X) nefiind logice din punctul de vedere al realităţii bunului simţ. Astfel de formule atomice nu mai sînt **rbd**, valoarea lor de adevăr fiind F (fals). Astfel, căsătorit (Dan, Dan) are valoarea fals nefiind deci un **rbd**.

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

drum (Bucureşti, Braşov, 5000)

este o relaţie falsă, deci nu este o **rbd**, bunul simţ dictîndu-ne imposibilitatea parcurgerii distanţei de 5000 km între Bucureşti şi Braşov (o astfel de relaţie poate fi o **rbd** deci o formulă adevărată, într-un domeniu al excentricităţii, de exemplu, deci universul realităţii curente joacă rol esenţial în sfera sistemelor de producţie ale IA alese).

Formulele atomice pot avea ca termeni constitutivi şi simboluri funcţii sau, aşa cum le defineam, concatenări de predicate, ca de exemplu:

$$\text{căsătorit}\big(\text{părinte}(X,\text{Maria}),\text{părinte}(Y,\text{Maria})\big)$$

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

bărbat (Ion), femeie (Ioana).

Astfel căsătorit $\big($părinte (X, Maria), părinte (Y, Maria)$\big)$ devine un **rbd** sub restricţiile bărbat (X), femeie (Y).

2. Calculul propoziţional şi cuantificare

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

Apare atunci necesitatea naturală de compunere a formulelor atomice, care privite ca entităţi binare pot fi false sau adevărate. Astfel, sarcina compunerii formulelor atomice devine uşoară în sensul că fiind date 2 formule f_1 şi f_2, $f_1 o f_2$

poate fi o nouă formulă atomică în care simbolul "o" al compunerii ia valori în mulţimea operanzilor din calculul propoziţional astfel:
"o" $\in \{ \neg, \vee, \Rightarrow, \wedge, \Longleftrightarrow, \}$.

Dacă "o" = "$\vee$", atunci $f_1 \vee f_2$ se numeşte disjuncţia formulelor f_1 şi f_2, avînd proprietatea că, dacă f_1 şi f_2 sînt **rbd**-uri atunci $f_1 \vee f_2$ este un **rbd** (relaţie bine definită). Dacă "o" = "$\wedge$", atunci $f_1 \wedge f_2$ se numeşte conjuncţia formulelor f_1 şi f_2 cu proprietatea că dacă f_1 şi f_2 sînt **rbd**-uri atunci şi $f_1 \wedge f_2$ este un **rbd**.

Dacă "o" = " ", 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 numeşte implicaţie cu proprietatea că pentru f_1, f_2 **rbd**-uri şi $f_1 \Rightarrow f_2$ este un **rbd**. Vom vedea că, datorită valorii de adevăr a operatorului "$\neg$" putem genera **rbd** cu valori de adevăr absurde, dacă implicaţia are loc în domenii disjuncte ale vorbirii.

Considerăm utilă o reamintire a valorilor de adevăr generate de operatorii propoziţionali aplicaţi la două formule f_1 şi 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

$i=1,2$

Tabela "$\Rightarrow$":

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

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

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

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

unde M_{rbd} este mulţimea rbd-urilor sistemului folosit. Atunci avem proprietăţile 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 implicaţia are loc în domenii disjuncte ale vorbirii.

Considerăm utilă o reamintire a valorilor de adevăr generate de operatorii propoziţionali aplicaţi la două formule f_1 şi 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 ţinînd cont de tabelele de adevăr precedente.

Echivalenţa formulelor $f_1 \Rightarrow f_2 \iff \neg f_1 \vee f_2$ se scrie:

$$\varphi(f_1 \Rightarrow f_2) = (\neg f_1 \vee f_2) = \varphi(\neg f_1) + \varphi(f_2) - \varphi(\neg f_1) \cdot \varphi(f_2) =$$
$$= 1 - \varphi(f_1) + \varphi(f_2) - \left(1 - \varphi(f_1)\right) \cdot \varphi(f_2) =$$
$$= 1 - \varphi(f_1) + \varphi(f_2) - \varphi(f_2) + \varphi(f_1)\varphi(f_2) =$$
$$= 1 - \varphi(f_1) + \varphi(f_1) \cdot \varphi(f_2)$$

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

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

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

c) legea distribuţiei: $f_1 \wedge (f_2 \vee f_3) \iff (f_1 \wedge f_2) \vee (f_1 \wedge f_3)$;
$f_1 \vee (f_2 \wedge f_3) \iff (f_1 \vee f_2) \wedge (f_1 \vee f_3)$;

Dacă pînă acum am stabilit relaţii globale asupra formulelor atomice indiferent de mulţimea 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 mulţime de valori în domeniul studiat îl putem reprezenta funcţional ca bărbat (X) cu X variabilă în domeniul entităţilor unei baze de date asociată sistemului IA în contextul căruia s-a definit formula predicativă "bărbat (X)".

Avem următoarele 2 situaţii posibile:

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

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

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

a) $(\forall)X \Rightarrow$ bărbat (X);

b) $(\exists)X$ astfel ca bărbat (X).

În general pentru un predicat oarecare P, avînd o semnificaţie dată, semnific (P), propoziţia "$(\forall)X, P(X)$" are valoare de adevăr A pentru toate valorile lui X în domeniul entităţilor, iar "$(\exists)X$ astfel încît P(X)" are valoare de adevăr A pentru cel puţin o atribuire de entitate lui X.

Trecînd 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 $f_1 o f_2$. 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) f2: Ou (X,Alb) - cu X parcurgînd mulțimea găinilor;

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

Avem implicația cuantificată:

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

Cum variabila X parcurge o mulțime de entități definită, o 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 "(∀) 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,∃ o mulțime Y, a.î. 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:

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

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 demonstrației 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 origanizate ca o disjuncţie de literali (prin literali se înţeleg formulele atomice precum şi negaţiile 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 rezoluţiei în demonstrarea unei teoreme presupunînd transformarea **rbd**-urilor de la care se porneşte demonstrarea, în clauze. Formal, vom arăta că dacă **rbd**-urile:

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

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

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

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

i) conversia şi scrierea unui **rbd** sub formă de clauze deci sub formă de disjuncţii de **rbd**-uri;

ii) acţiunea rezoluţiei asupra clauzelor particulare;

iii) acţiunea rezoluţiei asupra clauzelor care conţin variabile.

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

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

a) Din clauzele părinte:

cl_1 şi $\neg cl_1 \vee cl_2$ rezultă rezolvanta $cl_2 \iff (cl_1 \Rightarrow cl_2)$;

b) Din clauzele părinte:

$cl_1 \vee cl_2$ şi $\neg cl_1 \vee cl_2$ rezultă rezolvanta: cl_2;

c) Din clauzele părinte: $cl_1 \vee cl_2$ şi $\neg cl_1 \vee \neg cl_2$ rezultă rezolvanta: $cl_1 \vee \neg cl_1$ şi $cl_2 \vee \neg cl_2$, ambele fiind tautologii.

(numim tautologie în domeniul propoziţiilor $\{Pi\}$ o funcţională φ cu $\varphi(Pi) = 1$ pentru Pi adevărată; $\varphi(Pi) = 0$ pentru Pi falsă);

d) Din $\neg cl$ şi cl rezultă rezolvanta NIL, numită şi clauza vidă.

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

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

Să introducem în continuare o noţiune şi anume aceea de **funcţională 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 acum funcţionala S definită pe mulţimea tuturor x-ilor

pentru care există imaginea S(x), deci S:{x} --> ImS.

Numim aplicaţia S funcţională Skolem iar scopul ei constă în eliminarea cuantificatorului existenţial (∃). Atunci avem echivalenţa:

$$(\forall)x"(\exists y)f(x,y) \iff (\forall)x\big(f(x,S(x)\big).$$

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

Procesul constă în următorii paşi logici:

1) Se elimină simbolurile de implicaţie, "⇒". Aceasta, este posibil, după cum am arătat, folosind echivalenţa "$f_1 \Rightarrow f_2$" ⟺ "$\neg f_1 \vee f_2$".

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

De exemplu o formulă de forma:

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

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

3) Standardizarea variabilelor. Acest procedeu constă în redenumirea fiecărei variabile sub un cuantificator (∀) sau (∃) 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 echivalenţa:

$$(\forall)\ x\ \big(f_1(x) \Rightarrow \exists(x)f_2(x)\big) \iff (\forall)x\big(P(x) \Rightarrow (\exists y)f_2(y)\big)$$
$$\text{sau } (\forall)\ x\ \big(f_1(x)\big] \wedge (\exists x)\big(f_2(y,x) \wedge \sim f_1(x)\big)$$

se pot înlocui cu:

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

4) Eliminarea cuantificatorilor existenţiali. Acest pas constă în Skolemizarea rbd-urilor după regula descrisă în definiţia funcţionalelor Skolem, cu observaţia că simbolurile folosite în funcţiile Skolem nu trebuie să fi apărut anterior în nici un rbd.

Dacă variabila existentă după cuantificatorul existenţial nu este precedată de nici o variabilă definită universal, în sensul că nu avem rbd ce începe cu "(∀x)∃y" ci rbd ce începe cu "(∃x)", atunci funcţionala Skolem nu are nici un argument ci se identifică cu o constantă, ca de exemplu: "(∃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 existenţiali, iar fiecare cuantificator universal îşi are propria sa variabilă în numerotare literală distinctă, toţi cuantificatorii universali se scot în afara regulilor rezultînd astfel o formulă ce are ca prefix mulţimea cuantificatorilor urmată de o formulă fără cuantificatori denumită formal matrice.

Astfel, o relaţie de forma:

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

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 conjuncţie de disjuncţii de literali (rbd-uri şi ¬ rbd-uri), forma obţinută fiind cunoscută ca forma normală conjunctivă. De exemplu: $(f_1 \vee f_2) \wedge (f_1 \vee f \neg f_3) \wedge f2$ este o formă normală conjunctivă sau o conjuncţie de disjuncţii. În scrierea acestei forme se folosesc regulile

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

7) Eliminarea cuantificatorilor universali.

Eliminarea este imediată din 6) ţinînd cont că ordinea de eliminare nu are importanţă.

8) Eliminarea simbolurilor "$\wedge$".

Putem astfel înlocui $(f_1 \wedge f_2)$ cu mulţimea $\{f_1, f_2\}$ obţinînd în final o mulţime finită de rbd-uri ce sînt conjuncţii de disjuncţii de literali, rbd-uri numite clauze.

9) Redenumirea variabilelor

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

Limbaje de programare
în designul sistemelor de operare şi al aplicaţiilor.
Limbajul C (I).

Cum, cînd, ce C!

Ne propunem un curs de învăţare şi aplicare a celui mai utilizat limbaj de programare a generaţiei 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 dispoziţia 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;
- eficienţa codului generat este controlabilă de către utilizator;
- este un limbaj structurat: conţine mecanisme suficient de bogate pentru a permite codificare şi folosirea metodei programării structurate;
- concizitate: permite o exprimare compactă şi cu puţine cuvinte a programelor.

Ca precursori ai lui C pot fi consideraţi 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 şi 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 deficienţele lui B.

Dacă considerăm următorul mod de ierarhizare a limbajelor de programare:

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

atunci putem considera locul lui C între 3 şi 4.

Prin această poziţie în ierarhie, C prezintă o dualitate:

- posedă mecanisme de nivel scăzut prin care se poate face referire la elemente ale maşinii (registre, biţi, cuvinte, pointeri) dar suficient de abstracte pentru a asigura o independenţă relativă faţă de o maşină 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 şi mai puţin de 1000 în limbaj de asamblare.

2. Concepte de bază

Conceptul de bază în C este cel de funcţie. Funcţia este o entitate compilabilă separat. Poate accepta parametrii de intrare la apel şi poare returna o valoare după execuţie.

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

Invocarea funcţiei se face prin numele ei însoţit de o eventuală listă de parametrii şi urmat, eventual, de ;:

nume_funcţie(p_1,p_2,...,p_n);

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

Structura generală a unei funcţii este:

```
nume_funcţie(p1,p2,...,pn)
   -
   -   lista de declaraţii de date
   -   (parametrii de apel ai funcţiei)
   -
   {
   -
   -   instrucţiuni specifice funcţiei reprezentate de:
   -    - instrucţiuni de declarare a datelor locale funcţiei;
   -    - instrucţiuni executabile (inclusiv apeluri de funcţii);
   }
```

- { } includ sau definesc un bloc de instrucţiuni sau o secvenţă.

În C, la apelul unei funcţii, datele se transmit prin stivă. Funcţia primeşte o copie a valorilor datelor necesare. În unele cazuri datele pot fi furnizate şi prin adrese.

Funcţiile în C sînt recursive. Dacă funcţia nu are parametrii atunci se defineşte prin nume_funcţie().

Structura programului este:

```
main()
  {
    -
    -   instrucţiuni
    -
    -
  }.
```

Orice program în C operează cu trei periferice standard (influenţă UNIX) şi anume:

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

Prin specificare >>spec_fis se va realiza adăugarea la sfîrşitul fişierului identificat de spec_fis.

Pentru operaţiile de I/O C nu conţine instrucţiuni specifice ci un set de funcţii standard pentru efectuarea acestora. Aceste funcţii sînt singurele care se adaptează la specificul unui calculator.

Pentru rularea unui program se parcurg paşii:

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

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

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

În această secvenţă de instrucţiuni funcţia printf() este o funcţie standard a limbajului C care realizează afişarea liberă sau, conform unui format specificat, a textelor introduse ca literali şi/sau a conţinutului variabilelor sau rezultatului evaluării funcţiilor specificate ca argument, pe ieşirea standard.

Prin #include <stdio.h> se specifică faptul că la compilarea programului se vor include specificaţiile date în fişierul <stdio.h>. Acest fişier conţine declaraţii, 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 şi 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 fişierului va fi AFIŞEAZA.C.

În aceste condiţii vom avea următoarea secvenţă de operaţii:

```
>XCC AFIŞEAZA.C  ⇒ AFIŞEAZA.S
>; realizează compilarea programului
>XAS  -l  -d  AFIŞEAZA  ⇒ AFIŞEAZA.OBJ, AFIŞEAZA.LST
>; realizează asamblarea cu ştergerea fişierului .S(-d)
>;. şi generarea fişierului de listare .LST(-l)
>TKB AFIŞEAZA/CP=AFIŞEAZA,LB:[1,1]C/LB
>; ediţia de legături
>RUN AFIŞEAZA
 De CRACIUN ne-am luat raţia de LIBERTATE!
```

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

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

2. Compilarea programului cu comanda

C88 <nume_fişier> [<opţiune>]

unde

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

<opţiuni> pot fi precedate de "_" (influenţă UNIX) şi sînt:

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

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

M - produce un format obiect INTEL;

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

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

Onume - specifică producerea unui fişier obiect cu numele "nume". Dacă "nume" nu conţine şi extensii atunci va primi implicit extensia .O;

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

3. Asamblarea programelor de tip .A:

ASM88 <nume_fişier> [<opţiuni>]

unde

<nume_fişier> este numele fişierului care este supus analizei asamblorului;

<opţiuni>:

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

M - produce un fişier obiect format INTEL;

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

Td - utilizarea fişierelor temporare la asamblare;

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

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

4. Obţinerea codului executabil:

BIND <nume_fişier$_1$><nume_fişier$_2$>...[<opţiuni>]

Această comandă solicită existenţa bibliotecii standard DESMET CSTDIO.S.

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

<opţiuni>:

A - în lista de fişiere avem şi fişiere obiect obţinute prin asamblare;

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

Fnume - numele fişierelor destinate lui BIND pot fi introduse într-un fişier 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[nume] - o listă sortată cu numele programelor de tip .S declarate "public" şi

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

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

Sub sistemul de operare UNIX se vor parcurge următorii paşi:

1. Compilare:

a) faza 1:

MC1 [=stivă][>fişier_listă] fişier [<opţiuni>]

=stivă - dimensiunea zonei de recursivitate (implicit 2048 bytes);
fişier_listă - fişierul pe care se direcţionează mesajele;
fişier - specificatorul fişierului sursă;
opţiuni - se dau cu litere mici precedate de -:

-a - abandonează optimizările pentru pointeri;
-b - forţează alinierea la byte;
-c - comentariile se prelucrează în stivă
-d - cauzează includerea începutului informaţiilor în fişierul "quad" (numărul liniei inclus în fişierul quad);
-id - citeşte toate #include... din discul "d";
-od - crează fişierul de ieşire (quad) pe discul "d";
-x - schimbă clasa de memorare pentru declaraţiile externe (external).

b) faza 2:

MC2 nume_fis [<opţiuni>]

opţiuni:
-od - crează fişierul de ieşire (.OBJ) pe discul "d";
nume_fis - fişierul de intrare (.Q).

2. Obţinerea codului executabil:

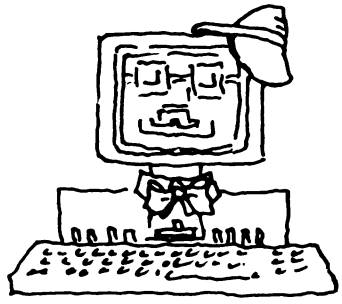
LINK C.OBJ + nume_fis$_1$.OBJ + ... + MC.LIB

Se obţine un program executabil cu numele nume_fis$_1$.EXE.

3. Executarea programului:

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

=stivă - toate obiectele declarate cu clasa "auto" vor fi alocate pe stivă;
<fis.in - numele fişierului care va substitui intrarea standard;
>fis.ies - fişierul pe care se va redirecta ieşirea standard; fis.ies va fi deschis pentru adăugare;
<args> - sînt argumente destinate funcţiei main() (citate în program prin argc şi argv) care vor fi transmise acesteia ca doi pointeri (argc dă numărul de elemente pe care le conţine argv).



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

Designul şi exploatarea foilor de calcul electronice (I)

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

Sigur, este greu să defineşti în puţine cuvinte utilitatea unor pachete de programe a căror documentaţie conţine sute de pagini. De aceea vom trece la o scurtă trecere în revistă a acestora la sfîrşitul căreia orice cititor va avea o primă imagine asupra utilităţii şi utilizării acestor tipuri de pachete de programare.

Vom începe cu pachetul de programe 1-2-3, produs al firmei Lotus Development Corporation, larg răspîndit 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 poziţia 1,1; IV 8192 pentru elementul din poziţia 8192,256). Fiecare celulă poate conţine informaţia care poate fi modificată interactiv. La un moment dat utilizatorul nu poate selecta decît o singură celulă în care se află cursorul şi care se numeşte celula curentă. Conţinutul 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 informaţii care alcătuiesc împreună un panou de control.

Prima linie a panoului conţine informaţii despre celula curentă incluzînd adresa celulei, conţinutul celulei şi, la cerere, prin opţiunile selectate: lăţimea coloanei, formatul acesteia, protecţia aferentă.

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

A treia linie afişează, fie sub menu-ul selectat opţiunile din acesta (pentru modulul MENU), fie o scurtă descriere a opţiunii selectate din menu.

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

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

Indicatorul dată şi oră (la partea inferioară stîngă a ecranului) indică data şi 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. Utilizînd combinaţii ale tastei END cu tastele HOME, UP, DOWN, RIGHT, LEFT cursorul poate fi mutat în colţul drept al zonei ocupate cu date din foaie, respectiv ultima poziţie ocupată pe direcţia specificată.

Introducerea şi editarea datelor de la tastatură

În celula selectată de cursor se pot introduce şi/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.

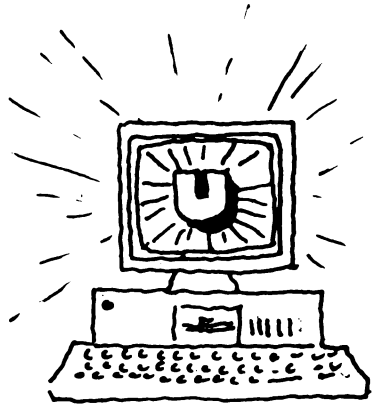
La validarea datei, 1-2-3 efectuează acţiunile:

- verifică existenţa erorilor; dacă există se emite un semnal sonor, se deplasează cursorul în celula respectivă şi se trece automat în modul EDIT;
- dacă nu este găsită nici o eroare, data se introduce în celula curentă; data anterioară dispare şi nu mai poate fi recuperată;
- în funcţie 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 reaminteşte 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 conţine. Pentru o dată în curs de introducere se porneşte cu pasul următor;
- se apasă tasta EDIT. Indicatorul de mod trece în modul EDIT. Utilizînd tastele direcţionale cursorul se poziţionează corespunzător;
- se inserează sau se şterg caractere în poziţia selectată de cursor;
- pentru validare se apasă tasta RETURN.



Baze de date distribuite (I)

1. Definiţie şi obiective

1.1. Motivaţie

Mediul economico-social este într-un continuu proces de schimbare. Acest proces se materializează şi prin apariţia de noi întreprinderi (firme, corporaţii, etc.), dispariţia altora, diviziunea unora sau reunirea în conglomerate cu piese componente de diverse orientări şi structuri organizatorico-funcţionale.

Indiferent care este modul lor de evoluţie (exceptînd dispariţia "fizică") ele cooperează tot timpul printr-un masiv flux informaţional.

Spaţiul geografic nu mai reprezintă un impediment al acestor restructurări iar accesibilitatea tehnicii de calcul, ca suport al fluxului informaţional, reprezintă o problemă de opţiune.

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

Iniţial 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, reţele de calculatoare, etc.), aveau acces, diferenţiat, toate structurile întereprinderii.

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

Diseminarea informaticii prin explozia microcalculatoarelor şi necesitatea unei cît mai bune modelări şi adaptabilităţi la viaţa reală a dus la apariţia bazelor de date distribuite.

În suita de articole, pe această temă, vom prezenta caracteristicile arhitectura bazelor de date distribuite precum şi modalităţile de realizare proiectării lor. La sfîrşitul expunerii vom prezenta două studii de caz privind proiectarea bazelor de date distribuite.

1.2. Definiţie

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

Logic integrată: din punct de vedere al utilizatorului BDD este percepută ca o singură BD. El interacţionează cu o BDD în acelaşi mod în care ar opera cu o BD de uz centralizată. Utilizatorul nu trebuie să cunoască modul de partiţionare, replicare şi distribuire a datelor. Singurul lucru pe care îl cunoaşte este schema conceptuală a bazei de date, schemă denumită schema globală. Prelucrările iniţiate

la un nod pe care se află partiţii (părţi) ale bazei de date antrenează, în general, prelucrarea informaţiilor aflate în alt nod.

De exemplu, bazele de date locale BDL1, BDL2 şi BDL3 stocate în nodurile unei reţele notate (corespondent) N1, N2 şi, 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 (aparţinînd oricăruia dintre tipurile de utilizatori ai bazelor de date) care are acces şi exploatează o bază locală. El cunoaşte şi manipulează numai schema alcelei baze de date locale (sau o subschemă a acesteia). Manipularea bazei de date locale include oricare din bazele de date din reţea (chiar dacă se efectuează cu terminal virtual utilizînd numai facilităţile puse la dispoziţie de software-ul de reţea).

Un **utilizator global** aparţine oricăruia din tipurile de utilizatori ai bazelor de date cu singura diferenţă că el cunoaşte şi are acces la schema (sau o subschemă globală a) bazei de date globale. Utilizatorul global exploatează schema globală, conform autorizărilor şi drepturilor sale de acces, de aceeaşi 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 partiţionată iar partiţiile 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 şi administrată local.

Baza de date distribuită (o mulţime de colecţii de date şi legăturile dintre ele) este fragmentată conform pricipiilor:

- plasarea datelor memorate în nodul de producere şi utilizare frecventă a lor;
- minimizarea transportului de date prin reţeaua de calculatoare.

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

- partiţionarea mulţimii de colecţii în submulţimi de colecţii de date;
- partiţionarea unei colecţii de date în fragmente.

Partiţionarea unei colecţii de date poate fi realizată în două moduri:

- **orizontal** - fragmentele obţinute au aceeaşi schemă conceptuală (structură) ca şi colecţia, dar diferă între ele prin datele pe care le conţin;
- **vertical** - fragmentele (fiecare în parte) conţin doar o parte din schema conceptuală a colecţiei.

Este admisă şi combinarea acestor două moduri.

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

1.3. Obiective

În concordanţă cu definiţia un sistem de baze de date distribuite trebuie să asigure cel puţin următoarele obiective:

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

2° **evitarea unei foarte înalte centralizări a resurselor,** centralizare care cauzează foarte mult eficienţei şi eficacităţii sistemului: o rată înaltă a

centralizării cauzează un cost ridicat de prelucrare şi 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 aplicaţiile existente;

4° **să facă mai uşoare operaţiile de menţinere şi restructurarea a bazei de date cu menţinerea unei rate înalte a disponibilităţii:** deoarece cea ce vede un utilizator global reprezintă o schemă globală, care are drept corespondent diverse scheme locale, operaţia de restructurare a unei scheme locale nu afectează cu nimic utilizatorul global;

5° **să permită proiectarea structurii organizatorice şi funcţionale a sistemului analizat:** prin faptul că, în general, bazele de date locale se află în locurile în care se produc informaţiile pe care le conţin, o arhitectură distribuită permite o emulare mai puternică, a sistemului informaţional, conformă structurii organizatorice şi funcţionale;

6° **să mărească gradul de utilizare a sistemului:** prin emularea cadrului organizatoric şi funcţional şi prin disponibilitatea datelor se obţine creşterea numărului de utilizatori ai sistemului.

Realizarea acestor obiective atrage după sine considerarea şi 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 transparenţei alocării fizice a datelor faţă de utilizator:** exceptînd utilizatorul (utilizatorii) special (i), administratorul bazei de date, ceilalţi utilizatori nu trbuie să cunoască locul în care sînt alocate datele pentru a-şi formula întrbările adresate bazei de date;

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

- **asigurarea unui sistem eficient de catalogare:** programele de aplicaţie trebuie să fie disponibile în toate nodurile reţelei pentru a realiza o exploatare eficientă a aplicaţiilor 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 corespondenţei schemei locale cu schema globală, programe a căror prezenţă este obligatorie în toate nodurile reţelei;

- **asigurarea independenţei logice, fizice şi distributive a datelor:** independenţa logică şi fizică trebuie asigurată similar ca bazele de date centralizate. Independenţa distributivă se referă la faptul că dacă schimbăm nodul în care este stocată o partiţie a bazei de date distribuite acest lucru nu trebuie să influenţeze aplicaţiile şi schema conceptuală globală.

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

- prevenirea creşterii redunanţei sau a inconsistenţei datelor la dezvoltarea de noi aplicaţii;

- definirea unor noi instrucţiuni pentru standarde de definire şi utilizare a datelor;

- administrarea eficientă a cererilor;
- utilizarea eficientă a resurselor de telecomunicaţie (reţelei de calculatoare); etc.

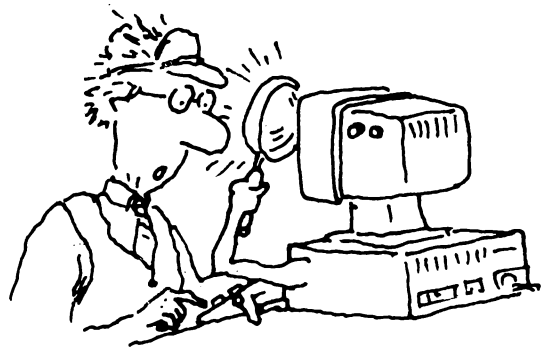
1.4. Bază de date relaţională distribuită

Definiţia bazei de date distribuită se adaptează modelului relaţional astfel:

"O bază de date relaţională distribuită (BDDR) constă dintr-o colecţie de tabele (relaţii) fiecare din ele putînd fi stocată într-un singur nod al reţelei sau poate fi răspîndită într-o reţea de calculatoare. Fiecare relaţie distribuită poate fi fragmentată **orizontal** sau **vertical** în acord cu un criteriu de distribuire (în general un predicat de distribuire)".

O **tabelă (relaţie)** este **locală** dacă este stocată în întregime într-un singur nod şi **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 distribiută) funcţionează ca un sistem centralizat în timp ce fizic se adaptează repartiţiei componentelor unei întreprineri sau administrări.



Teme pentru informaticieni.
Prelucrarea listelor în C şi PASCAL.

Prelucrarea listelor în C şi PASCAL (I)

În cele ce urmează vom analiza posibilitatea mînuirii structurilor de date dinamice în Limbajele C şi Pascal, date care, spre deosebire de variabilele statice (de tip vector sau matrice, declarate în secţiunile de date şi alocate într-un spaţiu 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, funcţie de utilizarea sa. Avem deci nevoie de definirea unei structuri de date cu un membru care să fie o referinţă de tip pointer la o structură de acelaşi tip cu cea definită primar, permiţînd astfel înlănţuirea unui număr nespecificat de structuri de acelaşi tip. Vom numi un astfel de lanţ de structuri înlănţuite **listă** de date, fiecare element al listei avînd deci aceeaşi formă declarată. Definim o listă dacă definim forma elementelor sale. Astfel, definire unei liste formată din numere întregi presupune definire fiecărui element astfel:

În C:

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

În Pascal aceeaşi structură admite definirea:

```
Type   listă = ↑element
   element = record
                 număr:int;
                 adresa_următorului_element:listă
             end;
Var Lis:listă;
```

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

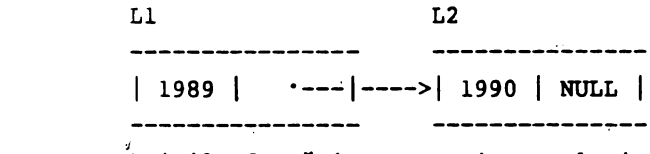
- În C:

```
struct   listă   L1,L2;
```

- Atribuirea de valori o facem prin:

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

L1. adresa_următorului_element = L2.adresa_următorului_element = NULL;
- Înlănţuirea celor două liste ne dă:
  L1. adresa_următorului_element = &L2
- schematic avem:



- 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:
- declaraţia 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 înşiruire secvenţială de elemente de acelaşi tip, cu un prim element adresat de un pointer de început (sau cap de listă), fiecare alt element conţinînd adresa succesorului 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
                        succesor din listă */
  typedef struct lista ENTITATE;
  typedef ENTITATE     *LIS;
};
```

Alocarea spaţiului pentru definirea de mai sus, se poate face dinamic în funcţie de nevoia de elemente noi în listă.
Limbajul C prin funcţia <malloc()>, apelată prin:

malloc (spaţiu)

întoarce un pointer pentru un obiect cu un număr de <spaţiu> octeţi (parametrul <spaţiu> 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 conţină 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 funcţie care să execute recursiv paşii de mai sus, generînd un modul sursă mai compact şi mai uşor de manevrat. Este ceea ce vom realiza în Pascal. Să scriem deci codul sursă de mai sus beneficiind de facilităţile limbajului Pascal care conţine instrucţiunea <new> pentru alocarea de spaţiu unui nou element al listei.

Să definim astfel o procedură pentru crearea şi 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 poziţia din vîrful 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 condiţie de oprire semnalată de utilizator în program.

Putem procesa şi operaţia de scriere a elementelor unei liste create prin procedura de mai sus, impunînd condiţia 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 şi iteraţia joacă un rol important în prelucrarea datelor.

Să prezentăm astfel, în limbajul C, două rutine de generare a unei liste dintr-un şir de întregi, una realizată prin recursie, alta prin iteraţie.

Amîndouă funcţiile generate întorc un pointer în cadrul listei rezultate:

- rutina prin recursie

```
Lis şir_în_listă (l)
  integer    l[ ];
     {
```

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

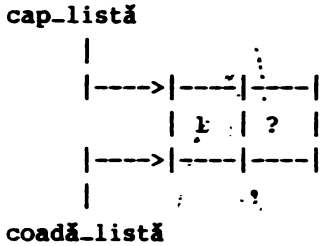
Următoarea rutină foloseşte generarea listei prin punerea în evidenţă a capului de listă şi a cozii sale (restul elementelor) procedeu utilizat ca strategie de bază în programarea logică.

- rutina prin iteraţie (l)

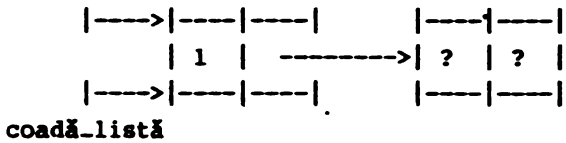
```
Lis şir_în_listă (l)
  integer    l;
  {
   Lis  cap_listă = NULL, coadă_listă;
   int k;
   if (l[0] != '\0' ){
      cap_listă = (Lis) malloc (sizeof(ENTITATE));
      cap_listă -> număr = l[0];
      coadă_listă = cap_listă;
      for (k = 1; l[k]!='\0'; ++k){
        coadă_listă -> următorul=(Lis)malloc(sizeof(ENTITATE));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

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

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



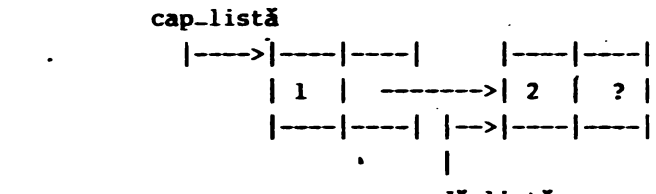
Bucla "for" executată cu i = 1, crează un nou element:



Acţiunea instrucţiunii

coadă_listă = coadă_listă -> următorul

avansează coada listei, cu atribuirea valorii, <număr>=2.



Valoarea lui l[2]='\0'impune înscrierea în ultimul element a valorii NULL(NIL) în locul lui "?".

Următoarele trei rutine realizează operaţii specifice cu liste şi anume: numărarea elementelor unei liste,tipărirea recursivă a elementelor unei liste şi concatenarea a două liste.

1. Numărul de elemente ale unei liste:

În C:

```
număr (cap_listă)
Lis cap_listă;
{
    if (cap_listă ==NULL)
        return (0);
    else
     return (1+număr (cap_listă -> următorul));
}
```

În Pascal:

```
function număr (cap_listă:Lis);
  begin
   if (cap_listă= NIL)then
          număr:=0
   else
       număr: = număr(cap_listă↑.următorul)
   end.
```

2. Scrierea elementelor unei liste.

În C:

```
tipăreşte (cap_listă)
 Lis  cap_listă;
 {
    if (cap_listă == NULL)
        printf("Lista este vidă");
    else {
        printf("%c ->",cap_listă->număr);
              tipăreşte(cap_listă->următorul);
    }
 }
```

În Pascal:

```
procedure tipăreşte (cap_listă:Lis);
   var L:Lis
   begin
    L:= cap_listă;
```

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

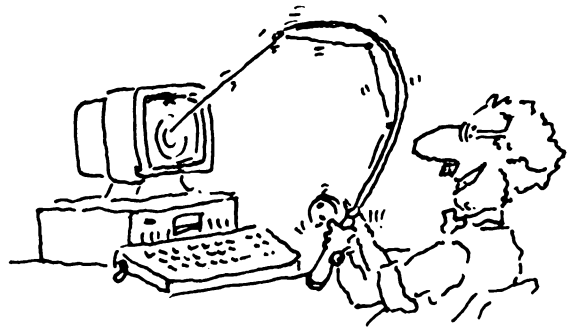
3. Concatenarea a două liste:

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, noutăţi, istoric în informatică

Din istoria calculatoarelor personale (I)

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

Dezvoltarea tehnologiei hardware a deschis noi căi. Nivelul investiţiilor, 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 aceeaşi utilizatori, maşini proprii mai performante decît microprocesoarele zilelor noastre, cu instrumente de I/E grafice şi sonore. Dezvoltarea va asigura ocolirea compromisurilor şi limitelor impuse de sistemele cu divizarea timpului. Tehnologiile de reţea noi, de viteză mare vor păstra avantajele sistemelor cu divizare a timpului: informaţii distribuite, comunicaţii între utilizatori, partajarea perifericelor scumpe. Vom păşi în lumea calculatoarelor personale multilaterale şi lumea aceasta diferă de lumea sistemelorcu divizarea timpului aşa cum şi sistemele cu divizare a timpului diferă de prelucrările batch.

Referinţe: 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 eficienţa muncii intelectuale. Distanţa, pe scara productivităţii profesionale, între oameni apropiaţi ca posibilităţi, dintre care unul este înarmat cu mijloace corespunzătoare iar altul este neînarmat, se va mări, evident, rapid. Prin aceasta se şi explică ritmurile ridicate de creştere a vînzărilor de calculatoare personale din ultimii 10 ani. În 1976 s-au vîndut 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 eficacităţii deosebite a calculatoarelor personale în activitatea profesională a milioane de oameni care lucrează în sfera informaţională a economiei naţionale trebuie să se considere faptul următor: circa jumătate din întregul tiraj al calculatoarelor personale este cumpărată de profesionişti de nivel mediu şi superior ai mecanismului economic din SUA din banii proprii "obţinuţi prin sudoare". Pentru acest contingent de cumpărători preţul calculatorului personal constituie aproximativ salariul pe o lună. În 1990 preţul unui calculator s-a redus la 1/3 dintr-un salariu lunar.

Alfabetizarea computerizată în masă şi tirajele de multe milioane în producţia de calculatoare personale - maşinile unelte ale celei de-a doua revoluţii industriale - imprimă un curs puternic dezvoltării forţelor de producţie. Întîrzierea în dezvoltarea industriei calculatoarelor personale - acolo unde ea va exista - va necesita, deja la sfîrşitul deceniului nostru, eforturi costisitoare în masă pentru a preîntîmpina numai cele mai evidente urmări legate de influenţa calculatoarelor personale asupra ritmurilor de creştere a

productivităţii muncii în sfera informaţională a economiei naţionale. Iar, în ansamblu, apariţia în viaţa noastră a fenomenului calculatorului personal poate provoca probleme care sînt comparative ca amploare cu mişcarea cunoscută din istorie pentru lichidarea analfabetismului.

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

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

. . . . . .

Evoluţia a închis un ciclu în spirală, cu calculatoarele personale ne întoarcem la un nivel mai înalt la utilizarea, în apropierea omului, a maşinilor din generaţia unu şi doi. Maşinile vechi se utilizau în mod "personal" din cauza capacităţii lor reduse, în schimb microcalculatoare pot fi aduse în apropierea omului datorită preţului lor scăzut.

Cîteva date din istoria calculatorului personal

1959 - Anul de naştere al microelectronicii, primul circuit integrat (TEXAS INSTRUMENTS).

1971 - Primul microprocesor cu 4 biţi, 4004-Intel.

1974 - Microprocesorul cu 8 biţi Intel 8080.

1975 - Microprocesorul 6502 la preţul de 25 $.
Cluburi de construire a microcalculatoarelor.
Apariţia revistei pentru microcalculatoare Byte.

1976 - Primul calculator personal Apple II.

1978 - Pachetele Wordstar, Visicalc, dBase II.

1981 - Apariţia calculatorului profesional IBM PC.

1983 - Pachetul integrat Lotus 1-2-3.

1984 - Microprocesoarele din familia Intel 80816, 80286 pe 16 biţi şi 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 evoluate C, Pascal, Assembler.

1986 - Apariţia tehnicilor în inteligenţa artificială Microlisp, Prolog, procesoare de limbaj natural.

1987 - Explozia tehnologică în arhitectura hardware-calculatoare cu discuri hard şi Streamer de 30, 60 MB.

1988 - Apariţia sistemului de operare OS/2 pentru lucrul multitasking pe microcalculatoare, reţele de comunicaţie micro-micro, micro-mainframe.

1989 - Discuri de masă rigide de 100, 200 MB precum şi tehnologia de procesare paralelă.
Introducerea discurilor optice read-write cu capacitate peste 1 MO.

1990 - Consolidarea şi dezvoltarea noilor tendinţe astfel:

1. Utilizarea de noi tipuri de microprocesoare care permit modul de lucru protejat (80286, 80386 şi 80486);

2. Mărirea capacităţii memoriei interne operative (16MO - 4GO);

3. Creşterea vitezei de prelucrare;

4. Mărirea vitezei de transfer la operaţiile de I/O prin creşterea volumului de date transferate (5MO/S - 33MO/S);

5. Mărirea capacităţii de stocare a discului hard (1,2 GO);

6. Extinderea posibilităţilor de lucru în mod grafic;

7. Oferirea unor mecanisme care să permită integrarea facilă a micro în reţele de calculatoare eterogene.

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

a) ISA (Industry Standard Architecture) propusă şi dezvoltată de IBM la construirea lui PS/2. Prin acest standard debitul pentru operaţiile 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 reţea se poate realiza în arhitecturi de tip SNA;

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

Din punct de vedere software se degajă următoarele tendinţe:

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 şi 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 reţele de calculatoare eterogene;

6. Încorporarea unui SGBD relaţional în software-ul de bază (limbajul utilizat de SGBD este SQL).

Categorii de calculatoare personale

În cei 10 ani de la apariţia lor, calculatoarele persoanle s-au diversificat atît din punct de vedere al capacităţii cît şi 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 biţi (Intel 8080, Z80, 6502), au o memorie dimanică de pînă la 48 Kocteţi. De obicei se livrează în forma unei tastaturi care încorporează atît unitatea centrală cît şi circuitele auxiliare. Perifericele acestei categorii de calculatoare sînt aparate electronice care se găsesc în jurul casei: televizorul, cu rolul de afişare, şi casetofonul (eventual magnetofonul) pentru înregistrarea programelor şi a datelor.

Tastatura, de obicei, este tastatură de tip folie.

Au generator sonor (adesea pînă la 5 octave).

Interpretorul BASIC este "ars" în memorie.

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

Reprezentative sînt calculatoarele firmelor Sinclair şi 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 şah 40-70 DM (1985)
- diverse interpretoare (MPROLOG, FORTH etc.).

Calculatoare personale semiprofesionale (sau calculatoare CP/M)

Sînt calculatoare realizate cu microprocesoare de 8 biţi. Ceea ce le deosebeşte, ca arhitectură, de calculatoarele familiale este existenţa perifericelor mai pretenţioase (unităţile de disc FLOPPY, imprimante, eventual cuplor de teletransmisie). Arhitectura rămîne totuşi închisă, de tip "foloseşte-l şi 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 maşini BASIC ci au sistem de operare cel mai răspîndit fiind sistemul de operare CP/M. Aplicaţiile generice cele mai uzuale: VISICALC, WORDSTAR, dBASE etc.

Aplicaţiile concrete pentru C64 (întocmit pe baza unor anunţuri din MIKROSZAMITOGEP MAGAZIN):

- evidenţa personalului (1000 de persoane cu cîte 19 date fiecare);
- evidenţa comenzilor şi a stocurilor;
- repartizarea costurilor;
- evidenţa mijloacelor fixe;
- evidenţa mijloacelor de transport (pentru 20 de unităţi şi 300 de maşini);
- calculul salariilor (pentru 850 muncitori);
- evidenţa materialelor;
- facturare.

Calculatoare personale profesionale

Caracteristici:

- sînt realizate cu microprocesoare de 16 sau 32 biţi;
- au memorie dinamică pînă la 1 M;
- unităţile de disc Winchester (pînă la 2x20 Mbaiţi);
- arhitectura deschisă: se pot ataşa noi plachete, configuraţia se poate dezvolta ulterior.

Modele reprezentative:

IBM PC:

- apărut în 1981;
- microprocesor de 16 biţi (Intel 8086);
- o configuraţie 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 biţi (MOTOROLA);
- utilizare cu "icoane" şi "şoricel" (mouse) pentru comanda cursorului;
- soft de bază (128+64K, FLOPPY, soft de bază):
  63655 OS (1984).

Situaţia vînzărilor în domeniul calculatoarelor profesionale (1984):

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

Apariţia modelelor IBM PC, XT, AT pe piaţă a produs o avalanşă atît în creaţia software-ului pentru aceste modele cît şi fabricarea unor modele "compatibil IBM PC".

Modele (compatibile IBM PC):

- HP 150 II:
  - memorie pînă la 640K;
  - Winchester de 4,8 sau 15M;
  - configuraţia 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).

Aplicaţii: birotică, gestiune, calcule ştiinţifice, CAD, CAM etc.

Aplicaţii în gestiunea economică (pe baza unor anunţuri din MIKROSZAMTIOGEP MAGAZIN):

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

Calculatoare personale portabile

Sînt realizate foarte compact, de obicei cu afişare cu cristale lichide. Sînt echipate cu model acustic pentru o cuplare uşoară la reţeaua telefonică (cuplarea = formarea numărului + aşezarea receptorului pe modem).

Modelele mai vechi sînt mai voluminoase (gen valiză):

- OSBORNE;
- IBM PC;
- COMPAQ.

Modelele mai noi sînt relizate 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 fabricanţilor de calculatoare ca aceştia să dezvolte sisteme comercializabile (în jur de 10000 $) şi care să satisfacă criteriile impuse în raport. Această maşină fictivă se numeşte SPICE (Scientific Personal Integrated Computing Environment).

Din specificaţia hardware:

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

Din specificaţia software:

- calcule ştiinţifice şi tehnice, sisteme de prelucrare
- prelucrarea de texte, redactarea de documente (text [illegible]) verificarea ortografiei
- comunicaţii de reţea (gestiunea [illegible]

centralizată, corespondenţă).

Protocolul de comunicaţie de tip Xerox şi ARPA-INTERNET.

Cercetarea legată de calculatoare personale de capacitate mare a luat amploare în S.U.A. pe la mijlocul anilor '70 şi sistemele sînt prezente pe piaţă din 1980. În acest domeniu centrul de cercetare din Palo Alto a firmei Xerox deţine un rol conducător. A apărut cu principii de sistem noi: limbaj de programare de nivel înalt sprijinit de o arhitectură adecvată, reţea locală de viteză mare, "şoricelul" (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 Equipement Corporation;
- Simbolics 3600 - maşina LISP.

Software şi aplicaţii pentru microcalculatoare profesionale şi semiprofesionale

În Ungaria este un obicei ca un institut, în primul rînd, să cumpere sau să închirieze un calculator şi, abia după aceea, să se întrebe la ce i-ar putea folosi. Cîţiva angajaţi sînt instruiţi să opereze calculatorul şi să programeze în BASIC şi se aşteaptă ca el să scrie programele necesare.

Acesta este drumul eşecului. Nu aduce decît dezamăgire, reîntoarcerea la metodele tradiţionale şi descoperirea informaticii.

Ordinea corectă este următoarea:

- a decide care sînt problemele parţial sau integral programabile;
- trebuie însărcinat un specialist în marketing pentru a găsi programul sau pachetul de programe adecvat pentru scopul propus;
- angajaţii sînt instruiţi în programarea structurată, independentă de maşină;
- în acelaşi timp, cu un specialist din afară, se alege calculatorul adecvat pentru pachetul de programe;
- se cumpără pachetul de programe şi maşina;
- specialiştii, instruiţi în programare structurată, vor învăţa utilizarea maşinii şi a pachetului de programe.

Pentru prima dată, în 1983 vînzările de software pentru microcalculatoare au depăşit un miliard de dolari.

	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 (WordStrar)	50
7. Digital Research (CP/M)	44
8. Lotus Developement (1-2-3)	38

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

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

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

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

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

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

2) Aplicaţii cu utilitate directă:

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

- pachete de programe pentru satisfacerea necesităţilor de calcul şi arhivare dintr-un anumit domeniu (asigurări, vînzări de locuinţe, industria hotelieră, probleme de notariat, vînzări de ziare), desene animate, planificarea instalaţiei de încălzire centrală pentru o locuinţă, şah, educaţie, proiectarea circuitelor, proiectarea plachetelor imprimante).

3) Produse pentru dezvoltare de aplicaţii, aplicaţii generice:

- pachete pentru gestiunea bazelor de date;

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

- grafică economică;

- pachete pentru prelucrarea documentelor (word processor);

- pachete pentru comunicaţii între microcalculatoare şi între calculatoare personale şi calculatoare mari;

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

Implementarea APLICAŢIEI CONCRETE nu se face prin programare clasică ci prin definirea problemei în terménii generalizaţi ai aplicaţiei generice.

Pachete pentru gestiunea bazelor de date

Au apărut pe piaţă în formă de generatoare de aplicaţii. Reunesc următoarele funcţii:

- definiţia 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 aplicaţii, chiar şi aplicaţiile complicate şi mari pot fi implementate într-un timp de zece ori mai scurt şi deoarece în aceste implementări nu trebuie utilizaţi programatori şi analişti, aceste sisteme deschid o nouă direcţie de utilizare a microcalculatoarelor.

Pachetul tipic este dBASE II şi varianta lui îmbunătăţită dBASE III sau

IV. Se utilizează sub sistemul de operare CP/M pe calculatoarele cu 8 biţi şi sub MS-DOS pe cele de 16 biţi.

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

O altă categorie de generatoare de aplicaţii:

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

Acestea pot fi utilizate în reţele de calculatoare personale şi minicalculatoare pentru că sînt implementate pe ambele categorii de calculatoare. Un exemplu: conducerea la faţa locului a unor activităţi de construcţii printr-o aplicaţie realizată în Oracle pe calculator IBM PC într-o reţea coordonată de o aplicaţie centrală scrisă tot în Oracle pe un megamini VAX.

Preţul 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 relaţii de calcul între anumite coloane sau între anumite linii şi efectuarea mai multor iteraţii - pe un asemenea tabel - pentru aflarea otpimului.

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

Aplicaţii: vînzări, rabat comercial, prognoză comercială, în planificare la aplicaţii de genul "ce se întîmplă dacă", evidenţă etc.

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

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

Pachete pentru prelucrarea documentelor

Documentul este un ansamblu redactat din texte şi ilustraţii. Pachetele reunesc parţial sau integral următoarele funcţii:

a) introducerea şi prelucrarea textelor (introducerea textului, despărţirea 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 aplicaţii economice pot realiza pe bază de tabele, grafice economice.

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

Pachete de comunicaţii

Calculatoarele personale profesionale pot fi utilizate ca staţii de lucru în reţele sau ca terminale conversaţionale.

Pentru aceasta sînt dotate cu cuploare (interfeţe) de transmisie şi 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 aplicaţie a calculatorului în birotică: pe baza unor date extrase dintr-o bază de date se întocmeşte un raport, cu ajutorul unui program de prelucrare de texte, iar în acest raport anumite părţi sînt redate sub formă de tabele sau grafice.

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

Denumirea de 1-2-3 arată că s-a reuşit reunirea a trei funcţii într-o singură aplicaţie generică:

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

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

Pachetul SYMPHONY (tot firma LOTUS 1984) adaugă încă două funcţii la posibilităţile lui 1-2-3 şi anume: prelucrarea de texte şi comunicaţii de date.

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

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

Reţele locâle

Aplicaţiile pe microcalculatoare, care pînă acum funcţionau izolate, se transformă în sisteme informatice distribuite, reunite prin reţele de comunicaţie.

Însuşi calculatorul personal se transformă în staţie personală de calcul.

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

- dinamism la nivelul staţiei de lucru la înregistrarea, prelucrarea, afişarea informaţiilor (să ne gîndim la desenarea unei figuri sau la modificarea lui cu pachet de grafică);
- dinamism în schimbul de informaţii între staţiile de lucru;
- dinamism în "comportamentul" sistemului informatic (utilizatorul este cel care defineşte sau redefineşte sistemul de decizii, baza de date, relaţiile dintre punctele de lucru etc.).

O asemenea reţea (denumită şi reţea locală - local area network) se compune din:

- staţii de lucru constituite din calculatoare personale;
- puncte de activităţi specifice pentru descrierea reţelei, cu activităţi

care sînt partajabile între utilizatori:

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

Aceste puncte sînt deservite de calculatoare specializate.
- reţeaua de comunicaţii.
Din "Proposal for a joint Effort in Personal Scientific Computing".

Servicii de reţea

Într-un proiect mare, de obicei, iau parte mai mulţi. 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 iniţia apelul unuia sau a mai multor colegi, prin reţea. Cei chemaţi pot accepta sau pot refuza apelul. Teleconferinţa se desfăşoară prin periferice de I/E de sunet iar transmisia prin reţea compactat, digitizat (realizat: voice switching - Xerox - 1984). În cursul discuţiei anumite dialoguri pot fi înregistrate într-o bază proprie pentru a fi reascultate mai tîrziu. La terminarea conferinţei chiar întregul text poate fi expediat, pe baza unei liste, celor interesaţi.

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

Gestionarea apelurilor telefonice

Funcţionarul este departe de locul de muncă dar aşteaptă un apel telefonic important. În acest caz, o maşină dedicată acestui scop primeşte apelul, anunţă că apelatul lipseşte şi solicită numele şi adresa apelantului. Un program de recunoaştere a vorbirii poate înscrie apelul pe lista celor care trebuie rechemaţi.

Servicii de informare

A sosit un coleg nou în institut. Cînd se anunţă de prima dată în sistem acesta îi solicită informaţii biografice şi cu un SPICE dotat cu o cameră de luat vederi este înregistrat portretul lui. În ziua următoare sistemul prezintă noul coleg celorlalţi colegi şi afişează 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 şi datele biografice).

Software şi 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 apariţia microprocesoarelor 80286 şi 80386 din familia Intel ce beneficiază de modul protejat s-a simţit nevoia dezvoltării unui nou sistem de operare care, în acelaşi timp, să poată elimina şi limitările fizice şi logice ale celor vechi.

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

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

1) Poate adresa o memorie fizică de pînă la 16M bytes;
2) Permite execuţia de aplicaţii concurente;
3) Asigură protecţia între aplicaţii simultane independente;
4) Permite multitasking;
5) Permite comunicarea şi sincronizarea între programe;
6) Beneficiază de o interfaţă de aplicaţie programabilă (API);
7) Compatibilitatea cu aplicaţiile scrise sub MS-DOS sau PC-DOS;
8) Permite scrierea şi asigurarea la sistem a driver-erelor utilizator;
9) Conţine un program de instalare şi dezvoltare de sistem.

Odată cu creşterea puterii sistemelor de calculatoare personale gestionarea unei baze de date precum şi dezvoltarea complexităţii structurilor impune utilizarea unor limbaje de interogare şi 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 conţine o puternică bază de date relaţională precum şi 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 relaţional inventat de E.F. Codd la Centrul de cercetări din San Jose. Interesant este aici şi posibilitatea execuţiei sub OS/2 a mai multor procese concurente permiţînd accesul la aceeaşi bază din procese OS/2 distincte, cu păstrareea consistenţei lor, asigurînd şi suportul complet al tranzacţiei (orice citire sau scriere în bază este făcută în cadrul unei tranzacţii).

De asemenea, remarcabilă este realizarea, în cadrul sistemelor OS/2, a programelor suport pentru interfaţa Multipat/2 ce se constituie ca submicro-calculator permiţînd 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 şi graficelor în procesul de afişare sau tipărire.

Informatică în învăţămîntul liceal şi universitar.

Ghid de probleme rezolvate pentru elevi şi studenţi

Această secţiune îşi propune să prezinte soluţiile sub formă de programe sursă a unei mulţimi de probleme diverse ce apar în activitatea curentă de programare. Programele, scrise în diverse limbaje de programare moderne, caută să răspundă atît cerinţelor didactice în privinţa învăţării strategiilor de formalizare algoritmistică cît şi dilemelor curente ce intervin în scrierea aplicaţiilor de către analişti consacraţi. Sub motivaţia enunţurilor, programele prezentate fac uz de modele şi proceduri care, fără îndoială, cercetate în detaliu, oferă cheia unor algoritmi cu o largă întrebuinţare în probleme curente de programare.

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

R: Programul scris în Turbo Pascal (v. 3) citeşte numerele x,y de la tastatură şi, în urma descompunerii celor două numere în factori primi, calculează cmmmc prin produsul factorilor primi comuni şi necomuni la puterea cea mai mare. Se scrie la imprimantă atît descompunerea celor două numere cît şi 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,'lptl');rewrite(lst);
```

```
k:=1;
write(lst,"Introduceţi numărul x,x= ");
readln(x);
writeln(lst,x);
write(lst,"Introduceţi 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];
contoar1: t:=x mod s;
if (t=0) then
  begin
     s:=s*a[i,1];
     j:=j+1;
     go to contoar1;
     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;
```

```
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.
```

```
Introduceţi numărul x,x=36
Introduceţi 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 funcţiei:

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

pe intervalul [0,x].

R: Programul scris în Turbo Pascal (v. 3) realizează calculul recurent folosind facilităţile subrutinelor recursive de tip <function> cu scrierea la imprimantă a calculului efectuat pentru valoarea specificată a lui n şi 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,'lptl');rewrite(lst);
               write("introduceţi n=");
               write(lst,"introduceţi n=");
               readln(n);
               writeln(lst,n);
               write("introduceţi x="
               write("introduceţi 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 şir de caractere, media aritmetică a valorilor din şir şi poziţiile elementelor din şir a căror valoare este mai mare decît valoarea primului element al şirului.

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,'lptl');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];
```

```
 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,"Poziţiile din şir 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
Poziţiile din şir 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
```

4. Să se scrie, în Basic, un program care să utilizeze fişiere 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 fişierului este formată din patru cîmpuri:

- N$ (15 caractere) nume;
- T$ (6 caractere) valoarel;
- R$ (6 caractere) valoare2;
- Ç$ (6 caractere) valoare3.

Cîmpurile nume, valoarel şi valoare2 pot fi actualizate. Tastatura lui Enter lasă cîmpul neschimbat.

Cîmpul valoare3 se calculează folosind cîmpul valoarel sau cîmpul valoare2 (cînd acesta este strict pozitiv).

Se pot crea maxim 100 înregistrări.

La sfîrşitul prelucrării se tastează 0 pentru NR.ÎNREG.

b) O altă variantă de utilizare a unui fişier 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:CLOSE1
110 END
```

Înregistrarea fişierului este formată din patru cîmpuri:

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

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

valorile din cîmpul de valoare3 şi se afişează înregistrările în grupe de cîte 20.

c) În continuare prezentăm un program ce utilizează două fişiere 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 fişiere random. În descrierea cîmpurilor sînt folosite componente vectoriale.

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

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

```
2    REM PROGRAM DE CALCUL A INTEGRALELOR CU METODA
3    REM DREPTUNGHIURILOR,TRAPEZELOR ŞI 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 FUNCŢIEI
101  REM ŞI A DERIVATEI
102  CLS
110  PRINT "AŢI DEFINIT FUNCŢIA? <D/N>"
120  INPUT Q$
130  IF Q$="D" THEN GO TO 1000
140  PRINT "TASTAŢI LA TERMINAL LINIA:"
150  PRINT "10 DEF FNF(X)=<FUNCŢIA DORITA>"
160  PRINT "CALCULAŢI VA ROG DERIVATA FUNCŢIEI"
170  PRINT "TASTAŢI APOI LINIA:"
180  PRINT "20 DEF FNF1(X)=<FUNCŢIA DERIVATA>"
190  PRINT "CEREŢI ACUM EXECUŢIA PROGRAMULUI"
200  STOP
1000 REM RUTINA DE DEFINIRE CONDIŢII
1001 CLS
1002 INPUT "AŢI DEFINIT INTERVALUL ŞI DIVZIUNILE? <D/N>";V$
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 "ALEGEŢI 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 "DORIŢI SA CONTINUAŢI? <D/N>";C$
1240 IF C$="D" THEN GO TO 1000
1250 STOP
2000 REM METODA TRAPEZELOR
2010 CLS
2020 PRINT "METODA TRAPEZELOR"
2021 LET E=0
2030 FOR I=2 TO N
2031 LET E=E+FNF(X(I))
2040 NEXT I
2050 LET P=E+(FNF(X(1))+FNFNX(N1)))/2
2060 LET P1=P*(B-A)/N
2070 PRINT "VALOAREA INTEGRALEI"
2080 PRINT "PE INTERVALUL |";A;",";B;"| ESTE:
2090 PRINT "|(F(X))=";P1
2100 INPUT "DORIŢI SA CUNOAŞTEŢI EROAREA? <D/N>";C$
2110 IF C$="D" THEN GO TO 2200
```

```
2120 INPUT "DORIŢI SA CONTINUAŢI? <D/N>";L$
2130 IF L$="D" THEN GO TO 1000
2140 STOP
2200 PRINT "CALCULAŢI VA ROG DERIVATA A DOUA"
2210 INPUT "DERIVATA A DOUA ESTE:";V$
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$="D" THEN GO TO 3200
3110 GO TO 1230
3200 PRINT "CALCULAŢI VA ROG DERIVATA A DOUA ŞI A TREIA"
3210 INPUT "RENUNŢAŢI? <D/N>";L$
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 fişierele specificate de utilizatori la un fişier specificat.

R: O soluţie este prezentată în programul următor şi este valabilă pentru fişiere secvenţiale de tip text.

```
#include <stdio.h>
 main()
```

```
{
FILE*fi,*fl; /*declararea pointerilor care vor fi asociaţi fişierelor
               implicate*/
int c;
static char rezfis[65]=""; /*zonă iniţializată cu spaţii care va conţine
                             numele fişierului de intrare*/
   char file[65],cc;
   fl=fopen(rezfis,"au"); /*deschiderea fişierului rezultat ca fişier
                           nedefinit ("u") şi în adăugare ("a")*/
   if(fl==NULL) /*fişierul de ieşire nu există?*/
    {printf("\n Introduceţi numele fişierului rezultat:");
     scanf("%s",rezfis); /*citeşte numele fişierului*/
     if(fl=fopen(rezfis,"au")==NULL)
       {
        printf("\n Nu pot să deschid...%s...abandon!",rezfis);
        exit(-1);
       }
    }
    while(file[0]!=EOF)
     { printf("\n Introduceţi numele fişierului de concatenat:");
       scanf("%s",file);
       fi=fopen(file,"ru"); /*deschidere în citire*/
       if(fi==NULL);
        {
          printf("\n Nu pot să deschid fişierul...%s...",
                   "încercaţi din nou",&file);
         continue; /*ca un comentariu*/
       }
       while((c=getc(fi))!=EOF) /*citeşte un caracter*/
         {
           cc=c;
           putc(cc,fl); /*scrie în fişier*/
         }fclose(fi); /*închide fişierul de intrare*/
    }
    fclose(fl); /*închide fişierul de ieşire*/
    printf("\n Operaţia s-a terminat O.K.!");
}
```

7. Să se scrie un program care transformă ecranul unui terminal PC în ecran grafic de rezoluţie 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 cerinţele 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 privinţa tipurilor de vieţuitoare ce se pot încadra în anumite criterii variabile care, satisfăcute, dau sistemului posibilitatea de a-şi îmbogăţi baza de cunoştinţe, iniţial 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)
    şterge_date_bază
    verif_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,şterge_date_bază.
run:-
    write("\n Incapabil să determin; am nevoie să actualizez baza!"),
    write("Voi reveni.\n\n"),şterge_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),
    verif_istoric(X,Y,Replica).
verif_istoric(X,Y,da):-
    asserta(xpozitiv(X,Y)).
verif_istoric(X,Y,nu):-
    asserta(negativ(X,Y)),
    fail.
şterge_date_bază:-
    retract(xpozitiv(_,_)),fail.
şterge_date_baza:-
```

```
    retract(xnegativ(_,_)),fail.
şterge_date_bază:-
    write("\n\n Apăsaţi tasta spaţiu pentru ieşire"),
    readchar(_).
vietate_este(leopard) if
    are_atribut(mamifer),
    are_atribut(carnivor),
    pozitiv(are,culori_simple),
    pozitiv(are,puncte_negre),!.
vietate_este(tigru) if
    are_atribut(mamifer) and
    are_atribut(carnivor) and
    pozitiv(are,culori_simple) and
    pozitiv(are,dungi_negre),!.
vietate_este(girafa) if
    are_atribut(cu_unghii) and
    pozitiv(are,gît_lung) and
    pozitiv(are,picioare_lungi) and
    pozitiv(are,pete_închise),!.
vietate_este(zebra) if
    are_atribut(cu_unghii) and
    pozitiv(are,dungi_negre),!.
vietate_este(stîrc) if
    are_atribut(pasăre) and
    not(pozitiv(acţiune,zboară)) and
    pozitiv(are,gît_lung) and
    pozitiv(are,picioare_lungi),!.
vietate_este(pinguin) if
    are_atribut(pasăre) and
    not(pozitiv(acţiune,zboară)) and
    pozitiv(acţiune,înoată) and
    pozitiv(are,culoare_alb_negru),!.
vietate_este(albatros) if
    are_atribut(pasăre) and
    pozitiv(acţiune,zboară),
    pozitiv(acţiune,zboară_bine),!.
are_atribut(mamifer) if
    pozitiv(are,păr),
    pozitiv(acţiune,dă_lapte),!.
are_atribut(carnivor) if
    are_atribut(mamifer),
    pozitiv(acţiune,mănîncă_carne),
    pozitiv(are,dinţi_ascuţiţi),
    pozitiv(are,gheare),!.
are_atribut(cu_unghii) if
    are_atribut(mamifer),
    pozitiv(are,ţipăt),
    pozitiv(acţiune,ierbivor),!.
are_atribut(pasăre) if
    not(pozitiv(are,păr)),
```

```
    not(pozitiv(acţiune,dă_lapte)),
    pozitiv(are,pene),
    pozitiv(acţiune,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 şi pentru orice sistem hardware. Este indicată folosirea unor tehnici de programare care să permită implementarea lor pe o gamă cît mai largă de calculatoare.

1. Se dau două segmente, S_1 şi S_2, cunoscute prin coordonatele extremităţilor. Ştiind că distanţa dintre S_1 şi 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 $x_o\in S_1$, $y_o\in S_2$ cu proprietatea că $d(S_1,S_2)=d(x_o,y_o)$ şi să se calculeze $d(x_o,y_o)$. Să se scrie programul corespunzător în limbajul FORTRAN IV.

Rezolvare (model):

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

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

Algoritmul este următorul:

Pasul 1: Se citesc datele de intrare X(I),Y(I),I=1,4 şi se calculează M_1 = $= \big(Y(2)-Y(1)\big)/\big(X(2)-X(1)\big)$, M_2 = $\big(Y(4)-Y(3)\big)/\big(X(4)-X(3)\big)$, N_1 = $Y(1)-M_1 \cdot X(1)$, N_2 = $Y(2)-M_2 \cdot X(2)$; M_1 şi N_1 corespund segmentului S_1 iar M_2 şi 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 = $\max\big(X(1),X(2)\big)$, YMAX = $\max\big(Y(1),Y(2)\big)$, XMIN = $\min\big(X(3),X(4)\big)$, YMIN = $\min\big(Y(3),Y(4)\big)$. Dacă XMIN $\leq$ XMAX cele două segmente au puncte comune şi deci distanţa este D = 0; în caz contrar D = XMIN - XMAX. Se scrie mesajul "segmentele date fac parte din drepte confundate iar distanţa dintre ele este D =" şi 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 intersecţie (XI,YI) al dreptelor generate de segmentele S_1 şi S_2. Se calculează K_1 = $\big(XI-X(1)\big) \cdot \big(XI-X(2)\big)$ şi K_2 = $\big(XI-X(3)\big) \cdot \big(XI-X(4)\big)$. În cazul cînd $K_1, K_2 \leq 0$, cele două segmente se intersectează în punctul (XI,YI), se scriu coordonatele punctului (XI,YI) şi valoarea distanţei (D=0) apoi se merge la pasul 1.

Pasul 4: Folosind subrutina DIST se determină distanţele D(I), $1 \leq I \leq 4$ de la fiecare vîrf al unui segment la cel de-al doilea segment. Apoi se calculează distanţele 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 = $\min\{D(I), 1 \leq I \leq 8\}$ şi indicele K pentru care se realizează acest minim apoi se merge la pasul 5.

Pasul 5: Se afişează valoarea distanţei dintre cele două segmente şi punctele între care se realizează apoi se merge la pasul 1.

Observaţie: Algoritmul se opreşte cînd s-a detectat EOF la pasul 1.

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

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

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

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

Subprogramul MIN(D,VD,K) calculează VD = $\min\{D(I) \neq 0, 1 \leq I \leq 8\}$ şi 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 şi 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 şi se merge la pasul 5.

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

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

Pasul 1: Se citesc variabilele M (de tip întreg) şi X (de tip real) apoi se iniţializează I = 1.

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

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

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

Rezolvare:

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

Următoarele probleme sînt propuse spre rezolvare, fără indicaţii.

3. Să se determine o strategie de rezolvare a problemei celor 8 dame (pe o tablă de şah să se plaseze 8 dame care să nu se atace reciproc) atît într-un limbaj convenţional cît şi 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 şi o varză. El are la dispoziţie o barcă în care nu au loc decît două fiinţe sau o fiinţă şi un obiect. Cum poate reuşi săteanul a transporta în siguranţă pe malul celălalt lupul, capra, varza ştiind 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 redirecteze o întrerupere (este indicată de INT 5) către un program rezident care să permită tipărirea unui ecran grafic.

7. Concepeţi o rutină care să genereze toate submulţimile cu k elemente ale unei mulţimi cu n elemente.

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

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

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

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

La graniţa dintre informatică şi matematică
Jocuri, probleme distractive, strategii

Lexicul informaticii. Paradigma calculatoarelor.

Deosebitul dinamism care caracterizează informatica se reflectă în terminologia specifică domeniului atît prin creşterea în volum cît şi prin mutaţii semantice şi relaţionale în cadrul acesteia. Limitînd investigaţia numai la nivelul sferei semantice "calculator (electronic)" se poate constata că această noţiune 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 mulţimii elementelor acestei paradigme lexico-semantice îşi are sorgintea atît la palierul lexical cît şi, în special, la cel frazeologic (sintagmatic).

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

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

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

c) precum şi a celor formate prin dublă prefixare (super-microordinator, super-minicalculator, mega-minicalculator etc.), printre care unele sînt net contrastante (mega-minicalculator). Situaţia este generată de etimologiile diferite ale celor trei termeni de bază (de menţionat 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) specialiştii domeniului, informaticienii, utilizează în exclusivitate, de la începuturile informaticii româneşti, forma calculator (primul calculator românesc, realizat în 1957, se numea CIFA - Calculatorul IFA; alt exemplu timpuriu: CET - calculator electronic tranzistorizat);

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

c) dicţionarul lingvistic cel mai răspîndit - DFX - tratează deficitar cei trei termeni, în ambele ediţii, lăsînd impresia că au semantică parţial diferită (calculator... s.n. - maşină sau instalaţie cu ajutorul căreia se efectuează automat operaţii matematice şi logice...; computer s.n. - calculator electronic, ordinator...; ordinator s.n. - calculator numeric universal...) ceea ce poate induce confuzie alimentînd predilecţia publiciştilor de a utiliza

xenisme; inconsecvenţa DEX-ului este eliminată în Dicţionarul de neologisme (ediţia 1978) prin includerea sintagmei calculator electronic la computer şi ordinator făcîndu-se trimitere la aceasta (principiul circularităţii sinonimelor);

d) sinonimia de tip sinecdocă (parte pentru întreg) prin utilizarea termenului procesor şi a derivatelor obţinute prin prefixare (monoprocesor, biprocesor, multiprocesor).

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

Elementele sferei semantice analizate constituie un dicţionar-tezaur structurat taxonomic ceea ce facilitează reliefarea relaţiilor semantice de tip sinonimie, antonimie, polisemie şi hiponimie/hiperonimie (relaţie de incluziune/apartanenţă în structuri lexico-semantice).

Deoarece terminologia domeniului se află în plin proces de formare atît ca urmare a progresului tehnic, care conduce la creşterea performanţelor şi la caracteristici tehnico-funcţionale tot mai ridicate ale echipamentelor, semantica multor noţiuni are caracter dinamic-glisant. Cîteva exemple elocvente:

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

b) minicalculatorul se definea ca un calculator de capacitate mică şi preţ redus pentru ca acum să desemneze maşini cu performanţe care odinioară erau considerate specifice calculatoarelor de capacitate medie-mare;

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

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

Dicţionarul-tezaur include cvasitotalitatea noţiunilor utilizate în literatura de specialitate şi publicaţii 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 opţionale din sintagme, forma echivalentă din limba engleză sau explicaţii suplimentare, după caz).

Dicţionar-tezaur structurat taxonomic.

1. calculator (c.)/calculator electronic/maşină 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 generaţia I-a, a 2-a, a 3-a, a 3,5-a, a 4-a, a 5-a;

c) c. de capacitate mică, medie/medicalculator, 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;

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. ştiinţific (set extins de operaţii);

b) c.b. programabil (C.B.P.) în limbaje neevoluate (notaţia poloneză inversă sau limbaj de tip algebric), în limbaje evoluate (BASIC);

c) c.b. procesor de texte/cuvinte.

6. c. ceas (brăţară) sau ceas-calculator:

a) cu funcţii identice c. de buzunar;

b) specializate: medical/biologic, dicţionar bi- sau plurilingv, agendă electronică.

7. c. specialitate funcţional:

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) şi 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 distanţă (remote processor), cu staţii terminale/satelit (remote job entry, remote batch);

f) terminal inteligent/terminal programabil/terminal pentru achiziţie /prelucrare date (microcalculatoare).

8. c. universal şi 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, spaţiale, aviaţie);

d) c. naval;

e) c. radar;

f) automate de şah;

g) c. muzical.

9. s. mono/uni-calculator, mono- sau multiprocesor:

a) s. monocalculator/simplex/uniprocesor/unicalculator;

b) s. biprocesor sau bi-microprocesor, cu procesoare identice sau diferite/complementare;

c) s. monocalculator multiprocesor sau multi-microprocesor.

10. s. mono/multi-flux de instrucţiuni/date:

a) s. cu un singur flux de instrucţiuni şi un singur flux de date (Single Instruction Single Date stream - SISD)/sistem serial (cauclatoare uzuale);

b) s. cu multiflux de instrucţiuni şi un singur flux de date (Multiple

Instruction Single Date stream - MISD)/supercalculator/s. pseudo-multiprocesor (monoprocesor multi-UAL cu prelucrări paralele diferite); tipuri: conductă (pipeline) cu UAL înseriate şi uni/multifuncţional cu multi-unităţi funcţionale de tip static/dinamic;

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

d) s. cu multiflux de instrucţiuni şi multiflux de date (Multiple Instruction Multiple Data stream - MIMD)/s. multiprocesor porpriu-zis/s. strîns cuplate; tipuri: cu magistrală unică, paracomputer/ultracomputer/cu memorie comună, cu comutator încrucişat, cu memorii multipoartă, s. multitask şi 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 (generaţia a 5-a), supercalculator inferenţial (generaţia 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 configuraţie extinsă/completă formate dintr-un c. principal, un procesor frontal, un procesor dorsal şi un procesor de rezervă;

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

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

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

a) componente principale: maşină inferenţială paralelă pentru rezolvarea problemelor, maşină pentru baza de cunoştinţe, maşină de interfaţă inteligentă;

b) maşini funcţionale pentru: comunicaţii, conducerea proceselor, service, asistare utilizator, simulare, calcule numerice ultrarapide, date de tip abstract, BD relaţionale, manipulare simbolică, euristică;

c) supercalculator inferenţial, maşină personală inferenţială;

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

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

a) c. abstract/maşină abstractă/maşină 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 încrucişate

Se consideră o mulţime $A \neq \Phi$ finită numită alfabet şi $\pi \notin A$ un element numit punct negru. Definim:

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

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

Se notează $w = (w_1,w_2,\ldots,w_p) = w_1w_2\ldots w_p$.

Dacă $w = w_1w_2\ldots w_p \in V$ atunci $|W|=p$ reprezintă lungimea lui w.

Fie $D \subset V$, $D \neq \Phi$ o mulţime finită numită dicţionar. Fie D_T, D_N două submulţimi $\subset D$ cu proprietăţile $D_T \cup D_N = D$ şi $D_T \cap D_N = \Phi$, mulţimi numite dicţionar tematic, respectiv dicţionar matematic.

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

Definiţie:

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

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

Definiţie:

$w = w_1w_2\ldots w_p$ este inclus în K ($w \subset K$) dacă:

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

astfel încît:

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

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

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

Definiţie:

K are licenţa dacă $\exists w \subset K$ astfel încît $w \notin D$.

Definiţie:

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

Fie $\sim$ o relaţie de echivalenţă în D numită relaţie de familiaritate iar $D/\sim$ mulţimea claselor de echivalenţă.

Definiţie:

K are cuvinte din aceeaşi familie dacă:

$$\exists\ w_1,w_2 \subset K,\ w_1 \neq w_2 \text{ şi } w_1 = w_2.$$

Definiţie:

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 $i \in \{i-1,i+1\} \cap I_n$ şi $j \in \{i-1,j+1\} \cap I_n$.

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

Definim funcţia: $\varphi_T : D \to \mathbb{N}$,

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

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

Definiţie:

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

Definim funcţia: $\varphi_p : A \cup \{\pi\} \to \{0,1\}$, 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.

Definiţie:

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

Definiţie:

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

$(i,j) \neq (k,l)$ sînt vecine dacă:

a) $\{i,j\} \cap \{l,n\} \neq \phi$ şi $\{k,l\} \cap \{l,n\} \neq \phi$

sau

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

Definiţie:

K are semiînchideri dacă:

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

a) $\forall r = 1,\ldots,s-1$, $K^t_{i,r\ j,r}$ vecin cu $K^t_{i,r+1\ j,r+1}$ şi $K^t_{i,s\ j,s}$ vecin cu $K^t_{i,1\ j,1}$;

b) $r-1 \leq \text{card}\{K^t_{i,r\ j,r} | K^t_{i,j} = \pi\} \leq r$.

Fie $d_T, d_p \in [0,1]$.

Definiţie:

K este careu corect încrucişat dacă şi numai dacă:

L) K nu are licenţe;

B) K nu are bare;

F) K nu are cuvinte din aceeaşi familie;

T) K are cel puţin 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 mulţimi unde $\forall x \in D$, S_x reprezintă mulţimea semnificaţiilor (definiţiilor) lui X.

Fie $D' \subset D$. Definim $S_{D'} = \bigcup \{y | y \in S_x\}$ un sistem de definiţii pentru D'.

Evident $\bigcup_{w \subset k} w \subset D$.

Notăm $S = S \cup \bigcup_{w \subset k} w$ un sistem de definiţii pentru K.

Definiţie:

$\sum = (K,S)$ se numeşte un careu de cuvinte încrucişate dacă:

a) K este corect încrucişat;

b) S reprezintă un sistem de definiţii pentru K.

O problemă de ordonare în teoria grafurilor

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

Fiind date $x,y \in E$ numim drum de la x la y o succesiune $\{x,b_1\},\{a_1,b_2\},\ldots \{a_n,y\} \in V$ dacă $\forall i = \{1,\ldots,n\}$ $b_i = a_i$. 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 $E' \subseteq E$ şi $V' \subseteq V$ şi cel puţin una din incluziuni este strictă (scriem $G' \subsetneq G$). $G' \subseteq G$ se numeşte componentă conexă a lui G dacă G' este conex şi este maximal cu această proprietate.

Un graf G este un arbore dacă este conex şi graful G' obţinut 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 mulţimea arborilor care formează o pădure relaţia notată $\sim$. Fie $A_1 = (V_1,E_1)$ şi $A_2 = (V_2,E_2)$ doi arbori ai unei păduri cu rădăcinile respectiv a_1 şi a_2 de nume numel şi nume2.

Spunem că $A_1 \sim A_2$ dacă şi numai dacă prin definiţie $|V_1| > |V_2|$ sau dacă $|V_1| = |V_2|$ şi numel > nume2, unde > este relaţia de mai mare pe mulţimea numerelor naturale şi > este relaţia de ordine lexocografică naturală.

Definiţie: Două structuri de date prezentate sub formă de păduri sînt echivalente dacă, pentru o specificaţie corectă, elementele desemnate sînt aceleaşi.

Pentru a avea elementele definite fără ambiguităţi facem următoarele ipoteze:

1. Rădăcinile tuturor arborilor care formează pădurea sînt distincte;

2. Pentru orice nod subarborele care are drept rădăcină nodul respectiv nu conţine un alt nod cu acelaşi nume;

3. Pentru orice nod neterminal fiii săi au nume distincte.

Fie $A_1,\ldots,A_n$ arbori şi $A_1 = (V_iE_i)$, $\forall i = \{1,\ldots,n\}$ şi fie pădurea P formată cu aceşti arbori $P = (\bigcup_{i=1}^{n} V_i, \bigcup_{i=1}^{n} E_i)$

Propoziţia 1: Relaţia $\sim$ este o relaţie de ordine totoală.

Demonstraţie: Presupunem $A_i \sim A_j$ şi $A_j \sim A_k$ $i \neq j \neq k, i \neq k$.

Putem avea următoarele cazuri:

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

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

(iii) $|V_i| > |V_j|$ şi $|V_j| = |V1| \Rightarrow |V_i| > |V_k| \Rightarrow A_i \sim A_k$;

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

Deoarece $A_i \sim A_j \Rightarrow$ nume (A_i) > nume (A_j)

$A_j \sim = A_k \Rightarrow$ nume (A_j) > nume (A_k)

Din tranzitivitatea relaţiei de ordine lexicografică rezultă că avem: nume (A_i) > nume (A_k) **.

Din relaţiile * şi ** rezultă $A_i \sim A_{1,k}$.

În mod analog dacă $A_i \sim A_{k,j}$ rezultă că nu putem avem şi $A_y \sim A_i$. Din

această propoziţie şi folosind ipoteza 1 rezultă că putem face o renumerotare a arborilor astfel încît să avem:

$$A_1 \sim A_2 \sim \ldots \sim A_n.$$

Fie un arbore A. Graful obţinut prin suprimarea nodului rădăcină şi a muchiilor adiacente cu aceasta formează o pădure de componente $B_1, B_2, \ldots, B_k$.

Conform rezultatelor anterioare şi ipotezei 3 putem face o renumereotare a componentelor B_j astfel încît $B_1 \sim B_2 \sim \ldots \sim B_k$.

Propoziţia 2: Fie o pădure P de componente $A_1, A_2, \ldots, A_n$. Fie pădurea P' de componente $A'_1, A'_2, \ldots, A'_n$ obţinută prin aplicarea paşilor 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_1 ordonarea făcîndu-se conform relaţiei $\sim$;

II. Se ordonează componentele pădurii P conform relaţiei $\sim$.

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

Demonstraţie: Fie o specificaţie Y_1 of Y_2 of ... of Y_k pe care o presupunem corectă. Din definirea specificaţiei rezultă că există şi este unic un $1, 1 \leq d \leq n$ astfel încît $Y_k \in A_1$.

Fie G_k subarborele lui A_1 de rădăcine Y_k. Deoarece P verifică dacă rădăcina arborelui A_1 este rădăcină a arborelui $A'\pi_k(1)$ în P' deci există un unic $1', 1' = \pi_k(1)$ astfel încît $Y_k \in A'_{1'}$.

Fie G'_k subarborele lui A'_1, de rădăcină Y_k.

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

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

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

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

Fie G_{i-1} subarborele lui B_l de rădăcină Y_{i-1} şi G_{i-1} subarborele luf $B'_{l'}$ de răcăcină Y_{i-1}.

Această inducţie inversă demonstrează propoziţia dacă arătăm că $B_1, B_2, \ldots$ B_s satisfac ipotezele 1., 2. şi 3.

Deoarece ipotezele 2. şi 3. se referă la un nod din P şi orice nod din T este un nod din P rezultă că ele sînt verificate şi 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 Yi sînt rădăcinile arborilor $B_1, B_2, \ldots, B_s$ şi sînt distincţi, rezultă că T verifică şi ipoteza 1. Acest lucru complementează demonstraţia propoziţiei.

Curier editorial

Aşadar, să inaugurăm prin intermediul revistei primul nostru contact cu cititorii. Cum nu putem aprecia în acest stadiu de început opiniile dorinţele sau sugestiile dumneavoastră vă vom veni noi în întîmpinare prin cîteva propuneri şi informaţii utile.

Astfel, să intitulăm prima rubrică a curierului editorial:

1. Sugestii şi servicii software

În cadrul acestei secţiuni aşteptăm de la cititori un contact de propuneri şi oferte de colaborare sau, de ce nu a unei rubrici de mică publicitate informatică.

Astfel anunţăm cititorii că la sediul redacţiei pot primi informaţii, sfaturi şi ajutor în obţinerea sau cunoaşterea unor pachete software sau cărţi privind lucrul pe microcalculatoarele personale.

- Utilitare: Laplink III (pachet de comunicaţie); Lotus Magelan; Mirror; Sideways; Freelance; Xtree; Norton Utilities; C-Toolbox;
- Software integrat: Lotus; Symphony; Framework III;
- Procesare de text: Word Perfect 4.1. şi 5.1.; WordStar 2000 Plus 3.0.;
- Pachete de grafică: Vermont Vision 1.0; Design CAD 3-D;
- Baze de date: Oracle 5.0 şi 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;
- Reţele locale: Novell; 3+Comm; Token Ring; Crosstalk Lan.
- Pachete contabil-finnciare: Sun Account; SBT.

O altă rubrică s-ar intitula:

2. Biblioteca revistei

Aici, redacţia va pune în timp la dispoziţia cititorilor o bibliotecă cu cărţi şi reviste de specialitate sub forma lor fizică sau pe suport magnetic.

Sperăm ca şi revista să poată fi obţinută pe suport magnetic, permiţînd celor ce n-au citit-o sau vor să primească numai anumite articole să închirieze suportul magnetic cu problemele revistei. Propunem aici şi contribuţii personale ale cititorilor pentru îmbogăţirea bibliotecii cu documentaţie şi software.

O altă rubrică o vom numi:

3. Corespondenţe şi schimburi

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

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

Şi, în sfîrşit, o ultimă rubrică este:

4. Sfaturi şi servicii hardware

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

specialiştilor noştri sau prin sugestiile dumneavoastră.

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

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

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

C U P R I N S

ASSOCIATION OF SPECIALISTS
IN DATA BANKS AND MICROSYSTEMS
BUCAREST
STR:TURDA;NR.114,bl35,sc.B,ap 71

PRESENTATION OF OUR SOCIETY AND SERVICES

OUR SOCIETY

It was fondated in january 1990 with the name mentioned above.
The president of the society is Mr. prof.**ADRIAN NEGRU**,mathematicien,and with 10 years of experience in the field of microsystems,data bases and artificial inteligence.
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 then 200 specialists in mathematics,informatics, management and marketing,our society had developed the following sec - tions of its present and future activity:

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

- -editing informatics magasines,books and studies mainly concerning the microsystems (PC) family
- -designing and inplementation 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 diferent branches of informatic activity offering the following services:

- -instalation of microsystems(PC)
- -project development at request
  software testing and debugging
- -customised software packages and maintenance
- -lectures in programming languages,artifficial inteligence, diffrent 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 instalation and testing for PC LAN
- -consululting and diagnosys for hardware bugs

2. ACTIVITY

Our society has its personal technical resourses 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.

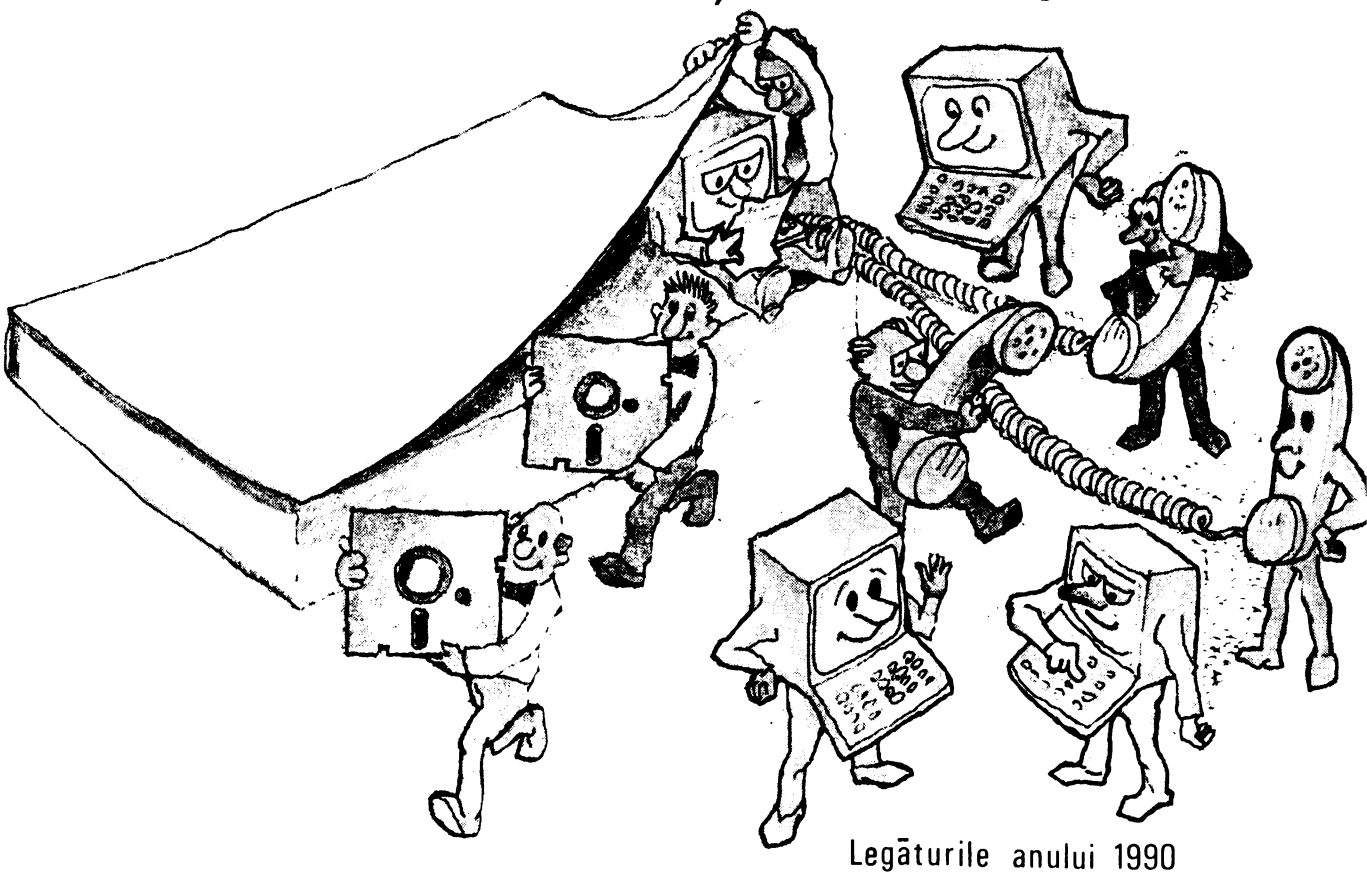
PRESEDINTE

ADRIAN NEGRU

VICEPRESEDINTE

ALEXANDRU BABIN

Imaginea lumii informaționale contemporane



Legăturile anului 1990

IN ACEST NUMAR :

LEI 25

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