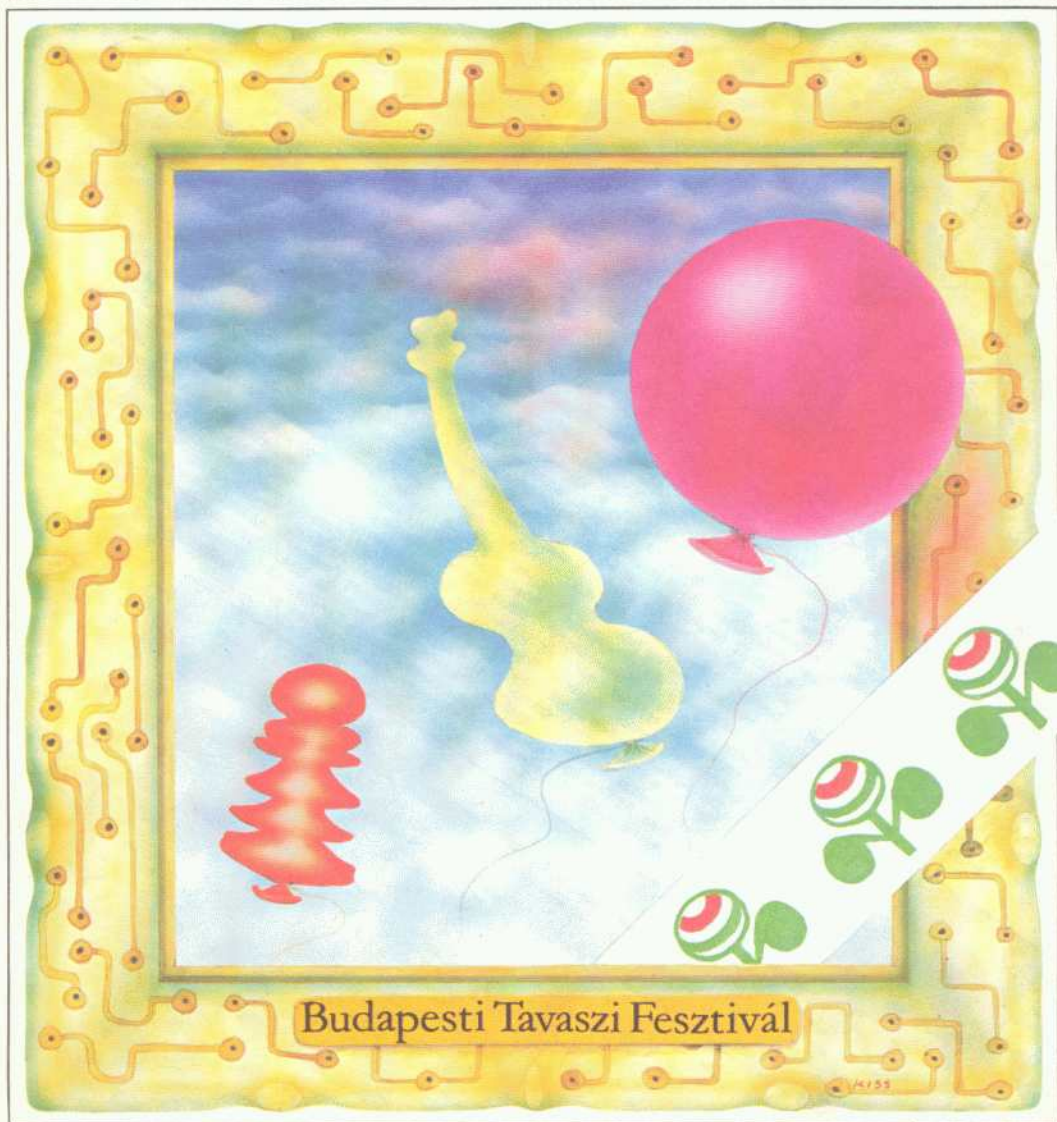


mikro számítógép magazin

Ára: 30 Ft



Budapesti Tavasz Fesztivál

IV. Országos Mikroszámítógépes Találkozó (március 17—22.) 1989/3.

M, '89



IV. Országos Mikroszámítógépes Találkozó

március 17-22.

Im már hagyományosan, negyedéves rendezés meg a Neumann János Számítógéptudományi Társaság a kiállításal egybekötött rendezvény sorozatát, az Országos Mikroszámítógépes Találkozót. A Budapesti Fesztivál keretén belül megtartandó '89-nek a szlogenje: „A számítástechnika az idegenforgalomban”.

A Hobby Computer Club bemutatja új alkotásait és tagjai szívesen adnak tanácsot az érdeklődőknek. Jelen lesznek az NJSZT területi szervei és sok híres nemzetközi számítástechnikai szakember is. A korábbi évekhez hasonlóan a versenyek és programok idén is nagy számban tartandók, mint az oktatási programok egyéni és országos csapatversenyek, valamint a házi építésű számítástechnikai eszközök vetélkedése. Az Országos Műszaki Múzeum kiállítása is megje-

petést tartogat az adatrógzító gépeket bemutatásával. A rendezvények közül kiemelkedik a Közművelődési Konferencia, amelyen többek között megismerhetjük a MUPID rendszer hazai felhasználásának lehetőségeit. A számítástechnika felhasználása az egészszégyűben idén sem maradhat el Ehhez kapcsolódik a Munkahely és otthoni kiállítás, amely a mozgássérültek mutat példákat. Az ifjúsági napon a Nemes Tihmer Országos Középiskolai Számítástechnikai Tanulmányi Verseny első fordulóján szü-

letett legjobb megoldásokat mutatják be a versenyzők. A sakkzóra és a brijdzselők idén is sok meglepetés vár: új gépeket és programokat próbálhatnak ki.

A kiállító cégek is sok új dongságot mutatnak meg és általában minden meg is vásárolható. Az NJSZT standján olvaszhatnak a Mikroszámítógépes Magazin regebbi számait, így teljessé tehetik gyűjteményüket. Az Országos Mikroszámítógépes Találkozót idén sem szabad kihagyni. Várjuk az érdeklődőket a BNY területén, a 25-ös pavilonban!





mikro számítógép magazin

A NEUMANN JÁNOS SZÁMÍTÓGÉPTUDOMÁNYI TÁRSASÁG LAPJA

A szerkesztőbizottság
vezetője:
Kovács Győző

A szerkesztőség
munkatársai:

Bakos Tamás
(programozástechnika)

Broczkó Péter
(hírek)

Kovács Győző
(levelezés)

Nagy Imre
(tanuljunk együtt)

Petróczy Judit
(könyvek)

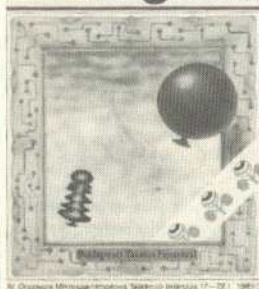
Pinke György
(NJSZT, alkalmazások)

Soltészné Vizi Zsuzsa
(tervezőszerkesztő)

Simonyi Endre
Szabenszki Sándor
Szulyovszky Csaba
Tamásné Lakó Erika
Terebessy Ákosné
Varga János

Címképünk:
Kiss Ilona munkája

mikro
számítógép
magazin



Felelős szerkesztő:
Könyves Tóth Pál

Szerkesztőség:
1027 Budapest, Fő u. 68.
Telefon: 154-250

Levélcím:
1371 Budapest
Pf. 433

Kiadja:
MTESZ Neumann János
Számítógéptudományi Társaság
1054 Budapest, Báthori u. 16.

Levélcím:
1368 Budapest 5. Pf. 240

Telefon: 329-349

Felelős kiadó:
Tóth Istvánné ügyvezető
főtitkárhelyettes

Terjeszti a Magyar Posta
Előfizethető a hírlapkezesítő
hivataloknál
és a Posta Hírlap-előfizetési
és Lapellátási Irodáján
(1900 Budapest XIII.,
Lehel u. 10/A)
vagy átutalással a 215-96 162
pénzforgalmi jelzőszámra.

Megjelenik havonta.
Egy szám ára 30,- Ft
Előfizetési díj,
egy évre 360,- Ft
félfévre 180,- Ft
Külföldön terjeszti
a Kultúra,
1389 Budapest, Pf. 149
és a Magyar Média
1932 Budapest, Pf. 279
88-1552



Szikra Lapnyomda
Budapest (89-0013)
Felelős vezető:
CsönDES Zoltán vezérigazgató

INDEX: 25 629
ISSN 0236-6088

TARTALOM

2	μ'89, a IV. Országos Mikroszámítógépes Találkozó Feladatok — megoldások
10	Többszörösen ad, aki ma ad
12	A Scanntronik nem ült a babérijain
27	Rendszerfejlesztési eszközök
28	Merre tart a világ?
31	Systek '88
34	Olvastunk...
42	Mit kell tudnunk az adóról?
47	Adok — veszek — cserélek

TANULJUK EGYÜTT!

3	A Pascal rejtjelmei
5	Print Screen lemezre
7	Polinomok szorzása és osztása

CSIPEGETŐ

14	Statistikai tömörítés
14	Sorscsepés (ellen) — használati utasítás
15	A főlősleges — Színváltó
16	Videomonitor a Junoszy-402 BC-ből
17	„Kormányváltás”
17	TOP-lista

PROGRAMOZÁSTECHNIKA

18	A számítógépek motorja II.
20	Programozási fogások és melléfgások
21	Nyomatás „egy az egyben”

ENTERPRISE

23	Programmódosítás futás közben
23	Rekurzív növény
24	Megkérdeztük az Enterprise-ről
25	Mielőtt valaki hamar „hibázna”
26	Mi a manó?

μPROGRAMOK

36	Nagyobb karakterméret!
38	A Tasword módosítása
38	Listázás elleni védelem

μKLUB

39	Adom a magyarázatot!
39	Egy- és kétsorosok
40	A The Newsroom magyar változata I.

SAKK

44	Fejlődés
----	----------

AZ OLVASÓ ÍRJA

KÖNYVEK—HÍREK—ÉRDEKESSÉGEK

3

14

18

23

36

39

44

46

48

Ennek a szerkesztésig cikknek a megírása előtt kíváncsiságból elolvastam az elmúlt években írt márciusi „mikros” cikkeimet, már csak azért is, hogy még véletlenül se írjak valami hasonlót az ideai „ünnepe” alkalmából.

Nem tudom, hogy ki hogy van vele, nekem a μ '89 is — remélem — ünnepe lesz. Azt is remélem persze, hogy nem csak nekem, de sokunknak az aktuális márciusi egy hét nem egyszerűen egy a sok számítástechnikai kiállítás közül, hanem az a kiállítás, amely a szakma együtt vonul fel az amatőr mozgalommal, ahol a számítástechnikával foglalkozó diákok kiállításai, bemutatói ugyanolyan fontosak, mint egy új gép vagy egy csodaszoftver bemutatója.

μ '89

A IV. Országos Mikroszámítógépes Találkozó

Arra is büszkék vagyunk, hogy az eddigi mikroszámítógépes találkozókra jelentősen támogattuk a kisvállalkozókat, főleg a kevésbé tőkeerős szervezeteket, hiszen számukra is kell valahol a bemutatkozáshoz megfelelő alkalmat biztosítani. Azt is elmondhatom, hogy ennek az országos találkozóknak már követője is van, tudniillik 1988 őszén Bécsben rendeztek egy ifjúsági konferenciát és kiállítást, s az erre kapott meghívólevelében azt írták, hogy a vállalkozásra a μ '88 inspirálta a rendezőket.

Az egyik osztrák barátom azt mondta, ha nem látja, akkor nem hiszi, hogy a Budapesti Tavasz Fesztiválon nem csak muzsikát hallgatnak, balettet néznek és színházba járnak az emberek, az Utazás kiállításra pedig nem csupán a nyári szabadságukat igyekeznek megtervezni, de nagyon sokan eljönnek a „mikróra” is, ahol napokon keresztül tanulmányozzák a hardvert és szoftvert amatőrök alkotásait. Megnézik, hogy hova jutott, illetve fejlődött-e az iskolai számítástechnika, részt vesznek kulturális rendezvényeken, sakkznak a számítógépekkel, meghallgatják és megpróbálják megérteni a számítógépes zene bonyolult és érdekes mikrovilágát.

Mi, a μ '89 rendezői akkor vagyunk igazán elégedettek, ha a pavilonunk reggeltől estig tele van látogatókkal, akik nem csak a kiállításon nézelődnek, a számítógépes boltokban vásárolnak, de beülnek az előadásokra és részt vesznek a μ '89 más programjain is.

Az idén is reméljük, hogy a szakmaért és az oktatásért felelős főhatóságok, de talán más kormányzati szervek is odafigyelnek erre a találkozóra, és némi anyagi támogatást is adnak ennek a

nyereségesre tervezett, sőt a váratlan kiadások miatt gyakran veszteséges rendezvénynek. Az elmúlt években még kaptunk például pályadíjakat az IpM-től, a KSH-tól és az OMFB-től.

A μ '89 és általában az eddigi találkozók szerény árbevételűek a kiállítóktól származik, ezért különösen hálásak va-

gyunk a kiállítás minden résztvevőjének, de különösen azoknak, akik eddig minden alkalommal előttek bemutatni az informatikában képzetlen felhasználóknak tervezett szoftver- és hardvereszközöket.

Az idén először szerződést kötöttünk a Comexpo-val, a Compair szervezőjével, és így már az ideai mikroszámítógépes találkozót és a Compairt is közösen rendezzük meg. Az együttműködés eredményeképpen valószínűleg csökkenteni tudjuk a szervezés költségeit, hatékonyabbá tesszük a propagandamunkát, és nem utolsósorban az egész évben folyamatos kapcsolatot tartunk a kiállítóinkkal, akiknek a kívánságait már az előkészítő munka során igyekszünk figyelembe venni. Szeretnénk a „mikrót” továbbfejleszteni. Nem kisebb tervről álmodozom, mint egy közép-európai nagy találkozóról, ahol a szakma és az amatőrismus együtt jelenik meg, kicserélik a tapasztalatokat és bemutatják a számítástechnika alkalmazását nemcsak a technikában és a tudományban, hanem az irodalomban, a zenében, a képzőművészetben és az élet sok más területén is. Egy ilyen nagyszabású terv nyilvánvalóan csak népes „szponzorcsapat” támogatásával valósítható meg, akik ezt a tervet nem csupán elfogadják, de sokféle módon segítik is. Szeretnénk a szervezők csapatát is áldozatkész, ügyes kollégákkal erősíteni, akik vállalják, hogy a napi munkájuk mellett tárgyalnak, leveleznek, ügyeket intéznek és közben tudják, hogy a munkájukat legfeljebb szerény jutalommal fogjuk honorálni.

Végül olvassa ezt az írást, azt hiszem, hogy egyetlen dologról nem irtam még: a μ '89 programjáról. A kedves látogató már megszokhatta, hogy állandó, de azért mindig valami újat hozó programjaink, mint például az amatőr kiállítás,

„Puszták pástorai — mit is tudunk?

Községi kiskiscolát jártam magam is csak — mit tudok én?

Csak bibliát tudok, meg öreg meséket, csak azt, hogy a zab mit zümmög a szelnek...

S még ezt: ünnepnapokon húzni a harmonikát.”

(Szergej Jeszenyin: Falusi szertartáskönyv Fordította: Rab Zsuzsa)

az iskolai programok versenye, a játékos programok versenye, a diák-pedagógus fórum, a sakkzó gépek szimultánja, a szoftver- és hardverbörze, a számítógépes brídzes és más programok mellett mindig van legalább egy, különleges érdeklődésre számot tartó esemény. Az idén kettő is lesz.

Az egyik a Tourcomp szimpózium és kiállítás, ami elsősorban a turizmus szakembereinek szól, akik az informatika turisztikai alkalmazását mutatják be, tehát szállodai információs rendszereket, a turistairodákban alkalmazott programokat, speciális célgepeket, egyszerűen mindazt, amit a mai informatika a turizmus támogatására alkotott.

Nem kevésbé lesz érdekes a párizsi UPIC szerelése. A μ '89-en a sokak által jól ismert görög—francia zeneszerző, Xenakis zenei laboratóriumát látjuk vendégül, a budapesti francia követség galánis támogatásával. Az UPIC-ben dolgozó zeneszerzők műveiket nem a szokásos módon köztávza komponálják, hanem rajzolják, majd a rajzot transzformálják át zenevé egy speciális szoftver segítségével.

Az UPIC szakemberei a μ '89 alatt speciális tanfolyamot tartanak azoknak a jelentkezőknek, akik szeretnék a zene szerzésnek ezt a különös módját megtanulni. A „legjobb tanulók” kompozíciót és persze a professzionális francia, illetve magyar zeneszerzők egy-két darabját is a záróhangversenyen fogjuk bemutatni. Úgy tervezem, hogy a hangverseny sem lesz szokványos: azt szeretném ugyanis, hogy a zeneszerzők ne csak bemutassák, hanem el is magyarázzák kompozíciójuk lényegét, és így egy kicsit bepillantassunk az alkotás folyamatába is.

Kedves olvasóink, ez jutott eszembe a μ '89-re készülve. Az idén is csak azt kérem, amit az eddigi három mikroszámítógépes találkozó alkalmával kértem: jöjjenek ki a BNV-re, nézzenek meg bennünket, s az „ünnepnapokon húzzuk együtt a harmonikát”, mert az idén is igaz szeretettel várja önöket:

Kovács Gyöző



A PASCAL REJTELMEI

Az eljárások írása – ciklusszervezés

4. Második programunk

Sorozatunk első részében (lásd az 1989/1. számunkat) már szó volt a Pascal egyik erősségéről, a strukturált programozást támogató eljárások használatáról. Eljárásokat első programunk egyes változataiban is használtunk, de most másról, a felhasználó által írt eljárásokról lesz szó. A megoldandó feladat ugyanaz, mint amivel már foglalkoztunk: két bemenő adaton elvégezzük a négy alpműveletet.

Ezúttal viszont a feladatot megoldó programot két önálló részre bontjuk, s ezeket egy-egy eljárással valósítjuk meg, bemutatva az eljárás megírásának elveit, szintaxisát, illetve azt, hogyan fűzhető össze az eljárások programmá. Ezenkívül csak egyetlen új Pascal ismeretet közlünk, a magyarázatok – kommentek – programba illesztésének szabályát.

Egy Pascal programban tetszőleges helyen és mennyiségben írhatunk magyarázatokat a `|` jelek közé helyezve. Mivel ezek nem utasítások, végük jelzésére nem szükséges a `;` karakter.

Ezek után lássuk a lényegelt! Igaz ugyan, hogy verébre fogunk ágyúval löni, és programunk is terjedelmesebb lesz a kelletténél, de most ne a program rövidségére koncentráljunk.

A `gyak1_3` nevű programot két részre bontjuk. Az első rész kiírja a futtatási screen fejlécét, a második elvégzi a feladat lényegét: az adatbevitelt, a számításokat és az adatkivitelét. Az egyes részeket egy-egy eljárásként valósítjuk meg, a programtörzs pedig elvégzi a két eljárás hívását. A forrásprogram listáját a *10. ábrán* láthatjuk.

A program deklarációs része most nem csak változókat, hanem eljárásokat is deklarálnak.

Az eljárások feje az eljárás azonosítást szolgálja:

procedure fejlec;
illetve

procedure törzs;

Az eljárásoknak példánkban nincs deklarációs része. Annak ellenére, hogy az eljárások változókat használnak, ezeknek az eljárásokban való dek-

larálására nincs szükség, mert az eljárásokban csak a főprogram változónevei szerepelnek. Ezek az úgynevezett globális vagy más néven általános érvényű változók. Általában az eljárásokban előfordulhatnak olyan változók is, amelyeket a főprogramban vagy az adott eljárást hívó másik eljárásban nem deklaráltunk. Ezek az úgynevezett lokális, azaz helyi változók. Ezeket az adott eljárásban természetesen deklarálni kell. A globális és lokális változók problémáival, mivel a téma jelentősége a Pascalban igen nagy, a későbbiekben – amikor ezeket programjainkban alkalmazni fogjuk – részletesebben is foglalkozunk.

Az eljárások törzsét – a végrehajtható részt – a **begin** és az **end** alapjelek határolják.

A főprogram végrehajtható része a két eljárás hívásából áll. Az eljárások hívása most – mivel a hívó program és az eljárások között adatátadás nincs, mert minden változó globális, azaz a program bármely részéből elérhető – egyszerűen az eljárás nevének a „leírásából” áll.

Példánk is bizonyítja, hogy az eljárások alkalmazása a program végrehajtható részét nagymértékben egyszerűsíti. Ennek ugyan az az ára, hogy a teljes program látszólag bonyolultabbá válik, vegyük azonban figyelembe, hogy az eljárásokra bontás – ami valóban példánkban is szükséges – lehetővé teszi a programok egyes részeitnek egymástól független megírását, a már meglévő eljárások alkalmazását. A futtatáskor keletkező screent a *11. ábra* mutatja.

5. Harmadik programunk

Ebben a részben a programozástechnika egyik legfontosabb kérdésével, a ciklusszervezéssel foglalkozunk. A Pascal ebben a tekintetben a háromféle ciklusszervezési lehetőségével – még ha az **if...then...goto** típusú ciklusokat figyelmen kívül hagyjuk is

– szinte összkomfortot kínál felhasználóinak.

Először a **for...next** típusú ciklusok Pascal megvalósítását mutatjuk be, amit persze a BASIC-ben jártasak már ismernek. Újdonság lesz ezenkívül még egy „apróság” is, az **INKEYS** Pascal megfelelője, azaz a program várakoztatása egy billentyű lenyomásáig. Ez a **read** eljárás egyik, tulajdonképpen már említett lehetőségével valósítható meg:

read(kbd, változo);

ahol a **kbd** a fájlnev, a **változo** a változónev. Ha a **változo-t char** típusként deklaráljuk, az értelme a következő. A program a **kbd** fájlba – ez a billentyűzetet jelenti a Turbo Pascal számára – beolvas egy karaktert. A lenyomott billentyűnek megfelelő karakter a képernyőn nem jelenik meg, azaz nincs echo.

5.1 A for...to (downto)...do utasítás

E rövid kitérő után térjünk rá a ciklusszervezésre! Az utasítás formája a következő:

for ciklusváltozó = kezdőérték to végérték do

illetve

for ciklusváltozó = kezdőérték downto végérték do

ahol a **for** a **to (downto)** és a **do** lefoglalt szavak. A ciklusváltozó értéke a ciklusmag minden egyes végrehajtása után automatikusan változik: **to** esetén 1-gyel nő, **downto** esetén 1-gyel csökken. A ciklus lejár, ha a ciklusváltozó értéke to esetén 1-gyel nagyobb, **downto** esetén 1-gyel kisebbé válik a végértéknél. Figyelembe kell venni, hogy a ciklusutasítás csak a következő utasításra vonatkozik, azaz a ciklusmag csak egyetlen utasításból állhat. Ez a tény csak látszólag jelent korlátozást: ez az egy utasítás ugyanis a **begin** és az **end** alapjelek segítségével képzett, tetszőleges bonyolultságú, összetett utasítás is lehet. A ciklusok gyakorlatilag végtelen mélységben egymásba „skatulyázhatók”.



A programlistát a 12. ábrán láthatjuk. A program egyébként annyiszor oldja meg a már többször elvégzett feladatot, amekkora értéket adunk a ciklusszam változónak a ciklusokszama nevű eljárásban.

A for...to (downto)...do utasítással szervezett ciklusok használatánál több szabályt, illetve jellegzetességet kell figyelembe venni:

- a ciklusváltozó csak integer típusú lehet;
- a kezdőértéknek és a végértéknek a ciklusváltozóval azonos vagy kompatibilis típusnak kell lennie (a típuskompatibilitás kérdésével részletesebben a későbbiekben foglalkozunk);
- ha a kezdőérték és/vagy a végérték kifejezés, ezeknek a ciklus kezdetén kiértékelhetőeknek kell lenniük;
- a ciklusváltozó értékét a ciklusmagban nem szabad megváltoztatni;
- a ciklusváltozó a cikusból való kilépéskor elveszti értékét;
- a ciklus egyszer mindenképpen lefut, még akkor is, ha a ciklusváltozó értéke تو esetén kisebb, downto esetén nagyobb a kezdőértéknél (ún. hátultesztelt ciklus: a ciklusváltozó értékét a ciklusmag végrehajtása után vizsgáljuk).

5.2 A repeat...until utasítás

A for...to (downto)...do utasítással csak akkor szervezhetünk ciklust, ha előre ismerjük a ciklusmag ismétléseinek a számát. A teljesség kedvéért megemlítjük, hogy a cikusból való kilépés feltételes programelágazással — if...then vagy if...then...else utasítással — megvalósítható. Ily módon kellően nagy lépésszámot előírva és a cikusból feltételvizsgálattal kilépve ez a probléma megkerülhető. Meg kell

azonban jegezni, hogy egy magára valamint adó programozó ilyen „gusztustalan” megoldást nem választhat.

A profzionális módszert a repeat...until utasítás kínálja. A ciklus a repeat kulcsszóval kezdődik. Ezt követi a ciklusmag. A ciklust egy until utasításban előírt feltételvizsgálattal fejezi be. Ha a feltétel teljesül, az a vizsgált változó vagy kifejezés értéke true, a ciklus lefutott, és a program az until utáni utasítással folytatódik. Ha a feltétel nem teljesül (az eredmény false), a vezérlés ismét a ciklusmag kezdetére kerül.

A repeat...until utasítással szervezett ciklust tartalmazó program listája a 13. ábrán látható.

A repeat...until utasításon kívül a programban szerepel még egy eddig nem használt „újdonság” is. Mivel a cikusból való kilépést az ESC billentyűvel kívánjuk vezérelni (ennek decimális kódja 27), vizsgálni kell a lenyomott billentyű kódját, a ch változó értékét. A stringfüggvényekkel jelenleg nem kívánunk foglalkozni, ezért erre a célra a Turbo Pascalnak egyik egyszerű lehetőségét használtuk ki. Eszerint a következő kifejezés:

```
#érték
ahol az érték egy, a 0..255 intervallumba eső decimális egész szám, egy
```

karakterkódot jelent. A programban szereplő

until ch=#27
jelentése tehát: until (addig, ameddig) a ch char típusú változó kódjának értéke (azaz a lenyomott billentyű kódja) decimális 27.

A repeat...until utasítás elvételből következik, hogy a ciklus ebben az esetben is legalább egyszer lefut, ugyanis ez is hátultesztelt ciklus.

Előtesztelt ciklust a while...do utasítással szervezhetünk. Mielőtt azonban ennek magyarázatát elkezdenénk, kitérünk egy, a programban szereplő új Pascal ismeretre, a string adattípus alkalmazására.

A stringek definiálása a Turbo Pascalban igen egyszerű:

var változónev:string[hossz]
ahol a hossz egy integer típusú szám, amely a string karakterben „mért” hosszát adja meg. Értéke 1-től 255-ig terjedhet. (Tehát a string első karakterének sorszáma nem 0, hanem 1!)

5.3 A while...do utasítás

A while...do utasítással szervezett ciklusban a ciklusmag végrehajtására vonatkozó feltétel a ciklus elején van. Ha a feltétel teljesül, azaz a vizsgált

12. ábra

```
program gvak2.1;
(ciklusok for to do)
var ciklusszam:ciklusszam;integer;
    a:byte;
var ch:char;
procedure fajlec;
begin
    clrscr;gotoxy(3,3);
    writeln('Harmadik gyakorló programunk');
end;
procedure ciklusszam;
begin
    clrscr;writeln('A ciklusok száma:');
    readln(ciklusszam);
end;
procedure torzs;
begin
    writeln;writeln('Az adatok');
    write('a=');readln(a);
    writeln;writeln('b');
    gotoxy(2,9);writeln('A művelet és eredmények');
    write('a*b=');writeln(a*b);
    write('a/b=');writeln(a/b);
    write('a+b=');writeln(a+b);
    write('a-b=');writeln(a-b);
end;
ciklusszam;
for ciklus:=1 to ciklusszam do
begin
    fajlec;
    torzs;
    writeln;writeln('Nyom le egy billentyűt');
    readln(ch);
end;
clrscr;
writeln('Program vége');
end.
```

10. ábra

```
program gvak2;
(ciklusok repeat until)
var a:byte;
procedure fajlec;
(a repeat nevű eljárás leírása)
begin
    clrscr;gotoxy(3,3);
    writeln('Harmadik gyakorló programunk');
end;
procedure torzs;
(a programtorzs nevű eljárás leírása)
begin
    gotoxy(2,9);writeln('Az adatok');
    write('a=');readln(a);
    write('b=');readln(b);
    gotoxy(2,9);writeln('A művelet és eredmények');
    write('a*b=');writeln(a*b);
    write('a/b=');writeln(a/b);
    write('a+b=');writeln(a+b);
    write('a-b=');writeln(a-b);
end;
itt kezdődik a program végrehajtható része)
begin
    fajlec;
    torzs;
end.
```

13. ábra

```
program gvak2.2;
(ciklusok repeat until)
var a:byte;
ch:char;
procedure fajlec;
begin
    clrscr;gotoxy(3,3);
    writeln('Harmadik gyakorló programunk');
    write('');
end;
procedure torzs;
begin
    gotoxy(2,9);writeln('Az adatok');
    write('a=');readln(a);
    write('b=');readln(b);
    gotoxy(2,9);writeln('A művelet és eredmények');
    write('a*b=');writeln(a*b);
    write('a/b=');writeln(a/b);
    write('a+b=');writeln(a+b);
    write('a-b=');writeln(a-b);
end;
repeat
    fajlec;
    torzs;
until ch=#27;
clrscr;
writeln('Ugyanúgy, folytatásukon az a billentyű');
readln(ch);
until ch=#27;
clrscr;
writeln;writeln('Program vége');
end.
```

11. ábra

```
Harmadik gyakorló programunk
Az adatok:
a=1
b=3
A művelet és eredmények:
a*b= 4.0000000000E+00
a/b= 3.0000000000E+00
a+b= 3.0000000000E+00
a-b= 1.3333333333E-01
>
```

14. ábra

```
program gvak2.3;
(ciklusok while do, stringek)
var a:byte;
ch:char;
procedure fajlec;
begin
    clrscr;
    write('Kérem a nevét');
    readln(ch);
end;
procedure torzs;
begin
    clrscr;gotoxy(3,1);
    writeln('Harmadik gyakorló programunk');
    write('');
end;
procedure torzs;
begin
    gotoxy(2,9);writeln('Az adatok');
    write('a=');readln(a);
    write('b=');readln(b);
    gotoxy(2,9);writeln('A művelet és eredmények');
    write('a*b=');writeln(a*b);
    write('a/b=');writeln(a/b);
    write('a+b=');writeln(a+b);
    write('a-b=');writeln(a-b);
end;
while ch=#27 do
begin
    fajlec;
    torzs;
    writeln;
    writeln('Ugyanúgy, folytatásukon az a billentyű');
    readln(ch);
end;
clrscr;
writeln;writeln('Program vége');
end.
```




változó vagy kifejezés értéke **true**, a ciklus lefut, ha nem teljesül (az értéke **false**), a vezérlés a ciklusmag utáni utasításra kerül. Ebből következik, hogy:

- ha a ciklusmagot legalább egyszer végére akarjuk hajtani, a **while**-ban előírt feltételt igaz voltát még a ciklusutasítás előtt biztosítani kell;

- a cikusból való kilépés megvalósításához a ciklusmagban meg kell változtatni a **while**-ban előírt feltételt úgy, hogy értéke hamissá (**false**) váljon; ebben az esetben a vezérlés a következő vizsgálat után a ciklusmagot követő utasításra kerül.

Példánkban a ciklusmag legalább egyszeri lefutását úgy tettük lehetővé, hogy a **ch** változóznak **#26** értéket adtunk, a **while** utasításban pedig a ciklusba belépést a

```
while ch <> #27
feltétel teljesüléséértől tettük függővé. Mivel az adott esetben ez teljesül, a vezérlés a ciklusmagra kerül. A ciklus végén a
```

```
read (kbd,ch)
utasítással a ch változóznak új értéket adunk; ettől függően a ciklus újból végrehajtható, vagy a vezérlés a következő utasításnak adódik át. A programlistát a 14. ábrán láthatjuk.
```

Ciklusokat egyébként szervezhetünk az **if**, **then** és a **goto** utasításokkal is. A Pascalban a **goto** utasítás alkalmazását viszont célszerű kerülni, mert így éppen a nyelv egyik előnye, az áttekinthető programok készítésének lehetősége vesz el. Emiatt az **if**, **then** és a **goto** utasításokkal megvalósított ciklusok használatát nem javasoljuk.

Nagy Imre

XT/AT -TULAJDONOSOKNAK!

Print Screen

lemezre

Biztosan sokmindenkivel előfordult már, hogy egy nagyobb felhasználói programnál szeretne volna kinyomtatni a HELP képernyő tartalmát vagy a

program egy képernyőjét. A megoldás egyszerű: tegyünk a nyomtatóba papírt és nyomjuk meg a Print Screen billentyűt. És ... meglepődve látjuk,

Az eredeti nyomtatás

C:\				D:\			
Name	Size	Date	Time	Name	Size	Date	Time
BASIC	SUB-DIR	88-10-08	10:13	ARCHIV	SUB-DIR	88-09-26	15:18
DOS	SUB-DIR	88-09-26	13:41	MUNKA ASM	SUB-DIR	88-09-26	13:48
FASTBACK	SUB-DIR	88-10-20	11:09	MUNKA BAS	SUB-DIR	88-10-08	10:13
GUIDE	SUB-DIR	88-10-08	11:02	MUNKA FAS	SUB-DIR	80-01-03	22:03
MASH	SUB-DIR	88-09-26	13:43	MUNKA TXT	SUB-DIR	88-09-26	13:49
NORTON	SUB-DIR	88-10-08	11:01	PCPAINT	SUB-DIR	88-11-09	7:00
PASCAL	SUB-DIR	80-01-03	22:27	STREEMER	SUB-DIR	88-09-27	11:01
lbmbio com	16369	85-12-30	12:00	UTILITY	SUB-DIR	88-09-26	13:57
lbmdos com	28477	85-12-30	12:00				
ansi sys	1631	85-12-30	12:00				
autoexec bat	80	88-11-02	14:27				
command com	23791	85-12-30	12:00				

C:\>autoexec

C:\>echo off

IBM Personal Computer DOS Version 3.20
The Norton Commander, Copyright (C) 1986 by Peter Norton

C:\>

A javítás utáni kép

C:\				D:\			
Name	Size	Date	Time	Name	Size	Date	Time
BASIC	SUB-DIR	88-10-08	10:13	ARCHIV	SUB-DIR	88-09-26	15:18
DOS	SUB-DIR	88-09-26	13:41	MUNKA ASM	SUB-DIR	88-09-26	13:48
FASTBACK	SUB-DIR	88-10-20	11:09	MUNKA BAS	SUB-DIR	88-10-08	10:13
GUIDE	SUB-DIR	88-10-08	11:02	MUNKA FAS	SUB-DIR	80-01-03	22:03
MASH	SUB-DIR	88-09-26	13:43	MUNKA TXT	SUB-DIR	88-09-26	13:49
NORTON	SUB-DIR	88-10-08	11:01	PCPAINT	SUB-DIR	88-11-09	7:00
PASCAL	SUB-DIR	80-01-03	22:27	STREEMER	SUB-DIR	88-09-27	11:01
lbmbio com	16369	85-12-30	12:00	UTILITY	SUB-DIR	88-09-26	13:57
lbmdos com	28477	85-12-30	12:00				
ansi sys	1631	85-12-30	12:00				
autoexec bat	80	88-11-02	14:27				
command com	23791	85-12-30	12:00				

C:\>autoexec

C:\>echo off

IBM Personal Computer DOS Version 3.20
The Norton Commander, Copyright (C) 1986 by Peter Norton

C:\>

Minden kedden 17-től 20 óráig
HCC ENTERPRISE klub
a VSZM
Közösségi Házban
(Bp. XI., Fehérvári út 120.)
Klubvezető: Romvári Gábor
Telefon: 810-950/473

A Tudományos Szervezési és Informatikai Intézet

előzetes megbeszélés szerint díjmentes programbemutatót tart (vieleken is) az általa forgalmazott oktatóprogramokból.

Horváth Zsuzsa 865-011/2663 mellék
vagy 813-187

Budapest, Pf. 454, 1372



hogy nem az van a papíron, amit a képernyőn láttunk, hanem csak valami hasonló.

Ennek az az oka, hogy a nyomtatók többségének a képernyőn megjelenő karakterek egy része (ASCII 1-től 31-ig) nem nyomtatható karakter, hanem — mint például a 27 — vezérlőkódot jelent. Ez nagyon elronthatja, vagy esetleg — ha a grafikus váltókód is benne volt — olvashatatlanná is teheti a szöveget. Ezt a hibát kétféleképpen lehetne kijavítani:

— az ASCII 1—31 kódok helyett szókora karaktert küldünk ki a nyomtatóra,

— a nyomtatás helyett a lemezre mentjük a képernyő tartalmát, amit ezután bármilyen szövegszerkesztő-

vel kijavíthatunk és többször is nyomtathatunk.

Ennek az utóbbi módszernek előnye, hogy egy képet többször, különbözőképpen is módosíthatunk, saját programjainkban felhasználhatunk, illetve a programjaink leírásába beleszerkeszthetünk. Éppen ezért ezt a megoldást ajánlatos választani, hiszen az eredményt sokféleképpen felhasználhatjuk.

Felmerül a kérdés: milyen billentyűvel vezéreljük ezt az új „képernyőnyomtatást”? A legegyszerűbb — mivel az eredeti Print Screen funkciót váltja fel — a PrtSc billentyűvel, mert ezt a felhasználói programok nem használják más, különleges feladatokra.

A DOS 5-ös funkciója a Print Screen. Ennek a rutinnak az indítási címét írja át a *program* második része, amely arra is vigyáz, hogy a programot ne lehessen egynél többször installálni. Az első elindítás után a program betöltődik a memóriába, és a számítógép kikapcsolásáig vagy újabb rendszer betöltéséig használható úgynevezett rezidens programként.

A PrtSc billentyű megnyomásakor a vezérlés átadódik az IT_CIM címre, és elmenti a regiszterek tartalmát. Erre a főprogramba való visszatérés miatt van szükség. Ezután a pillanatnyilag nyitott alkönyvtárban létrehozza az első nem létező SCREEN.X fájl — ahol X.A..Z tartományban van. Így a már meglévő fájlokat nem írja felül. Ha már SCREEN.A-tól SCREEN.Z-ig az összes fájlnévet felhasználunk, akkor mentés nélkül visszakerül a vezérlés a főprogramba, éppen úgy, mint az eredeti Print Screennél, ha nincs nyomtató.

A lemezre nemcsak a képernyő tartalmát mentjük ki, hanem minden sor végén egy Return (ASCII 13) és egy LineFeed (ASCII 10) karaktert is.

A programot lefordítás után alakítsuk át COM kiterjesztésűre az EXE2BIN programmal, és csak a COM kiterjesztésű változatot próbáljuk meg lefuttatni.

Krizsák László

```

code segment
assume cs:code,ds:code,ss:code

dos equ 21h
kep equ 10h
it equ 5h
quit equ 20h
rquit equ 27h

org 100h
start: jmp start2
it_cim: cli

; regiszterek töltése
push ax
push ds
push cs
push dx
push bx
push cx
pop ds

; A pillanatnyilag nyitott alkönyvtárba
; nem szereplő első SCREEN.X file
; megkeresése
mov bl,"A"
wi: mov cx,0ffffh
mov [fcb+7],bl
lea dx,fcb
mov ah,40h
int dos
or al,al
jnz #2
inc bl
cmp bl,"Z"+1
jnz #1
jmp kilep

; A SCREEN.X létrehozása
#2: lea dx,fcb
mov cx,0
mov ah,3ch
int dos
mov bx,ax
push bx

; A képernyő adatainak kiolvasása
mov ah,15
int kep
mov cl,ah
mov ch,25
push cx

; Az eredeti kurzor pozíció töltése
mov ah,3
int kep
pop cx
push ds
xor dx,dx
ciklus: mov ah,2

```

```

int kep
mov ah,8
int kep
push dx

; Egy byte kimentése a lemezre
push cx
mov cx,1
mov [fcb+7],al
lea dx,fcb+7
mov ah,40h
int dos
pop cx
pop dx
inc dl
cmp cl,dl
jnz ciklus
xor di,dl
push ds

; Sorvég töltése
push cx
mov cx,2
lea dx,crlf
mov ah,40h
int dos
pop cx
pop dx
inc dx
cmp ch,dh
jnz ciklus

; A kurzor pozíció visszaállítása
pop dx
mov ah,2
int kep

; File lezárás
mov ah,5eh
pop bx
int dos

kilep: sti

; Regiszterek visszatöltése
pop bx
pop dx
pop cx
pop ds
pop ax
popf
iret

crlf db 0dh,0ah
fcb db "SCREEN.@",0
vege db 0
start2: cli

; PrintScreen IT (5) vektor átrása

```

```

mov ax,3500h+IT
int dos
mov ax,[word ptr it_cim]
cmp esi[bx],ax
jnz inic
lea dx,volt
mov ah,9
int dos
int quit

; Új IT vektor DS:DX-be
; Új IT vektor beállítása
inic: lea dx,it_cim
mov ax,2500h+IT
int dos

sti
lea dx,szoveg
mov ah,9
int dos
lea dx,vege

; Kilepés. A program a memóriában marad
rquit: int
volt db "A program már "
db "installálva volt!"
db 0dh,0ah,"$"
szoveg db "A Diskscreen program "
db "installálva.",0dh,0ah,"$"

code ends
end start

```




Polinomok szorzása és osztása

Műszaki feladatok megoldásakor gyakran előforduló matematikai művelet, hogy polinomokat kell egymással megszorozni vagy elosztani. (Az utóbbi a törtekifejezések egyszerűsítésénél is gyakran felmerül.) Több tagból álló polinomoknál ez meglehetősen munkaigényes feladat, amire ezért célszerű számítógépes programot készíteni.

Szorzás

Legyen a szorzandó az

$$f(x) = a_n x^n + a_{n-1} x^{n-1} + a_{n-2} x^{n-2} + \dots + a_1 x + a_0$$

a szorzó pedig a

$$g(x) = b_m x^m + b_{m-1} x^{m-1} + b_{m-2} x^{m-2} + \dots + b_1 x + b_0$$

alakú polinom. A két polinom szorzatát úgy képezzük, hogy minden tagot minden taggal megszorozunk:

$$s(x) = f(x) \cdot g(x) = a_0 b_0 + a_0 b_1 x + \dots + a_0 b_m x^m + a_1 b_0 x + a_1 b_1 x^2 + a_1 b_2 x^3 + \dots + a_1 b_m x^{m+1} + \dots + a_n b_m x^{m+n}$$

Könnyen belátható, hogy ily módon $m \cdot n$ számú szorzást kell elvégezni, majd az azonos hatványkitevőjű tagokat össze kell vonni. A sorozatosan ismétlődő szorzásokat és összeadásokat ciklusba szervezhetjük (1. ábra).

Osztás

A polinomok osztása már bonyolultabb művelet. Gondoljuk meg, mit csinálunk a számok osztásánál. Például:

$$\begin{array}{r} 693:29=23 \\ -58 \\ \hline 113 \\ -87 \\ \hline 26 \text{ (maradék)} \end{array}$$

Polinomok osztásánál voltaképpen ugyanígy járunk el. Legyen például a feladat ez:

$$f(x) = 3x^2 - 6x + 3$$

$$g(x) = 2x - 1$$

és képezni kell a

$$h(x) = \frac{f(x)}{g(x)}$$

hányadosot:

$$(4x^2 - 6x + 3):(2x - 1)$$

Az osztó első — legnagyobb kitevőjű — tagjával elosztjuk az osztandó legnagyobb kitevőjű tagját:

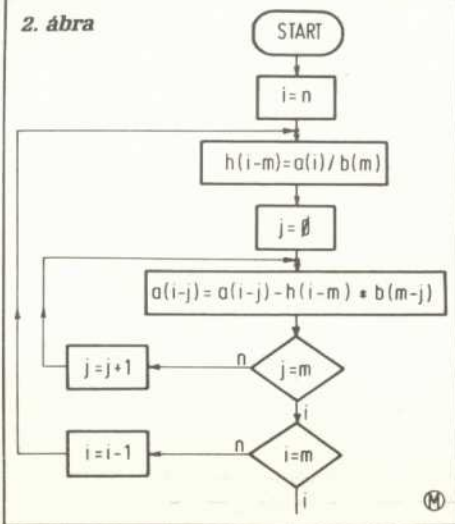
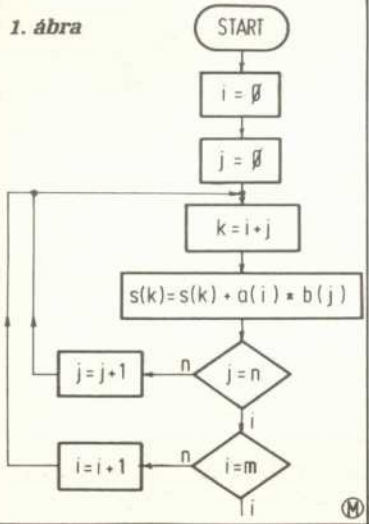
$$4x^2 : 2x = 2x$$

ez lesz a hányados polinom első tagja. Ezzel „visszaszorozunk”:

$$2x(2x - 1) = 4x^2 - 2x$$

Ezt leírjuk az osztandó megfelelő kitevőjű tagjai alá és kivonjuk azokból:

$$\begin{array}{r} (4x^2 - 6x + 3):(2x - 1) = 2x \\ -4x^2 + 2x \\ \hline 0 - 4x \end{array}$$



```

**** POLINOMOK SZORZASA ****
A szorzandó polinom folszása: 3
a(0)=5
a(1)=5
a(2)=-5
b(0)=-7

A szorzó polinom folszása: 2
A szorzó polinom együtthatói:
b(0)=-4
b(1)=-2

A szorzat polinom együtthatói:
s(0)=24
s(1)=-47
s(2)=20
s(3)=-35
s(4)=10
s(5)=-4
  
```

3. ábra

```

**** POLINOMOK OSZTASA ****
Az osztandó polinom folszása: 2
a(0)=6
a(1)=0
a(2)=0

Az osztó polinom folszása: 1
Az osztó polinom együtthatói:
b(0)=2
b(1)=-1

A hányados polinom együtthatói:
h(0)=2
h(1)=-2

A maradék polinom együtthatói:
a(0)=1
  
```



majd hozzáírjuk az osztandó következő tagját és újra elvégezzük az osztást az osztó legnagyobb kitevőjű tagjával:

$$(-4x+3):(2x-1) = -2$$

$$\frac{+4x-2}{0+1}$$

A hányados tehát $(2x-2)$ és van maradék is: 1, ez már nem osztható $(2x-1)$ -gyel.

Ezt a meglehetősen bonyolultnak tűnő eljárást egy séma szerint is elvégezhetjük. Ha ugyanazokat a jelöléseket alkalmazzuk, mint a szorzásnál, akkor a hányados polinom együtthatói így számíthatók ki, feltéve, hogy $n \geq m$ és $r = n - m$:

$$h_r = \frac{a_n}{b_m}$$

$$h_{r-1} = \frac{a_{n-1}}{b_m} - \frac{b_{m-1}h_r}{b_m}$$

$$h_{r-2} = \frac{a_{n-2}}{b_m} - \frac{b_{m-1}h_{r-1}}{b_m} - \frac{b_{m-2}h_r}{b_m}$$

$$\dots$$

$$h_0 = \frac{a_{n-m}}{b_m} - \frac{b_{m-1}h_1}{b_m} - \frac{b_{m-2}h_2}{b_m} - \dots - \frac{b_0h_{r-1}}{b_m}$$

Az osztás maradéka szintén polinom, amelynek együtthatói:

$$m_0 = a_0 - b_0h_0$$

$$m_1 = a_1 - bh_0 - b_0h_1$$

$$\dots$$

$$m_{m-1} = a_{m-1} - b_{m-1}h_0 - b_{m-2}h_1 - \dots - b_0h_{m-1}$$

Az osztás eredménye tehát ez:

$$h(x) = h_r x^r + h_{r-1} x^{r-1} + \dots + h_0 + \frac{m_{m-1} x^{m-1} + m_{m-2} x^{m-2} + \dots + m_0}{g(x)}$$

Az együtthatók eszerint a 2. ábrán bemutatott ciklussal számíthatók.

A program

Az elmondottak alapján felépített programot a lista mutatja. Az 1. ábrán látható ciklus a program 1210–1260-as soraiban, a 2. ábra szerinti ciklus a 250–300-as sorokban szerepel. Mivel a program Sinclair Spectrum gépre készült, amelynél a tömbök indexei nem vehetnek fel 0 értéket, ezeket 1-től kell indítani. Erre valamennyi tömbfeltöltésnél ügyelni kell.

A program a 20-as sorban kétféle művelet felsorolásával és választási lehetőséggel indul.

A számítási művelet mindkét esetben a kiinduló adatok beírásával kezdődik (140–230-as, illetve 1010–1100-as sorok). Az osztás csak akkor végezhető el, ha az osztó polinom fokszáma nem nagyobb, mint az osztandó ($n \geq m$). A program ezt a 210-es sorban figyeli, és ha ez a feltétel nem teljesül, figyelmeztetést ír ki.

Eredményként mindkét esetben csak az együtthatók jelennek meg a képernyőn, a betűkifejezéseket és kitevőjüket a felhasználónak kell hozzájuk rendelnie. Az együtthatók mellett álló indexszámok azonban egyáltalán utalnak a kitevőkre is. Példaként a 3. ábrán bemutatjuk a

$$(8x^3 - 5x^2 + 6x - 7)(3x^2 - 4x - 2) = -24x^5 - 47x^4 + 22x^3 - 35x^2 + 16x + 14$$

szorzás, a 4. ábrán pedig a

$$(4x^2 - 6x + 3):(2x - 1) = 2x - 2 + \frac{1}{2x - 1}$$

osztás eredményét, ahogy azt a számítógép kiírja.

Lázár Károly

```

1000 REM *****
1010 REM POLINOMOK SZORZASA *
1020 REM ES OSZTASA *
1030 REM Lázár Károly 1988 *
1040 REM *****
1050 CLS
1060 PRINT TAB 10;"LEHETŐSÉGEK:"
1070 PRINT
1080 PRINT TAB 4;"Polinomok oszt
1090 as.....1:";PRINT TAB 4
1100 "Polinomok szorzása.....2"
1110 PRINT #0;"Melyiket választj
1120 a?"
1130 PAUSE 0
1140 LET v$=INKEY$
1150 IF v$<"1" AND v$<"2" THEN
1160 GO TO 50
1170 IF v$="1" THEN GO TO 100
1180 IF v$="2" THEN GO TO 1000
1190 CLS
1200 PRINT TAB 1;BRIGHT 1;****
1210 POLINOMOK OSZTASA ****;BR
1220 IGH 0
1230 INPUT "Az osztandó polinom fok
1240 száma:";n
1250 PRINT;"Az osztandó
1260 polinom fokszáma:";m
1270 DIM a(n+1);DIM h(n+1);DIM
1280 a(n+1)
1290 PRINT;"Az osztandó
1300 polinom együtthatói:"
1310 FOR i=n+1 TO 1 STEP -1;INP
1320 UT a(i);PRINT "a("i-1;")=";a(i)
1330 NEXT i
1340 INPUT "Az osztó polinom fok
1350 száma:";m
1360 DIM b(m+1)
1370 PRINT;"Az osztó pol
1380 inom fokszáma:";m
1390 IF m>n THEN PRINT;"PRINT B

```

```

RIGHT 1;PAPER 7;INK 7;"Az os
210 ztó polinom fokszáma nem lehet n
220 egynél nagyobb, mint az osztandó!";BRI
230 GHT 0;PAPER 7;INK 0;GO TO 190
240 PRINT;"Az osztó pol
250 inom együtthatói:"
260 FOR i=m+1 TO 1 STEP -1;INP
270 UT b(i);PRINT "b("i-1;")=";b(i)
280 NEXT i
290 LET g=1;GO SUB 2000;CLS
300 LET i=-m
310 FOR i=n+1 TO m+1 STEP -1
320 LET h(i-a)=a(i)/b(m+1)
330 FOR j=0 TO m
340 LET s(i-j)=a(i-j)-h(i-a)*b(j
350 m+1-j)
360 NEXT j
370 PRINT;"Az hányados p
380 olinom együtthatói:";PRINT
390 FOR i=n+1 TO 1 STEP -1
400 PRINT "h("i-1;")=";h(i)
410 NEXT i
420 PRINT;"Az maradék po
430 linom együtthatói:";PRINT
440 INPUT "Az osztandó polinom fok
450 száma:";n
460 LET a(n+1)=a(i)
470 PRINT "a("i-1;")=";a(i)
480 GO TO 3000
1000 CLS;PRINT TAB 1;BRIGHT 1
1010 **** POLINOMOK SZORZASA ***
1020 BRIGHT 0;PRINT
1030 INPUT "A szorzandó polinom fok
1040 száma:";n
1050 PRINT;"A szorzandó polinom fok
1060 száma:";m
1070 DIM a(n+1)
1080 FOR i=n+1 TO 1 STEP -1;INP
1090 UT a(i);PRINT "a("i-1;")=";a(i)
1100 NEXT i

```

```

1050 INPUT "A szorzó polinom fok
1060 száma:";m
1070 PRINT;"A szorzó pol
1080 inom fokszáma:";m
1090 PRINT;"A szorzó pol
1100 inom együtthatói:"
1110 DIM b(m+1);DIM s(n+m+2)
1120 FOR i=m+1 TO 1 STEP -1;INP
1130 UT b(i);PRINT "b("i-1;")=";b(i)
1140 NEXT i
1150 LET g=2;GO SUB 2000;CLS
1160 PRINT;"A szorzat polinom eg
1170 yütthatói:";PRINT
1180 FOR j=1 TO m+1
1190 LET k=j+1
1200 LET s(k)=s(k)+a(i)*b(j)
1210 NEXT j
1220 NEXT i
1230 PRINT;"
1240 LET s(k)=s(k)+a(i)*b(j)
1250 NEXT j
1260 NEXT i
1270 FOR i=n+m+2 TO 2 STEP -1
1280 PRINT "s("i-2;")=";s(i)
1290 NEXT i
1300 GO TO 3000
2000 PRINT #1;"Helyesek az adato
2010 k? (i/n):";PAUSE 0
2020 LET v$=INKEY$
2030 IF v$<"1" AND v$<"n" THEN
2040 GO TO 2010
2050 IF v$="1" THEN RETURN
2060 IF v$="n" THEN PRINT;"PRIN
2070 T BRIGHT 1;"Írja be újra az adat
2080 okat!";BRIGHT 0;PAUSE 150;CLS
2090 GO TO (g=1)*100+(g=2)*1000
3000 PRINT #0;"Folytatjuk? (i/n):
3010 PAUSE 0
3020 LET v$=INKEY$
3030 IF v$<"1" AND v$<"n" THEN
3040 GO TO 3010
3050 IF v$="1" THEN GO TO 10
3060 IF v$="n" THEN CLS;"PRINT
3070 #0;6;"Köszönöm, befejeztük.";
3080 STOP

```


A PÉNZÜGYI SZÁMÍTÁSTECHNIKAI INTÉZET

felkínálja megvételre

a következő, használaton kívüli állóeszközeit:

*VT340 terminálok
VDT52102 terminálok
VDT52117 terminálok
QVT-100 Quadro terminálok
Tatung 6600A terminálok*

*VT16 mikrogép konfiguráció
Proper16 mikrogép konfiguráció*

DZM180 mátrixnyomtatók

*DCD300 beépíthető kazettás
egységek
DCU-2 ikerkazettás egységek*

*TPA-S rackfiókok, különféle
illesztő kártyák*

*Siemens 4581 (55MB) lemez-
egységek*



*A berendezések részben működőképeseek, részben alkatrészellátással
nem rendelkező régi berendezésekhez bontásra ajánljuk.*

Árak megegyezés szerint.

*Érdeklődni lehet: PSZTI TAF Hálózatfejlesztési osztály
Telefon: 888-983*

FELADATOK

– MEGOLDÁSOK

Sorozatunkat elsősorban középiskolásoknak szánjuk, de reméljük, hogy minden olvasónknak tanulási lehetőséget és szórakozást nyújt.

A feladatok a Nemes Tihámér országos számítástechnikai verseny színvonalának felelnek meg. Minden esetben olyat választunk, amely röviden, gyorsan megoldható, de megoldásához ötletre van szükség. A megoldást mindig a következő számban közöljük.

Mivel változatosságra törekszünk, különböző programozási nyelveket használunk. Az is előfordul majd, hogy egy feladatra több programnyelven is közlünk megoldást, ezzel is elősegítve az ismeretszerzést.

A szerkesztőség várja az olvasók, a versenyzők leveleit. A legötletesebb program beküldését könyvváltvánnyal jutalmazzuk. Ne feledjenek azonban a programhoz leírást is mellékelni!

9. feladat: Rekurzív rajz

Írjon programot, amely beolvas egy számot és kirajzolja az ennek megfelelő fokszámú rekurzív görbét. Az ábrákon néhány példa látható.

Megoldás

Hasonlítsuk össze a harmad- és negyedfokú ábrát! A negyedfokú ábra a közepén látható elemből és az abból kinyúló négy „ágából” áll. Ezek az „ágak” nagyon hasonlóak a harmadfokú ábrához. Nevezzük ezeket az „ágakat” ezért harmadfokú zárt vonaldaraboknak. Az 1. ábrán vastagított vonallal egy ilyen emeltünk ki.

Egy „ágot”, más szóval egy harmadfokú zárt vonaldara-

bot még három részre bonthatunk: egy oda- és egy visszafelé vezető vonalszakaszra, továbbá egy lezáró átlós vonalkára. Nevezzük ezeket a részeket harmadfokú nyitott vonaldaraboknak. A 2. ábrán ez látható kiemelve.

Ha most megfigyeljük a harmadfokú nyitott vonaldarabot, az három részre bontható: egy bal oldali és egy jobb oldali másodfokú nyitott vonaldarabra, valamint középen egy másodfokú zárt vonaldarabra. A részeket a 3. ábra mutatja.

Hasonló szabály mondható ki a másodfokú nyitott görbére is: az két elsőfokú nyitott és egy elsőfokú zárt vonaldarabból áll.

Az ábrákon megfigyelhető szabályok felismerése után a feladat megoldása már igen egyszerű, feltéve, hogy olyan programozási nyelvet választunk, amelyek a rekurziót támogatja. A program BASIC-ben vagy Assemblerben való megírását így szinte azonnal elvethetjük.

A feladat megoldására a legalkalmasabb talán a Pascal nyelv, de a ZX—Spectrum Beta BASIC-ben vagy C64-en Simon's BASIC-ben is igen szép megoldás írható.

A megoldást a két előbbi nyelven mutatom be. Nézzük először a Turbo Pascal 4.0 nyelvű, IBM PC-n futtatható

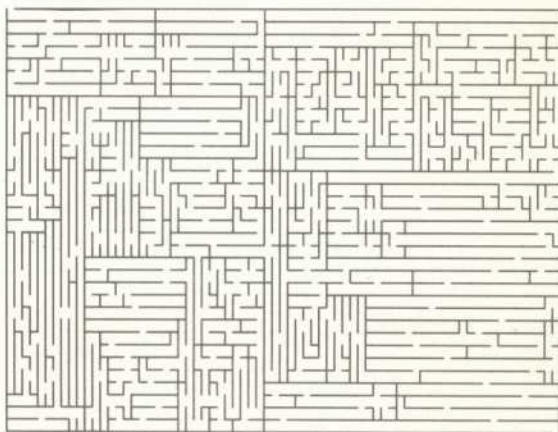
programot. A program két fő eljárásból áll. A RajzNyitott vonaldarab rajzolását végzi, a RajzZárt pedig a zártét. A paraméterezése és felépítése nagyon hasonló. Mindkét rutin megkapja, hogy melyik irányba (Irány) rajzoljon, és azt, hogy milyen fokszámú vonaldarabot. Mindkét rutin Irány-tól függően szétágazik és a megfelelő irányban rajzol.

A működés a fent említett szabályok alapján könnyen érthető.

Egy zárt vonaldarab két azonos fokszámú nyitott darabból áll és egy átlós vonalkából. Így a zárt vonaldarabot

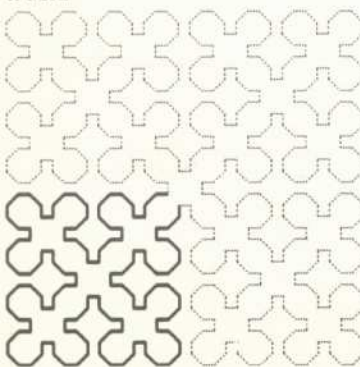
rajzoló szubrutin is két nyitott darabot rajzol (RajzNyitott segítségével) és ezek közé egy átlós vonalkát. Ha a fokszám nulla, akkor természetesen a nyitott vonaldarabok rajzolása elmarad.

Hasonlóan egy nyitott vonaldarab két eggyel alacsonyabb fokszámú nyitott darabból, egy eggyel alacsonyabb fokszámú zárt darabból és két összekötő vonalkából áll. Ennek megfelelően RajzNyitott a két alacsonyabb fokszámú nyitott szakasz rajzolásához önmagát, a zárt szakaszhoz RajzZárt-at hívja meg eggyel alacsonyabb fokszámmal. A tulajdonképpen

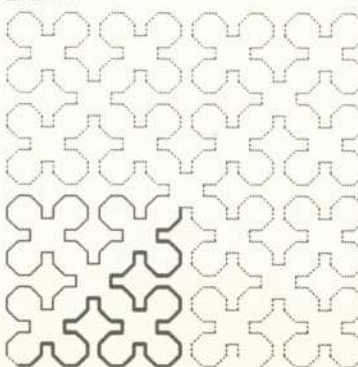


Egy labirintus

1. ábra



2. ábra



3. ábra



főprogramnak már kevés a to-
lga marad. A paraméterek be-
olvasása után a grafikus
üzemmód beállítása követke-
zik. Az ábra rajzolása egy át-
lós vonalka meghúzása után
teljes egészében a RajzZárt
rutinra marad. A főprogram
végezetül mintegy ellenőrzés-
képpen kiszűri a görbét. Ér-
demes felfigyelni arra is,
hogy a kép torzítását hogyan
küszöböli ki a Torzit konstans
és az Mx, My, Nx, Ny válto-
zó. Ez a torzítás szükséges
lehet akkor is, ha a képet
nyomtatni akarjuk.

A Beta BASIC-ban megírt
mások program szinte sorról
sorra megfeleltethető a most
vizsgált Pascal programnak.
A két szubrutint itt is megta-
láljuk (rajz__ny és rajz__z). Az
egyedüli különbség az irá-
nyok jelölésében van. Míg
Pascalban a felsorolt típus
szemléletes jelölést tett lehe-
tővé, itt az 1-4 számok jelölik
a megfelelő irányokat, s ez
kissé rontja a program olvás-
hatóságát.

A Beta BASIC lista

```

10 REM -----
    REKURZÍV RAJZOLÓ
    PROGRAM
    1000.09.10.
    Pinter Gábor
    -----
20 GOTO 8
30 GOTO 1
40 LET S=5
50 REM
60 PLOT 200-1000-(2*magyasag-1)
70 MAG-1) * (Mx-1) / (2*Mx-1)
80 RAJZ_1, magyasag
1000 REM
    Zárt szakszaj rajzolása
1010 DEF PROC rajz_x (irany,f
1020 GOTO ON irany:1040,1060,
1040 ORAY -M,N
1050 ORAY -M,N
1060 ORAY -N,N
1070 ORAY -M,130
1080 GOTO 1100
1090 GOTO 1100
1100 GOTO 1130
1110 GOTO 1130
1120 GOTO 1130
1130 RAJZ_NV 2,f
1140 ORAY -M,N
1150 GOTO 1130
1160 RAJZ_NV 2,f
1170 ORAY -M,N
1180 END PROC
1200 REM
    Nyitott szakszaj rajzolása
1210 DEF PROC rajz_ny (irany,f
1220 GOTO ON irany:1040,1060,
1040 RAJZ_NV 1,f-1
1050 RAJZ_NV 2,f-1
1060 RAJZ_NV 2,f-1
1070 GOTO 1080
1080 ORAY -M,N
1090 RAJZ_NV 2,f-1
1100 GOTO 1080
1110 RAJZ_NV 2,f-1
1120 GOTO 1080
1130 RAJZ_NV 4,f-1
1140 ORAY -M,N
1150 GOTO 1080
1160 RAJZ_NV 4,f-1
1170 RAJZ_NV 4,f-1
1180 END PROC
1200 END
    
```

A most következő és a to-
vábbi néhány feladat a labi-
rintus készítésével és bejárá-
sával kapcsolatos.

10. feladat: Labirintusrajzolás

Írjon programot, amely vé-
letlen labirintust rajzol. A labi-
rintusnak a következő tulaj-
donságai legyenek:
— egy be- és egy kijárata
van,
— a bejáratról a kijáratig
egy és csak egy út vezet,
— a bejáratról a labirintus
bármely pontjára el lehet jut-
ni.

Pinter Gábor

A Turbo Pascal 4. 0 lista

```

-----
    Rekurzív rajzoló program
    -----
    1988. 09. 23.
    -----
    Pinter Gábor
    -----
uses
    Graph;
type
    Iranytípus =
        (le_balra, fel_balra,
         fel_jobbra, le_jobbra);
const
    torzit = 1.43;
var
    Gd, Gm           : Integer;
    Mx, My, Nx, Ny, X, Y : Integer;
    Nagysag         : Integer;
    Ssin            : Integer;
    Ch              : Char;

procedure RajzNyitott
    ( Irany : Iranytípus;
      Fokszam : Integer ); forward;

{ Zárt görbe rajzolása }
procedure RajzZart
    ( Irany : Iranytípus;
      Fokszam : Integer );

begin
    if Fokszam=0 then
        case Irany of
            le_balra : LineRel(Mx,-My);
            fel_balra : LineRel(-Mx,-My);
            fel_jobbra : LineRel(-Mx,My);
            le_jobbra : LineRel(Mx,My)
        end
    else
        case Irany of
            le_balra :
                begin
                    RajzNyitott(le_balra,Fokszam);
                    LineRel(Mx,-My);
                    RajzNyitott(fel_jobbra,Fokszam)
                end;
            fel_balra :
                begin
                    RajzNyitott(fel_balra,Fokszam);
                    LineRel(-Mx,-My);
                    RajzNyitott(le_jobbra,Fokszam)
                end;
        end
    end
end
    
```

```

end (case)
end (RajzZart);

{ Nyitott görbe rajzolása }
procedure RajzNyitott
    ( Irany : Iranytípus;
      Fokszam : Integer );
begin
    if Fokszam(>0) then
        case Irany of
            le_balra :
                begin
                    RajzNyitott(le_balra,Fokszam-1);
                    LineRel(-Mx,0);
                    RajzZart(fel_balra,Fokszam-1);
                    LineRel(0,-My);
                    RajzNyitott(le_balra,Fokszam-1)
                end;
            fel_balra :
                begin
                    RajzNyitott(fel_balra,Fokszam-1);
                    LineRel(0,My);
                    RajzZart(fel_jobbra,Fokszam-1);
                    LineRel(-Mx,0);
                    RajzNyitott(fel_balra,Fokszam-1)
                end;
            fel_jobbra :
                begin
                    RajzNyitott(fel_jobbra,Fokszam-1);
                    LineRel(Mx,0);
                    RajzZart(le_balra,Fokszam-1);
                    LineRel(0,-My);
                    RajzNyitott(le_jobbra,Fokszam-1)
                end
        end (case)
    end (RajzNyitott);

begin
    repeat
        Write('Hányad foku legyen? ');
        Readln(Nagysag);
        Write('Visszaintes vonal hossza? ');
        Readln(My); Mx:=trunc(torzit*My);
        Write('Atlos vonal hossza? ');
        Readln(My); Nx:=trunc(torzit*My);
        Write('Ssin? (1-16) ');
        Readln(Ssin);

        Gd := Detect;
        InitGraph(Gd, Gm, '');
        if GraphResult (<) grOk then
            Halt(1);

        SetColor(Ssin);
        Y:=GetMaxY;
        X:=(GetMaxX+GetMaxY) div 2;
        MoveTo(X,Y-My);
        LineRel(-X,My);
        RajzZart(le_balra,Nagysag);
        SetFillStyle(SolidFill,Ssin);
        FloodFill(X-Mx,Y-My,Ssin);
        Read(CH);
        CloseGraph
    until (Ch<>chr(13))
end.
    
```

Közületek, figyellem!
 Mikroszámítógépet
 akarnak vásárolni?
 Tájékoztódnak a
 naprakész piaci helyzetről!
 Díjtalan ismertető!
MESZ Számítástechnika
 1368 Budapest, Pf.: 193

Kevés számítástechnikai cég mondhatja el magáról, hogy több mint harminc éve alakult. Az ilyen úttörők közé tartozó Élelmiszeripari Ügyvitelszervezési és Gépi Adatfeldolgozó Vállalat (ÉLGAV) alapításakor még csak a szűk értelemben vett ágazati feladatok megoldásában vett részt. Tevékenysége azonban a kor követelményeinek megfelelően alakult. Az évforduló alkalmából tekintünk át röviden az ifjúsági és humanitárius jellegű eredményeit.

Az ÉLGAV neve az úttörők körében ismerősen cseng: népszerű volt a Pajtsz újságban megjelent programozási pályázatuk. Az oktatással kapcsolatos ötleteket az ÉLGAV szakembereinek konzultációja mellett professzionális színvonalon, egységes megjelenési formában valósították meg, Commodore 64-es gépeken:

— az órarendkészítő program általános és középiskolákban egyaránt alkalmazható;

— az áramkörszerkesztő, -számító és -rajzoló program Blinszky László és Kétszery Zsolt munkája;

— a fizika-kémiai oktatóprogram első változatát két soproni általános iskola számítástechnikai szakköre Takács Sándor vezető tanár irányításával készítette;

— Kuti Ferenc Szentesről egy elektromosság-tant oktató programot írt.

A számítástechnikai munkáknál a vakok és a gyengénlátók foglalkoztatását az a tény akadályozza, hogy a gép által kiírt üzeneteket, listákat nem tudják elolvasni. E probléma megoldására Arató András és Vaspöri Teréz kifejlesztette a Homelab-4 olyan változatát, amely a képernyőre kiírt szöveget, szöveket egyúttal ki is mondja. Az így kifejlesztett, Brailab-nak nevezett gépnek ezzel a tulajdonságával, valamint az alacsony beszerzési árával látszólag a feladat meg is oldódott, hiszen készen volt egy olyan számítástechnikai eszköz, amit ezután csak használni kell. További akadályt jelentett azonban az, hogy ez a típus eredetileg csak házi, illetve oktatási gépek készült. Ekkor siettek segítségül az ÉLGAV szakemberei. Szoftver- és hardverfejlesztésük lehetővé tette, hogy mindazt, ami egy IBM PC-vel kompatibilis gép képernyőjén megjelenik, a Brailab hangosan ki is mondja. Ezáltal egy alig tízezer forintba kerülő Brailab csatlakoztatásával megoldható a vakok számára is az IBM PC-vel kompatibilis gépek kezelése: szoftvert fejleszhetnek rajta, vagy szövegszerkesztővel, adatbáziskezelővel felhasználói munkát is végezhetnek vele.

A nem beszélő, súlyosan mozgássérült gyerekek megkülönböztető- és felfogóképességének vizsgálatát, valamint fejlesztését szolgálja az a mozaik-összerakó program, amelyet a Gyógypedagógiai Pszichológiai Intézzel együttműködve fejlesztettek ki. Ennek futásakor a Commodore 64 képernyőjén hat vagy kilenc ábra jelenik meg. A fölöttük lévő minta vagy utasítás alapján ki kell választani azt, amelyik a mintával összefüggésbe hozható. A választás az erre a célra kifejlesztett — a mozgássérült gyermekek többsége által is kezelhető — nagyméretű vezérlőpulttal lehetséges. A helyes vagy helytelen válaszokat különböző hangjelzések kísérik. Ha a megoldás jól sikerült, akkor jellegzetes zene szólal meg és új feladat látható, de az F1 gomb megnyomásával akár azonnal megkerdezhető a helyes megoldás. Ott feladat van, az utolsóknak a végrehajtása után előlről kezdődik a feladatok megjelenítése. A hagyományos papír-ceruza tesztknél — a mozgásra való folyamatos koncentráció miatt —

Többszörösen ad, aki ma ad



IBM PC és a képernyőjén lévő szöveget „felolvasó” Homelab-4 — Brailab

a gyerek kimerül, mire a feladat tartalmi részéhez ér. A számítógép alkalmazásával viszont magára a feladatra összpontosíthat, s a reagálása igen egyszerű cselekvésre, a nagyméretű vezérlőpult megfelelő billentyűjének megnyomására szűkül. A program további jelentősége, hogy nem drága, speciális gépre, hanem házi számítógépre készült. A gyakorlatban tehát a gyerek az intézetben megtanulja, otthon pedig begyakorolja a használatát. Ezáltal a program a készségfejlesztési rendeltetését nemcsak az intézetben, hanem otthon, közvetlen családi környezetben is betöltheti.

Ezzel az áttekinthetssel még korántsem merítettük ki az ÉLGAV humanitárius tevékenységeinek ismertetését. Folytathatnánk a sort a Bárczi Gusztáv Gyógypedagógiai Tanárképző Főiskola szponzorálásával, térítésmentes szoftverátadásával stb. E tiszteletre méltó tevékenységi kör különös jelentőséget kap napjainkban, az egyre nehezebb gazdasági körülmények közepette. Kétszer ad, aki gyorsan ad, tartja a közmondás. Ezt bátran módosíthatjuk arra, hogy: többszörösen ad, aki ma ad. A szociális gondoskodás egyre soványodó pénzügyi lehetőségei mellett példamutató az ÉLGAV helytállása.

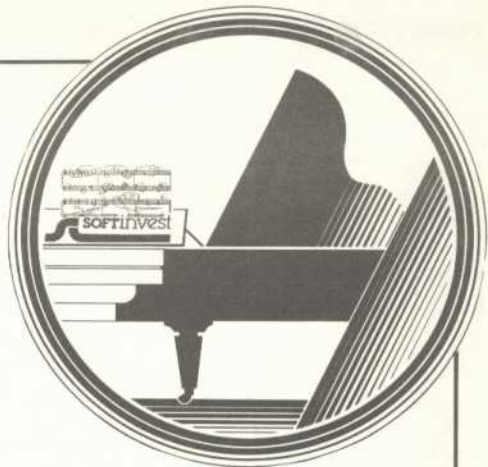
Brockzó Péter

Mozaiktesztelő mozgássérülteknek Az előtérben az e célra fejlesztett speciális vezérlőpult



Diagnózis?

PcVÍRUS



```
C:\>dir o d
. Volume n o
. Direct lyaI
ACAD-NCoiBEX
AAAA-NC1rXEX
AAAA-NC25XFi
Virus!!!
Ne_másoljon!
          1 -
          u 4 12
          h:3C i9 - 988
          hu 12-29-1988 VIR
          T8:36:30.611-0 -80 2:01
          iDIR>L569331-01-80 12:01a
          Cap <DI32244.SI1-01-80 12:04a
          r<(s317164981-01-80 12:05a
          C:\> fehu)12-29-16881-80US12:06a
          C:\>captu 1T8:34:23.019881bytes1freea
```

Terápia?

PcHIGI

Bővebb tájékoztatóval

és programbemutatóval rendelkezésére áll a



SZOFTVERKERESKEDELMI
ÉS FEJLESZTÉSI BETÉTI TÁRSULÁS
1591 Budapest, Pf. 218 Tel.119 - 067, 328 - 769

Egyre több olyan program készül, melynek feladata, hogy programokat és adatállományokat tömörítsen, azaz rövidebbé tegyen. Könnyítendő a dolgot, ilyen programok írásához kívánunk segítséget nyújtani.

A statisztikai tömörítés lényege, hogy bizonyos bajtók az átlagosnál gyakoribban, mások pedig az átlagosnál ritkábban fordulnak elő a fájlban. Ha a gyakori bajtókot 8-nál kevesebb biten ábrázoljuk azon az áron, hogy a ritkábbak 8-nál több bitet foglalnak el, akkor a gyakori nyereség és a ritka veszteség egyenlege nyereség lesz, tehát a fájl rövidül. Ezgakt matematikailag módon bizonyítható, hogy így minden fájlhoz szerkeszthető optimális kód, amellyel az adott fájl mérete az elméleti minimumra csökkenthető. Ilyen kód előállításáról szól Demetrovics—Denev—Pavlov: A számítástudomány matematikái alapjai című könyv 6. fejezetének 4. pontja: Az optimális kód Huffman-féle konstrukciója. Sajnos aki ezt kitalálta, az valószínűleg nem programozó volt, hanem matematikus, ugyanis a Huffman-elv szerint tömörített fájl szétpakolása lassú és bonyolult. A gyakorlatban érdemes engedni az optimumból némi gyorsaságnövelés fejében. Ezért most egy ilyen kompromisszumos tömörítő algoritmust vázolunk fel.

Minden egyes bajt kódja két részből áll: a bevezető bitekből és az adatbitekből. Legyen a bevezető bitek száma 2. Az adatbitek száma a bevezető bitek értékétől függ:

00 — 2 adatbit (összesen 4 bit)
01 — 4 adatbit (összesen 6 bit)
10 — 6 adatbit (összesen 8 bit)
11 — 8 adatbit (összesen 10 bit)

igy a négy leggyakoribb bajtot 2+2 azaz négy biten, a következőt 16 bajtot 2+4, azaz 6 biten, 2 bit nyereséggel; a következőt 64 bajtot 2+6, azaz 8 biten, nyereséggel; a maradék bajtokat 2+8, azaz 10 biten, 2 bit veszteséggel tudjuk ábrázolni. (Ezt a kódrendszert jelölhetjük így: 2+2, 2+4, 2+6, 2+8 vagy így: 2+(2,4,6,8).)

Látható, hogy 20 bajtot nyereséggel, 172 bajtot veszteséggel tudunk tárolni. Ezért a módszer akkor hatásos, ha ez a 172-féle bajt a fájlban maximum a felét teszi ki. Más esetben hatásosabb lehet a 2+(4,5,6,8) rendszer: 48 bajt nyereséggel, 144 bajt veszteséggel, vagy esetleg így harmadik. Sőt, a bevezető bitek számát is megváltoztathatjuk, ugyanis előállhat például a 4+(3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3,3) rendszer: 120-féle bajt nyereséggel, 136 bajt veszteséggel stb.

Mint látható, a lehetőségek száma szinte végtelen. Egész pontosan az 1+...alakú rendszerek száma 33, a 2+... alakúké 275, a 3+... alakúké 2765, a 4+... alakúké pedig 12 551, összesen 15 534. A közülük való választást a tömörítőnek kell elvégeznie.

Az adatbitek azt mutatják meg, hogy az adott bevezető bitekhez tartozó bajtók közül melyikről van szó.

AZ ALGORITMUS VÁZA

a) Statisztika. Meg kell számolni, hogy az egyes bajtók hányszor szerepelnek a fájlban, és ennek alapján sorba kell őket rendezni: a leggyakoribb bajt az első, a legritkább az utolsó, ami a gyakorisági sorrend.

b) Az optimális tömörítési rendszer kiválasztása. Mivel az egyes bajtók száma és a hozzájuk tartozó kód hossza is ismert, a tömörített fájl hossza kiszámítható anélkül, hogy a tényleges tömörítést el kellene végezni. Ekképpen tehát meg lehet keresni a legkisebb hosszút és a hozzá tartozó rendszert.

c) Tömörítés. Az eredeti állományt bajtonként olvassuk, megállapítjuk az egyes bajtókhoz tartozó kódot és azt bitenként kiírjuk.

A szétpakolónak természetesen ismernie kell a tömörítési rendszert és a bajtók gyakorisági sorrendjét. Bemutatunk egy konkrét példát. Legyen a választott rendszer a 2+(1,2,3,8), a gyakorisági táblázat: \$00, \$A2, \$FF, \$55, \$C2, \$BB, \$42, \$90, \$12, \$F3, \$39, \$A9, \$A0, \$0F, \$69, \$0B, \$3F stb. Ekkor az egyes bajtók kódjai:

Bajt	Bevezető bitek	Adat-bitek	Együtt
\$00 —	00	0	000
\$A2 —	00	1	001
\$FF —	01	000	0100
\$55 —	01	01	0101
\$C2 —	01	10	0110
\$BB —	01	11	0111
\$42 —	10	000	10000
\$90 —	10	001	10001
\$12 —	10	010	10010
\$F3 —	10	011	10011
\$39 —	10	100	10100
\$A9 —	10	101	10101
\$A0 —	10	110	10110
\$0F —	10	111	10111
\$69 —	11	00000000	1100000000
\$0B —	11	00000001	1100000001
\$3F —	11	00000010	1100000010
stb.			

A sorozatban leírt három tömörítési elv közül a bajtos a legjobb hatásfokú, utána következik a sorozatos, végül a statisztikai a leggyengébb. Egymás után is alkalmazhatjuk őket, ha a sorrendet nem változtatjuk meg. A statisztikai módszer annyira ösztönöz a biteket, hogy utána nemigen lehet már tovább tömöríteni a fájlt. A leghatékonyabb tömörítő programok ezeknek az algoritmusoknak valamilyen speciális keverékét használják.

Deierl András — Molnár Imre

MÉG AZT IS ÉRI BALESET,

AKI AKARJA

Sorscsapás (ellen)

- használati utasítás



Gépünket lehetőleg olyan helyiségbe tegyük, ahol szőnyegpadló is fellelhető. Öltözzünk műszálas szerelésbe, s minden, a szobában lecsoszogott kör után érintsük meg gépünk fémborítását.

Étekezzünk mindig a billentyűzet fölé, s a gombok közötti minél jobb helykihasználás érdekében felváltva fogyasztunk vajas-méz és szálmás zsendvicseket. S hogy a cementálódás tökéletes legyen, üdítünk, kávékn, tejnünk, sörünk töredékét lötytyintsük rá a már elhelyezett matériára.

A gépen néhány helyen található olyan kis nyílások, ahol belül aprócska propeller pereg. Ha ezekre a nyílásokra ragasztószalagot erősítünk, mókás hangokat hallhatunk.

Ha van lemezegységünk, az nagyon tudja értékelni, ha borítólemezen diktáljuk neki az ütemet.

Említsre méltó ötlet az is, hogy másnákat olyan elosztóra csatlakoztatjuk, ahová csatlakoztatjuk például háromfázisú avagy szénkefés, nagy teljesítményű szivattyút, automata mosógépet stb.

Botkormányunkat érdemes szilárd alapokra — beton, acél, gneis — rögzíteni, s elhúzódo szkander párvialdalban azt sem árt felfedezni, hogy a láb mindig kéznél van — a jobb erő kifejtés érdekében.

Ha esetleg környezetünkben véletlenül találunk egy magnóapparátot, először győződjünk meg arról, hogy nem csak átverésről van szó. Ezt úgy tehetjük meg, hogy alaposan beolajozott, bezsírozott s koszos ujjunkkal kitapogatjuk a lejátszófejet, majd a PLAY gomb lenyomásával egy időben sűrű változtatások közepette előre és hátra pörgetjük a kazettát. Ilyenkor többnyire líraian nyávog-hangot hallunk.

Ne feledkezzünk meg adathordozóinkról sem. Tároljuk őket mágneses környezetben, például tévé előtt, gesztótrafón, patkómágnesen. Minőségükről tapintással győződjünk meg. A floppykat érdemes szobában tárolni, de kisebb helyet foglalnak el, ha félbe hajtjuk őket.

Vámos Sándor



A FÖLÖSLEGES



Az eddigiekben már megismerkedtünk két stílussal; most is ezt tesszük, de egy kicsit más szempontból.

Nézzük, mi a teendője egy programozónak, ha kap egy feladatot. Természetesen ez a feladat a mi eddig használt példaprogramokunk!

Feladatleírás:

— olvassunk egy adat(work-)szalagot fájlvégjéig (TM tape mark-ig), ahol a rekordok 80 bájtosan blokkoltak. Ha a 80 bájtot 9–10. bájtra tartalmazza az "EZ" karakterláncot, akkor nyomtassuk ki az egész blokk tartalmát. Egyébként olvassuk tovább az állományt.

A feladat láttán programozónk elkezd gondolkodni, hogy milyen módon és milyen nyelven oldja azt meg. Utóbbin általában az adott feladattól függően határozza meg. Tehát mindig — az ismereteiben szereplő nyelvek közül — azt a nyelvet választja, amely éppen a leghatékonyabban használható.

Mindzekek után két feladatleírtet készít, ami kis programoknál fejben is összeállítandó, de persze célszerűbb mégis leírni. Az egyik a felhasználó szemszögéből, a másik a programozó szemzőgéből elemzi a megoldást.

A felhasználónak legfontosabb a könnyű kezelhetőség, a menütechnika, a lehetőség szerinti grafikai megoldások, a javítási lehetőségek, a hibakiszűrések (mind hardver, mind manware szempontjából), az izléses kivétel és a jó dokumentáció.

A programozó a strukturált felépítést, a többször használható utilityket (rutinokat), a rendszer sok részre bontását, a sebességet, a bemenet/kimenet kezelését és a hibák védelmét tartja leginkább szem előtt.

Természetesen az íméntiek mindegyike nagyon fontos egy program, programrendszer elkészítésénél, mert később az esetleges javításnál, módosításnál ezeknek a segítségével megelőlegezhetjük a könnyebb munkát.

Nézzük meg részletesen a fő szempontokat.

Könnyen kezelhetőség. Nem tudhatjuk, hogy ki fog dolgozni az általunk elkészített rendszeren. Lehet, hogy egy titkárnő, egy adminisztrátor stb. Tehát nem biztos, hogy ért a számítógéphez, s nekünk persze úgy kell megoldani a feladatot, hogy ők is tudják kezelni a rendszert. Ez pedig mindig lényeges!

Menütechnika. Az előbbiekből adódik, hogy célszerű a felhasználónak konkrét választási lehetőségeket adni. Ebből következik, hogy nem ajánlatos az ún. COMMAND parancsok bekérése, mert lassítja a feldolgozást, és rengeteg hibalehetőséget tartalmaz.

Lehetség szerinti grafika. Program barátságossá, látványossá teszi rendszerünket, ha a

statisztikákat, eredményeket némszakmokkal, hanem grafikonokkal is kiegészítjük. Ez nem olyan nagy munka, de az eredményt minden felhasználó értékeli.

Javítási lehetőség. Természetesen mindenhol előfordulhatnak hibák az adatfeldolgozás folyamán — általában az adatbevitelnél —, ezért célszerű az egyes adatszoportok után rákérdezni, hogy akar-e javítani a felhasználó.

A hibák kiszűrése. Egy jó rendszernek figyelnie kell az előforduló hibákat is. Itt gondolkodni például olyanokra, hogy a felhasználó elfelejtette a nyomtatót bekapcsolni, s a mi kis programocskánk éppen nyomtatni akar. Ha nem figyelünk az ilyen hibákat, akkor megállna az operációs rendszer egy nagyon szép, de igen kellemetlen hibázzennel, ami egy programozó számára igencsak ciki!

Izléses kivétel. Fontos, hogy a programrendszerünk a jó, hatékony működésen kívül még szép és izléses is legyen, mert ez megnyerheti a felhasználókat tetszését. Itt arra a csupán emberi szempontokra is gondolok, hogy netán még további megröndeléseket is kaphatunk, ha minden jól megy.

Dokumentáció. A felhasználó szinte ezt mondhatja az adott rendszer *hibájának*, hiszen ha bármi problémája van, ezt fellelőpóza, s támaszra lel benne. De ha ilyen nincs, akkor állandóan telefonálhat a programozónak, és elmegye a bizalma tőle. Célszerű a dokumentációnak az egyes menüket is tartalmaznia, bőséges magyarázatokkal.

Ha ezeket a tanácsokat megfogadjuk, s még ki is egészítjük, akkor biztos, hogy a felhasználóval szembeni kellemetlenségeinket elkerülhetjük!

A programozó szempontjai:

Strukturált felépítés. Programjainkat, amelyek a rendszert építik fel, tagoltan, blokkokban célszerű elkészíteni, mert így könnyebben áttekinthető és blokkonként kipróbálható.

Többször használható utilityk. Az olyan műveleteket, amelyek sokszor fordulnak elő a rendszer életében, célszerű utility (segédprogram, rutin) formájában megírni.

A rendszer felbontása. Ez lényegében meggyezik a strukturáltsággal, de fő erénye, hogy a felhasználónak be tudjuk mutatni a készülő rendszert.

Sebesség. Törekeznünk kell a viszonylag nagy sebességre, hiszen lehet, hogy több ezer adatot kell feldolgozunk.

A bemenet/kimenet kezelés. Ez mindig kevény része a rendszernek. Itt kell nagyon biztonságosan, kevés helyen, háttéráron megőrizni az adatot.

Hibák védelme. A saját biztonságunk érdekében célszerű belső hibáruhinokat elhelyezni, mivel ezekkel növelhetjük a rendszerünk biztonságát. Ilyen lehet például egy olyan rutin, amely minden B/K tevékenység esetén figyeli a háttéráron a szabad hely mennyiségét, és ha már nincs, erről értesíti a felhasználót.

A továbbiakban majd ezen elveket szert — hogy mi a fölösleges és mi nem az — mutatom be a példaprogramokat.

Bartos Gyula

```

1 REM ##### SOFTWARE C-16 #####
2 REM ##
3 REM ## SZINVALTO
4 REM ##
5 REM ## IRTA
6 REM ##
7 REM ## MUTSCHLER PETER
8 REM ##
9 REM ## INDUL SYS 1877
10 REM ## LEALL SYS 1857
11 REM ##
12 REM ## F1 - KERETSZIN
13 REM ## F2 - HATTERSZIN
14 REM ## F3 - CURSOLSZIN
15 REM ## F8 - A KEPERNVON
16 REM ## LEVO KARAKTEREK
17 REM ## SZINE
18 REM ##
19 REM #####
20
21 REM ##### BETOLTO #####
22
23 KEV1,"":KEY2,"":KEY3,"":
24 KEYS,""
25 FORI=0 TO 15#0-1:READ A#
26 A=DEC(A#) Q=0+A
27 POKE1792+I,A:NEXT I
28 IF Q<12052 THEN GOTO 30
29 SYS 1877:END
30 PRINT"HIRA A DATAKBAN":STOP
31
32 REM ##### ADATOK #####
33
34 DATA A5,06,C9,04,F0,0F,C9,05
35 DATA F0,11,C9,06,F0,13,C9,03
36 DATA F0,15,4C,0E,CE,EE,19,FF
37 DATA 4C,0E,CE,EE,15,FF,4C,0E
38 DATA CE,EE,3B,05,4C,0E,CE,EE
39 DATA 00,00,AD,00,00,A2,00,2D
40 DATA 00,00,9D,00,00,00,00,0A
41 DATA 9D,00,00,E8,00,F1,4C,0E
42 DATA CE,EA,00,00,8D,14,03,0A
43 DATA CE,8D,15,03,A2,00,4C,8E
44 DATA 8E,EA,EA,EA,EA,EA,00,8D
45 DATA 14,03,0A,07,8D,15,03,2D
46 DATA 88,08,A2,80,4C,8E,8E,EA
47 DATA EA,EA,00,00,00,00,00,00
48 DATA 00,00,00,00,00,00,00,00

```

A programmal a képernyő színeit változtathatjuk meg. Commodore 16-os és C plus/4-es számítógépen futtatható. A gépi kódú rész a BASIC memóriából egy bájtt helyet sem foglal el, mivel a \$700, című a magnóbufferben helyeztem el. A programot felhasználhatjuk saját programunk szubrutinjaként is. A színeket az F1+F2 + F3+HELP billentyűkkel állíthatjuk.

Mutschler Péter

Videomonitor a Junoszty-402 BC-ből

Szegény ember vízzel főz, avagy jobb híján egy televízió is megteszi. Mivel a tévé már megvolt, így én is ezt alakítottam át. Úgy csináltam, hogy továbbra is alkalmas a tévéműsorok vételére, illetve monitorozásra, szabvány DIN-csatlakozóval.

A videocsatlakozó-aljzaton van egy érintkezőpár, mely nyugalmi helyzetben zárt állapotban van. Ezt használtam fel egy 12 V-os relé kapcsolására. (Relének olyat válasszunk, amelyeknek több érintkező-párja van.)

A relét egy kisméretű panelra szereltem a DIN-csatlakozóval együtt. Ezt úgy készítettem el, hogy a DIN-csatlakozóra szegecselhető anyát (M3) raktam, így a panelt együtt a kívánt helyre erősíthető.

A tévékészülék hangfrekvenciás bemenetét megszakítottam (1. ábra) úgy, hogy közvetlenül a hangerő-szabályozó potenciométerre csatlakozhassam. (A rajzon ez a 2-es pont.) Ezt a fólia átvágásával oldottam meg. Átvágás után mind a két részre egy-egy hangfrekvenciális árnyékolt vezetékét forrasztottam, melyeken az árnyékolást szabadon hagytam. A vezetékek másik végét a relé megfelelő pontjaira kötöttem. Az árnyékolást „testeltem”.

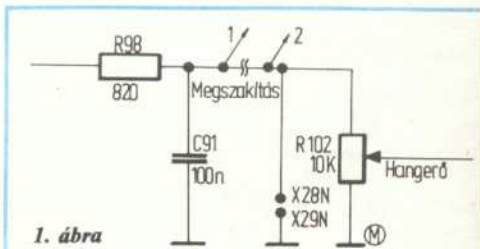
A videojel a demodulátor serlege alatt csatlakoztatható (2. ábra). Itt a megfelelő helyeken átvágtam a fóliát úgy, hogy az R38-as ellenállás a báziskörben maradt. VIGYÁZAT! Nálam a nyomtatási rajz és a panel maratása nem egyezett meg, ezért átvágás után az R38-as ellenállást át kellett forrasztani. A megfelelő pontokra (3-4, 5-6) 75 ohmos koaxiális kábelt forrasztottam. A kábeltől 3 méterre volt szükség. Ennyiből a belső vezeték és a csatlakozókábel is elkészült. (A vékony koaxiális kábel ára 170 forint méterenként.)

A kábelék másik végét a relé megfelelő pontjaira kötöttem (3. ábra). A relén helyeztem el az R és C alkatrészeket. A relé működtetéséhez szükséges feszültséget a stabilizátor panelről vettem, amelyet a videocsatlakozó érintkezőivel kapcsoltam össze. Ezután elkészítettem a csatlakozókábelt (4. ábra).

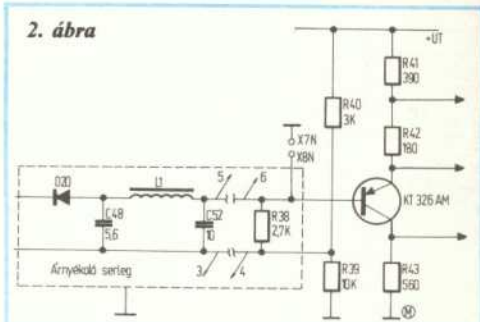
Ha a tévé bekapcsolásakor nem csatlakoztatjuk a számítógépet, akkor a beállított műsoroknak kell megjeleníteni. Ha a 6 pólusú tuchelt csatlakoztatjuk a tévéhez, az átkapcsol monitorra és a számítógépet bekapcsolva, annak bejelentkezését kell látnunk.

Ezzel el is készült a monitor. Érdeemes megcsinálni ezt az átalakítást, mert ezzel jó minőségű hanghoz és képhez jutunk. Én beépítettem a készülékbe egy fejhallgató-csatlakozót is, hogy a készüléket a család zavarása nélkül használhassam.

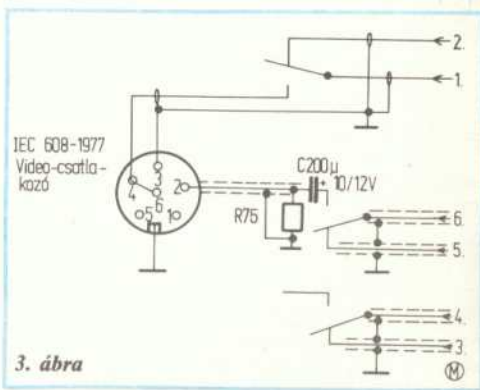
Bihácsi Gábor



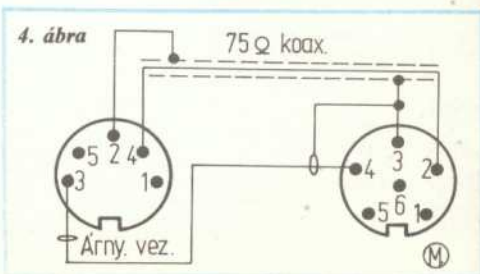
1. ábra



2. ábra



3. ábra



4. ábra



„KORMÁNYVÁLTÁS”

Ilyen például a MERCURY, aminek az átírásához az alábbi gépi kódú program szükséges:

Cím	Módosítás
\$07F8	80
\$B27F	A9 BF 20 40 FF 20 4A FF
\$B287	A9 DF 20 67 FF
\$FF40	8D 30 FD 8D 08 FF AD 08
\$FF48	FF 60 A2 FF 8E 90 FF 4A
\$FF50	80 07 48 A9 04 20 60 FF
\$FF58	68 4A 4A B0 08 A9 08
\$FF60	4D 30 FF 8D 30 FF 60 20
\$FF68	40 FF 4A B0 07 4A A9 02
\$FF70	20 60 FF 68 4A 4A B0
\$FF78	95 A9 01 20 60 FF A9 FD
\$FF80	20 40 FF 0A B0 05 A9 40
\$FF88	20 60 FF AD 90 FF 60 FF

A C Plus/4-es gépek billentyűzetének és botkormányának adatvonalai közös TED regiszterhez csatlakoznak. Ez az oka annak, hogy a programok, amelyeket csak botkormánnyal lehet irányítani, könnyen módosíthatók úgy, hogy billentyűzetről kezelhetők legyenek. Ehhez csak annyit kell tenni, hogy a megfelelő TED regiszter figyelését a \$DB70 címen kezdődő billentyűzetszámoló rutinra kell bízni. Az alábbiakban ismertetem néhány program módosítását:

A Program neve	Cím	Módosítás
BOOTY	\$1175	00 20 70 DB
HARVEY HEAD	\$1129	00
	\$115C	20 70 DB
BONGO	\$2C9D	00 20 70 DB
FURY	\$2712	00 20 70 DB
RUNNER	\$29E1	00 20 70 DB
INVADERS	\$1067	00 20 70 DB
BMX RACER	\$2030	00 20 70 DB
AUTOBAHN	\$383A	00
	\$3866	20 70 DB
STELLAR WARS	\$108A	00 20 70 DB
TREASURE ISLAND	\$7447	00 20 70 DB
	\$1610	00 20 70 DB
STARFORCE NOVA	\$2387	00 20 70 DB
	\$1534	00 20 70 DB
ROBO KNIGHT	\$281A	00 20 70 DB
MATRIX	\$13E0	A9 00 20 70 DB
AIRWOLF 16	\$2F85	00 20 70 DB
AUTO ZONE	\$1C3C	00 20 70 DB

Ilyenkor a botkormány helyett a kurzormozgató és a SHIFT billentyűk használhatók.

Davecz Ferenc

Görgetés TVC-re

A Mikroszámítógép Magazinban már jelent meg TVC-re scroll rutin, az 1986. decemberi számban. Ezt azonban túlságosan lassúnak találtam.

Ez a két rutin, amely jobbra, illetve balra görgetést valósít meg, már elég gyors ahhoz, hogy játékprogramokban is felhasználható legyen. A rutinok előnye az is, hogy mindhárom grafikus üzemmódban alkalmazhatók. A listán látható program a két rutinnal a memória bármely szabad területén működőképes. BASIC-ből az EXT 0, illetve az EXT 1 parancsokkal hívható.

Kócai László

```

1 |*****
2 | Gyors scroll jobbra/balra TVC-re
3 | (C)1988 BUDDHA Softouse
4 | Kócai László
5 |*****
10 KE=3735:I=0
20 READ A:IF A=-1 THEN GOTO 50
30 CH=CH+A:POKEKE+I,A:I=I+1
40 GOTO 20
50 IF CH=6392 THEN 60
55 PRINT"hiba a DATA sorokban!":END
60 PRINT"Kesz!!!":SOUND
70 POKE 33,3735 AND255:POKE34,3735/256
80 POKE 35,3759 AND255:POKE36,3759/256
90 PRINT"Jobbra:text0":PRINT"Balra: text1"
1000 DATA243,62,80,211,2,33,0,128
1010 DATA6,240,197,229,209,35,1,63
1020 DATA0,237,176,193,16,244,251,201
1030 DATA243,62,80,211,2,33,63,128
1040 DATA6,240,197,1,63,0,229,209
1050 DATA43,237,184,35,17,127,0,25
1060 DATA193,16,239,251,201,-1
    
```



A módosítás tehát mindössze annyiból áll, hogy a megfelelő program betöltése után belépünk monitorba és kiadjuk a > parancsot a fenti táblázatban szereplő operandusokkal. Például a BOOTY programnál a parancs:

> 1175 00 20 70 DB.
Ezután irányítására a billentyűzetet használhatjuk. A botkormány egy-egy irányra több billentyűvel is helyettesíthető:

Fel: <1> <9> <7> <5> <3>
Le: <CLR/HOME> <+> <P> <I> <Y> <R>
<W> <RETURN>
Jobbra: <2> <0> <8> <6> <4> <HELP>
Balra: <CTRL> <,> <L> <J> <G> <D>
<A> <E>

Tűzgomb

PORT1 esetén: <SHIFT> <@> <X> <X> <V>
<N> <,> </> <RUN/STOP>
PORT2 esetén: <E> <F3> <T> <U> <0> <-->
<+> <Q>

Egy kicsit bonyolultabb a megoldás akkor, ha egy program eredetileg botkormányt és billentyűzetet is használ.

Listánkat felhasználói, illetve játékprogramokból állítjuk össze. A legjobbakat, legérdekesebbeket a beküldött javaslatok alapján rangsoroljuk. Ehhez kérjük az olvasók közreműködését. C64-re, ZX-Spectrumra, Entertainise-ra, TVC-re, Atari-ra és IBM-re készült programsorokat várnunk havonta.

Címünk:
Mikroszámítógép Magazin
Szerkesztősége
1371 Budapest, Pf. 433
A diákszerkesztőség

TOP - lista

Felhasználói programok	TVC					Játék programok	C64				
	IBM	AMIGA	C-128	C-64	C-4 (G)		IBM	AMIGA	C-128	C-64	C-4 (G)
1 Alfa Paint						1 Menace M. II.					
2 Geos 1.6 d.						2 Lord of Mdn.					
3 Rookmon v3.0						3 Summer Ol.					
4 Alfa Cad +						4 LLL					
5 Windows						5 Kombók					
6 Disk manager						6 Super cobra					
7 Printfox						7 Barbarian II.					
8 Printmaster						8 Monopoly III.					
9 Print maker						9 Defender oto.					
10 Art studio 1.5e						10 Elite					

BBB

A számítógépek

Az operációs rendszerek

Oszályozási elvek

Az operációs rendszerek sok tekintetben különbözhetnek egymástól. A felhasználói igények, a különféle szolgáltatások színvonala, a rendszer vezérlésének módja, a hardverretek mind olyan ismérvek, amelyek alapján a rendszereket osztályozni lehetne. Például a „nagy-gépes”, „kisgépes” és „mikrogépes” csoportosítás a hardvert helyezi előtérbe, de nem mond semmit a rendszer működéséről. Ha „parancs-vezérelt” és „eseményvezérelt” rendszerekről beszélünk, túl nagy csoportokat alkotunk, mert csak egyetlen ismérvtől indulunk ki. Könnyen be látható, hogy nem egyszerű csoportosításra, hanem több szempont szerint felépíthető, sajtószerű rendszerre van szükség. A sok lehetséges szempont közül legalább a következőket célszerű figyelembe venni:

- a felhasználók számát (egy és több felhasználó),
 - a rendszer üzemdóját (kötegelt, időosztásos, valósidejű, többcélú),
 - a gép méretét (nagy-, kis- és mikrogépes).
- A fenti szempontok által meghatározott kombinációk közül néhány értelmetlen, mások már a feledés homályába merültek, a többieket pedig igyekeznünk figyelembe venni.

A fejlődés spirálja

Az ötvenes évek második felében a hardver a fejlett országokban is többszörösen meghaladta az emberi munka árát, ezért a legfontosabb feladat az volt, hogy a számítógéprendszert folyamatosan ellássák munkával. A megoldandó feladatokat (job) a megfelelő vezérlő információval együtt lyukkártyacsomagok rögzítették, és a kötegelt feldolgozás indulásakor a kártyaolvasóba helyezték. A feladatköteget egy viszonylag egyszerű operációs rendszer — amit gyakran monitornak is hívtak — fogadta. Tipikusan ilyen volt például a FORTRAN fordító hívása, az Assembler hívása és a kapott tárgyprogram elindítása. Minden feladat befejezése után a monitor visszakapta a vezérlést és beolvasta a kötegből (batch) a következő feladatot.

A ma is használt technikák közül már akkor megjelent a könyvtárszervezés, a programok áthelyezhetősége és a periféria függetlenségének az elve is. A könyvtár csupán kártyán, papíron, majd mágnasszalagon tárolt, a programozási nyelvhöz tartozó, általános szubrutinókat tartalmazott, az áthelyezhetőséget pedig az tette lehetővé, hogy a könyvtárban lévő programokat a memória természetleges helyére betöltve, végre lehetett hajtani.

A fejlődés során ezek a rendszerek elavultak, helyükre átadták a többlefelhasználás rendszereknek, de érdekes módon a személyi számítógépek megjelenése óta, természetesen új formában, ismét használni kezdtek ezeket. Mivel a hardver már olcsó, nem a gép munkával való ellátása, hanem az egyetlen felhasználó minél jobb kiszolgálása a cél, amit párbeszéd, barátságosan működő, segítőkész operációs rendszerrel lehetett legjobban elérni. Az egyfelhasználós rendszerek reneszánszában a CP/M (8 bites processzorok), az MS-DOS és a CP/M-86 (16 bites Intel processzorok) terjedt el legjobban. Különösen nagy kényelmet, de kisebb hatékonyságot nyújtanak a többszintű menürendszerrel, a menü teteleit ábrákkal, ikonokkal kifejező, a feladatválasztást pedig egyszerű rámutatással megoldó rendszerek (Apple Macintosh, Commodore Amiga). Ezeket dolgozza elsősorban a kezdő felhasználók szerezhetnek maguknak gyors sikerélményt.

A multiprogramozás alapelvei

Már a hatvanas évek elején nyilvánvalóvá vált, hogy a felhasználók soros kiszolgálása nem képes sem a központi egység, sem a peri-

feriák folyamatos terhelésére. Az alapötlet a kötegelt üzem keretei között született meg: megfigyelték, hogy egy feladat (angol neve: job) végrehajtásakor gyakran várni kell valamire (például a mágnasszalag cseréjére, a B/K művelet befejezésére stb.). Monoprogramozásnál a központi egység ilyenkor egyszerűen nem csinál semmi hasznosat, multiprogramozással viszont az operációs rendszer átkapcsol a következő feladatra. Az átkapcsolás — amennyiben az új feladat is várakozási állapotba kerül — további feladatok indítását is lehetővé teszi. Amikor a felfüggesztett feladatok közül valamelyik folytathatóvá válik, azt folytatni kell — természetesen az éppen aktív feladat felfüggesztése árán.

Mivel a multiprogramozásnál gyakorivá vált az a szituáció, hogy a memóriában több program tartózkodik, de ezek közül csak egy aktív, bevezetjük a „folyamat” (angolul: process) fogalmát, az éppen futó programnak a többitől való megkülönböztetésére. A folyamat tehát a program dinamikus megfelelője, program futás közben.

A térkezeléssel később foglalkozunk részletesebben, most inkább csak a megoldandó problémákat soroljuk fel. Az természetesnek tűnik, hogy a futtatandó programoknak áthelyezhetőnek kell lenniük, amit általános formában a logikai és a fizikai címtartományok bevezetésével lehet megoldani, aminek kulcskérdése a kettő közötti leképezés.

Megoldható a multiprogramozás úgy is, hogy egyidejűleg csak egy feladatot tartunk a memóriában, amely így az operációs rendszer által szabadon hagyott teljes területet urálhatja. Az ilyenkor használt technikát „swapping”-nek nevezzük. A swapping lényege, hogy a feladatváltáskor a felfüggesztett feladathoz tartozó teljes memóriaterület háttértárbá (lemezre) másolják, és onnan behozzák a következő feladatot.

Amikor a swappingnél bevezették a többszörös pufferhasználatot és a B/K tevékenységek idejét hasznos központi egység munkával fedték át, eljutottak a többparticiós multiprogramozáshoz. A memóriát több, különböző méretű, egybefüggő részre (partíciókra) osztották, amelyek mindegyikében egy végrehajtásra váró feladat foglal helyet. Attól függően, hogy a felosztás rögzített vagy változtatható volt, beszélünk fix vagy változó partícionálásról. Prototípusként az IBM OS/360-ban megvalósított algoritmusokat szokták említeni, amelyek megtellett angol nyelvű rövidítéssel MFT (fix partíciók) és MVT (változó partíciók). Mig az MFT algoritmus egyszerűbb, csak akkor hatékony, ha a számítógéprendszerrel állandóan közel egyforma feladatkeverékeket akarunk megoldani, az MVT bizonyultatva ütemezést és védelmet igényel, viszont bármilyen feladatkeverékhez képes alkalmazkodni.

Bármilyen ügyes algoritmusokat dolgoztak is ki a partícionált memória kezelésére, egy igen kellemetlen jelenség, az ún. elaprózódás ellen nem lehetett hatásosan védekezni. Egy bizonyos idő elmúltával a memóriában sok, össze nem függ apró terület alakult ki, melyek együttes mérete számottevő volt, mégsem lehetett őket hasznosítani. A megoldást a lapkezelés jelentette, amikor is a memóriát azonos méretű (például 1 kb-ot) lapokból állóknak tekintjük és minden feladathoz annyi — nem szükségszerűen folytonos fizikai cíneket elhelyezkedő — lapot rendelünk, amennyit igényel. A lapkezelő alrendszer fő feladata a felhasználó logikai tárgyienyű leképezni a fizikai lapokra.

Végül a lapkezelést általánosították a szegmentálással, ami tulajdonképpen változó méretű lapokból folytatott gazdálkodást jelent.

A multiprogramozás másik lényeges kérdése a központi egység ütemezése, ami nem más, mint a számítógéprendszer legfontosabb erőforrásával való gazdálkodás.

Az ütemezést a folyamatok különböző sorkorba rendezésével és a megfelelő sorkezelő algoritmusok segítségével oldják meg. Az aktív és várakozó folyamatok az ún. készenléti sorba kerülnek. Ezenkívül minden B/K egységhez tartozik egy sor, amely az egységre váró folyamatokat tartalmazza. Az ütemezés két, esetleg három szinten megy végbe. A magas szintű ütemező a külső tárolón várakozó feladatokból kiválasztja a készenléti sorba kerülőket, az alacsony szintű (központi egység) ütemező pedig ebből a sorból aktivizál egy folyamatot. A magas szintű ütemező célja az, hogy megfelelő, az erőforrásokkal egyenletesen terhelő feladatkeveréket állítson össze, amelyből azután a központi egység ütemezője valamilyen algoritmus alapján ellátja munkával a rendszert.

Az operációs rendszerek üzemdójáról szerinti felosztása rendszerint a kötegelt, az időosztásos és a valósidejű típusokat különbözteti meg.

Kötegelt (batch) rendszerek

A kötegelt elnevezés tulajdonképpen a számítógéprendszerhez való hozzáférés módjából ered. Az operációs rendszer egymástól független feladatokat végrehajtására vonatkozó igényeket fogad, ezekből olyan kötegeket (angol nevük: batch) hoz létre, amelyeket azután a saját, számára megfelelő időpontban hajt végre. Az igényeket kezdetben lyukkártyán, később csatornákon — terminálokon, távközlési vonalakon — is közölhették.

A kötegelt rendszerek alapvető sajátysága, hogy a programozókat elválasztja a géptől — általában a géptermekekől is kitiltották őket —, a rendszer kezelését pedig az erre a célra kiképzett gépkezelők, operátorok végzik. A programozó dolga, hogy egy futás eredményeiből minél gyorsabban előállítsa a következő futás anyagát, hiszen elsődleges cél a számítógéprendszer folyamatos terhelése. Ezt az egyfelhasználós rendszereknek csak kisebb-nagyobb mértékben érték el, s csak a multiprogramozás elterjedése, valamint egyes tevékenységek párhuzamosítása hozott igazi eredményeket.

A programozó által összeállított anyag lényegében egy „vezérlő nyelven” (job control nyelven) írott program, amely a saját végrehajtásához szükséges adatokat vagy ezek tárolási helyére való utalásokat is tartalmaz. A kötegelt rendszerek fejlettségét szintjét elég jól jellemzi az erőforráskezelés automatizálásának mértéke, ezen belül is a memória kezelése. A fix, majd a változó partíciók használata után a virtuális címzéssel új fejlődési szakasz vette kezdetét.

A kötegelt rendszerek azonban a fejlesztők minden igyekezetének ellenére nem tudták kielégíteni a változó felhasználói igényeket, és ezen még a nagy rendszerekhez illesztett telekommunikációs lehetőségek sem segítettek. Egyes területek — elsősorban az új szoftver-rendszerek fejlesztése — nem viselték el azt a két forduló közötti néhány órás várakozást sem, amit csak a jó felszerelt és jól szervezett számítógéppontok tettek lehetővé. A fejlesztők teljesítményt és munkájuk minőségét egy-

motorja II.

sztályozása

aránt javítja, ha a számítógéprendszerrel állandó és közvetlen kapcsolatban (angolul: on-line) vannak.

Időosztásos rendszerek

A kötegelt üzemmód gondjain igyekeznek segíteni az interaktív rendszerek, ahol a felhasználó állandó és folyamatos kapcsolatban van a programjával, annak írása és végrehajtása idején. A rendszer általában egy billentyűzetből és egy képernyőből álló terminállal átbevitt parancsok segítségével vezérelhető. A parancsok azonnal végrehajthatók és az eredménytől függően a programozó „on-line” dönt a következő lépésről. A hangsúly a gyors választáson van.

Az időosztásos (angolul: time-sharing) rendszer célja, hogy a számítógéprendszer interaktív felhasználókat több felhasználó számára párhuzamosan és ezért csökkentett költséggel, lehetőleg tegye. Ennek érdekében az operációs rendszer a központi egység megfelelő ütemezésével, valamint a multiprogramozás alkalmazásával a gép idejét több felhasználó között osztja meg. Különösen programok fejlesztésénél igaz az, hogy az éppen aktív folyamat rövid időn belül befejeződik vagy B/K műveletet igényel. Mivel az interaktív adatbevitel gyakran a billentyűzetről vehető meg, ami a gép szempontjából igen lassú, a rendszer gyakran változtatja a felhasználók kiszolgálását, mégis mindenki úgy érzi, hogy egyedül dolgozik. Az időosztásos rendszerek az erőforrások igazságos elosztása érdekében védekeznek az ellen, hogy egy folyamat — véletlenül vagy szándékosan — korlátlan ideig aktív maradjon. Erre a célra általában a gép órájának megszakítását használják oly módon, hogy az órajel valamilyen többszörösével kifejezhető „időszel” elmulásával másik folyamatra kapcsol át. Az időszel nagysága függhet az alkalmazási környezettől és a pillanatnyi terheléstől is.

Az időosztás elve már 1960-ban ismert volt, de a szoftver bonyolultsága és a technikai színvonal miatt mégsem terjedhetett el. Általában gyakorlati válassza mintegy tíz évig várható magára, azóta viszont népszerűsége egyre növekszik. A kötegelt rendszerek időosztásos alrendszerrel való bővítése mellett természetesen az időosztásos rendszerekben is gyakran gondoskodnak kötegelt üzemmódról, hogy a rutinszerű ismétlődő feladatok végrehajtását egyszerűsítsék. Ilyenek például az MS-DOS alatt végrehajtható „bat” kiterjesztésű állományok.

Az első közismertté vált időosztásos rendszert, a MULTICS-t 1965 táján egy amerikai egyetemeken fejlesztették ki a General Electric GE-645 típusú nagygépen. A fejlődés következő állomása a kisgepes kategória volt. Ennél az időosztásos rendszerek „prototípusa”, a Unix, sok eredeti megoldáson kívül jócskán merített a MULTICS ötleteiből is. A Bell Laboratóriumban kidolgozott rendszer eredetileg a DEC PDP-11 családjára készült el. Azóta a Unix-szerű rendszerek halmaza nagyon kibővült, a mikrogepek elterjedésekor pedig a CP/M, MP/M, majd az MS-DOS egyaránt Unix mintára készült.

Valósídejű rendszerek

Egyes speciális feladatok gépesítése olyan igényeket keltett, amelyek a valósídejű rendszerek kialakulásához vezettek. Egy ilyen rend-

szer tulajdonképpen úgy tekinthető, mint egy egyedi alkalmazás vezérlő mechanizmusa. Az adatok a gépbe érzékelőkről — általában mérőműszerekről — érkezik. A rendszer feladata az adatok elemzése és ha szükséges, olyan vezérlő mechanizmusok aktivizálása, amelyek befolyásolhatják az érzékelők által mért értékeket. Tipikusan valósídejű rendszert alkalmaznak például a forgalmi rendszerek (közlekedési lámpák) vezérlésében, az ipari folyamatok, tudományos kísérletek irányításában stb. A rendszer közös jellemzője, hogy a feladatok elvégzése szigorúan időhöz kötött. Rendszerhívással egyenértékű, ha egy feladat nem hajtható végre a hozzá rendelt időkorlátban belül. Amíg a kötegelt operációs rendszernek lényegtelen, az időosztásos rendszereknél fontos, de nem meghatározó, addig a valósídejű rendszereknél az időkorlát döntő jelentőségű.

Bár a valósídejű rendszerek logikája az emberi beavatkozással kapcsolatos követelményeket minimalizálni igyekszik, ezek teljesen mégsem nélkülözhetők. Különösen a rendszer paramétereinek beállításakor, az értéktartók túllépésekor, vagy hibás működéskor van emberi beavatkozásra szükség. A hangsúly nem az erőforrások optimális kihasználásán, hanem a biztonságos üzemén van. A teljesítményparamétereket úgy kell megválasztani, hogy az időkorlátok csúcsterhelésnél is tarthatók legyenek. Ennek érdekében gyakori a hardverredundancia és az, hogy a szabványos processzorkor egy speciális, módosított változatot alkalmaznak.

A teljes alkalmazási rendszerben a számítógépes alrendszer néha állítható észre. A programokat általában máshol fejlesztik, ide gyakran már csak megváltoztathatatlant, végrehajtható formában, ROM-ba égetve kerülnek. A valósídejű alkalmazások szoftverje a szokásosnál több szempontból is bonyolultabb. Egyrészt az egész programrendszernek gyakorlatilag hibamentesnek kell lennie, másrészt a programok belovését nem lehet a megszokott módon elvégezni, mivel a próbához szimulálni kell a teljes környezetet, az összes bekövetkezőt eseményekkel együtt. Erre azért van szükség, mert hibába működik egy program hibátlan laboratóriumi környezetben, arról is meg kell győződni, hogy minden folyamat akkor is az időkorlátján belül marad, ha a rendszer teljes terheléssel üzemel.

Az operációs rendszerek külön kategóriáját alkotják a több számítógép együttműködésén alapuló hálózatok.

Hálózatok és elosztott rendszerek

A számítógépes hálózatokban önálló rendszerek vesznek részt, amelyek különböző adatátviteli vonalak kapcsolnak össze: közvetlen adatbuszok, telefonvonalak stb. A hálózatnak az a célja, hogy a feladatokat észszerűen megosszák a tagok között, ami nyilvánvalóan költséges előnyökkel jár. Erre utal az elosztott vagy osztott (angolul: distributed) elnevezés.

Az elosztott rendszerben részt vevő számítógépek méretüket, teljesítményüket és funkcióikat tekintve igen változatosak lehetnek: mikroprocesszorok, terminálok, kisgépek, nagy, univerzális számítógépek stb. Az egyöntetűség kedvéért ezeket mint a hálózat csomópontjaira vagy egyszerűen csak mint gépekre hivatkoznak majd.

Elosztott rendszerek létrehozását négy fő cél indokolja: az erőforrások megosztása, a sebesség növelése, a megbízhatóság és a kommunikáció. Mivel ezek minden elosztott rendszerre vonatkoznak, a tárgyalást velük kezdjük.

Az erőforrások megosztása. Mivel sok, különböző lehetőségekkel felszerelt csomópont kapcsolódik össze, az egyik csomóponton használható olyan erőforrásokat, amelyek helyileg egy másik csomóponton vannak. Például az A gép használhatja a B géphez kapcsolt lezényamatot, miközben B dolgozhat az A-ban tárolt adatállományból.

A sebesség növelése. Ha egy nagy volumenű számítási feladatot konkurens módon futtatható részekre lehet bontani, az egyes részek szétosztathatók a csomópontok között és párhuzamosan hajthatók végre. Emellett a túlterhelt csomópontokról a munka egy része más gépekre irányítható (load sharing).

A megbízhatóság. Ha az elosztott rendszer egyik csomópontjában géphiba fordul elő, a többi csomópont elvileg működőképes marad. Ha a rendszer több nagy, önálló számítógépből áll, a gép kiesése nem nagyon zavarhatja a többi. Ha viszont a hálózat sok kis gépből épül fel, melyek mindegyike speciális funkciót lát el (például fájlrendszerek, terminál B/K), egy gép hibája megbéníthatja az egész rendszert. Általában igaz, hogy ha a rendszerbe elegendő redundanciát építettek be — mind a hardver, mind az adatok szempontjából —, egy vagy néhány kiesést a hálózat jól elviseli. Különösen bonyolult kérdéskör a hiba automatikus észlelése, átterés a sérült rendszerre üzemére, majd — ha elhárítása után — visszatérés a normális működésre.

A kommunikáció. Az összekötött csomópontok közötti információcsere általában a postahivatalok mintájára szervezettek meg, és „elektronikus postának” hívják. A hálózat minden felhasználója kap egy „postaládát”, amelyhez a csomóponton egyedi név tartozik. Postát bárki küldhet bárkinél a csomóponton belül, vagy más csomópontokra is. A cím a postaláda és a csomópont nevéből tevődik össze. Az elküldött postákat az operációs rendszer a címzett postaládájában tárolja. Elosztás után a címzett intézkedhet az üzenet töröléséről, tárolásáról, esetleg további címekre való továbbításáról vagy a válaszról.

Az elosztott rendszerek csomópontjait többféleképpen lehet összekapcsolni. A különböző konfigurációk kialakításánál elsősorban három fő szempontot szoktak figyelembe venni:

— A létesítési költséget. Mennyire költséges az egyes csomópontok közötti kapcsolat kiépítése?

— A kommunikációs költséget. Mennyi ideig tart egy üzenet két csomópont közötti átvitelét?

— A megbízhatóságot. Ha hiba lép fel egy csomópontban vagy egy összekötő vonalban, a rendszer maradék csomópontjai még kapcsolatban maradnak-e egymással?

A teljes kapcsolású hálózatban bármely két csomópont között közvetlen kapcsolat van. A létesítési költségek nagyok, ugyanakkor a rendszer igen megbízható és a kommunikációs költségek minimálisak.

A részleges kapcsolású hálózatban több direkt kapcsolat hiányzik. Az üzeneteket esetleg közbelső csomópontokon keresztül kell továbbítani. A létesítési költség kisebb, az üzemelés lassúbb és a megbízhatóság is csökken.

Külön említendő azok a hálózatok, amelyeknek közös információbusz van. Ezeknél a csomópontok egymással a közös vonalon át kommunikálnak. A létesítési költség a csomópontok számával arányos, a kommunikáció meglehetősen olcsó, de a busz átviteli sebessége könnyen szűk keresztmetszetre válhat. A rendszer működését egy-egy csomópont hibája nem zavarja, viszont a közös vonal megbízhatósága létfonosságú.

Az elosztott rendszereket két nagy csoportra oszthatjuk: számítógépes hálózatokra és helyi hálózatokra. Bár a felosztás elsősorban a csomópontok földrajzi eloszlására vonatkozik, az eltérő igények nyomot hagytak az operációs rendszerek felépítésén is. A számítógépes hálózatok több, önálló gépből állnak, amelyek nagy területen — például egy országban vagy földrészen — szóródnak szét, a helyi hálózatok pedig csak egy épületben vagy épületrészletben telepített processzorokból állnak. Igen népszerűek az IBM PC alapú helyi hálózati operációs rendszerek, mint a PC-NET, a NOVELL.

Programozási fogások és

melléfogások



Az elmúlt két alkalommal olyan programokat mutattam be, melyeknél az algoritmus helytelen megválasztása a szükségesnél több százszor hosszabb futási időhöz vezetett. Joggal kérdezheti bárki, hogy érdemes-e időt, energiát fordítani kijavításukra, hiszen csak egyszer futnak, és gyakran a végrehajtásra fordított idő többszöröse van szükség a program megírásához. Ebben van némi igazság, de gondoljunk arra, hogy ha valaki hozzászokik a felületes programozási stílushoz, az akkor sem tud jobbat produkálni, ha akar. Arról se feledkezzünk meg, hogy mindkét program egy-egy pályázati feladat „helyes” megoldásaként jelent meg, feltehetően oktató-bemutató céllal, ami viszont komoly hiba, mert a rossz programozási stílus vírusát terjeszti.

Mostani példánk Dusza Árpád—Varga Antal: A BASIC nyelvű programozás ábécéje című könyvéből való. Abban különbözik a korábbiaktól, hogy ez a program már nem egyszeri használatra készült.

A 30. oldalon a legnagyobb közös osztó kiszámításához javasolnak egy algoritmust, a következőképpen:

„A bemutatott algoritmus előnye az, hogy a gép aritmetikai egységét alig-alig „dolgoztatja”. Az alapgondolat az, hogy ha egy S szám osztója A-nak és B-nek is, akkor osztója a két szám különbségének is. Ez a különbség mindig kisebb, mint A és B közül a nagyobb, legyen ez például A. Ezután B-t és a-b-t vesszük az eredeti két szám helyett, és ismételjük az előbbi eljárást mindaddig, amíg a két szám egyenlő nem lesz. Belátható és bizonyítható, hogy ez a végső érték lesz a két szám legnagyobb közös osztója.”

A könyv 90. oldalán — az algoritmus újabb részletes leírását követően — megtaláltam az 1. listán látható programot. Az ELSE utasításra való tekintettel C16-os gépen próbáltam ki. A 999 992 és 999 999 számok legnagyobb közös osztójaként 7-et kaptam, erre több, mint huszonöt percet vártam. Nem értettem, hogy mivel telhetett el ez a rengeteg idő, hiszen a program az aritmetikai egységét alig-alig vette igénybe — legalábbis a könyv szerint. Mindenesetre megpróbáltam az ismételt kivonás helyett a maradékos osztás algoritmusát alkalmazni. Így jutottam a 2. listán látható programhoz, amely a másodperc tört része alatt megadta a helyes eredményt. Elképzelni sem tudom, hogy a könyv szerzői miért nem ezt a megoldást választották, hiszen két helyen is szerepel a könyvben maradékos osztást bemutató program (az egyik az 55., a másik a 96. oldalon).

A programok kipróbálásakor néhány rendellenességre is rábukkantam. Az eredeti program végtelen ciklusba kerül, ha az egyik szám nulla, és mindkét változat hibás eredményt ad egy vagy két negatív szám esetén. Az ELSE használatának nem örültem túlzottan, ugyanis sok géptípuson nem használható. Ennek ellenére nem tekinthető hibának, hiszen a könyv — ha a címéből ez nem is derül ki — a HT-1080Z BASIC nyelvű programozásáról szól.

Az újabb javítás a 3. listán látható. Ez már egyetlen speciális esetet kivéve, amikor mindkét szám nulla, hibátlanul

működik. A 30-as sorban zárjuk ki a negatív számokból eredő problémákat. Az ELSE utasítást úgy hárítjuk el, hogy mindig A-ba tesszük a nagyobb számot, ha B-ben van nagyobb, akkor a két változó tartalmát felcseréljük.

Valamivel kevesebbet kell begépelni, ha a program két sorát a 4. listán bemutatott módon megváltoztatjuk. Ekkor a futási idő csekély mértékben nő, de így is 0,3 másodperc alatt marad.

Barna László

1. lista

```
10 REM LEGNAGYOBB KOZOS OSZTO
20 INPUT "A KET SZAM";A,B
30 IF A=B THEN 60
40 IF A>B THEN A=A-B : ELSE B=B-A
50 GOTO 30
60 PRINT "A LEGNAGYOBB KOZOS OSZTO";A
```

2. lista

```
10 REM LEGNAGYOBB KOZOS OSZTO
20 INPUT "A KET SZAM";A,B
30 IF A=0 OR B=0 THEN 60
40 IF A>B THEN A=A-INT(A/B)*B
   : ELSE B=B-INT(B/A)*A
50 GOTO 30
60 PRINT "A LEGNAGYOBB KOZOS OSZTO";A+B
```

3. lista

```
10 REM LEGNAGYOBB KOZOS OSZTO
20 INPUT "A KET SZAM";A,B
30 A=ABS(A) : B=ABS(B)
40 IF B>A THEN X=A : A=B : B=X
50 IF B=0 THEN 80
60 X=A-INT(A/B)*B : A=B : B=X
70 GOTO 50
80 PRINT "A LEGNAGYOBB KOZOS OSZTO";A
```

4. lista

```
...
60 A=A-INT(A/B)*B
70 GOTO 40
...
```


IBM XT/AT

„egy

Ékezetes szövegek
átalakításaNyomtatás
az egyben”

Az egyik ékezetes szövegszerkesztőt használva ért az a furcsa meglepetés, hogy kinyomtatáskor az ékezetes betűk helyett mindenféle kriksszkraksz jelent meg a papíron. Lévé, hogy a megírt levél elég hosszúra sikeredett, sajnáltam volna annak információ-tartalmát veszni hagyni. Ezért elhatároztam, írok egy olyan rövid gépi kódú programot, amivel kiküszöbölhető a hiba.

Hogy ez másoknak is gondot okoz, az látszik többek között abból, hogy jelennek meg a Mikroszámítógép Magazinban is olyan cikkek, melyekben például a zárójel helyett É vagy Ü betű szerepel, ami természetesen nem emeli a szöveg olvashatóságát (lásd többek között a Magazin 1988/9. számának 38. oldalán a Ki ad magyarázatot? című cikket).

Az ékezetes betűk helyett azért jelennek meg mindenféle furcsa jelek a nyomtatásban, mert az általam használt ékezetes szövegszerkesztő a képernyőn más karakterkódokat rendelt hozzá az egyes ékezetes betűkhöz, mint amivel a nyomtató megjelenítette őket a papíron. Így az É betűhöz a képernyőn a 144 (hexadecimálisan 90h) kódszám tartozott, de a nyomtató ugyanezt a betűt a 255 (E1h) kódszámmal tudta a papírra kirni.

A megoldáshoz ezek után már egyszerű út vezet. Írni kell egy olyan gépi kódú programot, amely egy megadott szövegből kikeresi az ékezetes betűket és felcseréli őket a nyomtató által megkívánt, kiterjesztett ASCII kódokra. Teljes egyezés tapasztalható a nem ékezetes betűk kódszámainál, melyeknek kódszáma kisebb 128-nál (80h-nál), és tulajdonképpen ezek az igazi ASCII kódok, amik minden számítógépen változatlanok. A kiterjesztett ASCII kódok már változatosabb képet mutatnak. A különbö-

ző gyártók ugyanis általában eltérő kódokat kapcsolnak hozzá ezekhez a 128-tól (80h-tól) 255-ig (FFh-ig) terjedő számkódokhoz. Ekként nyílik lehetőség arra, hogy magyarul vagy svédül vagy akár kínaiul írjunk meg egy szöveget a számítógép képernyőjén vagy nyomtatóján.

Az itt ismertetendő gépi kódú program hét ékezetes kisbetűt (ü, é, á, í, ó, ú, ö), valamint négy nagybetűt (Á, É, Ó, Ü) keres ki az IBM XT/AT adott szöveges fájljából, és felcseréli azok képernyőkódjait a megfelelő nyomtatókódokra.

Az Intel 8088 processzorcsalád (8088, 8086, 80186, 80286) gépi kódú programjának listáján látható, hogy a KÉPKÓD táblázat tartalmazza az átalakítandó képernyőkódokat a következő sorrendben: „ü é Á É ö Ö ú í ó ú”. A NYOMKÓD táblázat ugyanezen betűk nyomtatáshoz szükséges kódjait tartalmazza az Epson RX-80 nyomtatónak, amelybe eleve beépítették a teljes magyar betűkészletet.

Mind a két táblázatot tovább lehet bővíteni, ha ez szükséges, mivel további 21 karakternek foglaltunk le helyet a táblázatokban. Ha bővítenénk, a BETU jelenlegi 000bh tartalmát is — azaz a decimális 11-et — meg kell növelni a hozzáadott karakterek számával. Megjegyezzük, hogy ilyenkor érdemes a két kódszámtáblázatban a betűk sorrendjét úgy megválasztani, hogy a leggyakrabban használt ékezetes betűk a táblázat végénél helyezkedjenek el, mivel a program a táblázatban szereplő kódszámokat hátulról visszafelé olvassa ki és hasonlítja össze az átalakítandó szövegrész egyes betűivel (lásd a lista 015C—016C sorait).

A program használatához szükség van a DEBUG programra és annak ismeretére. A DEBUG programot minden IBM kompatibilis XT vagy AT szá-

mitógépen megtaláljuk az operációs rendszerünkben, a DOS-ban. Figyelem! Annak folytán, hogy az átalakítandó fájl olvasását és lemezre írását a DEBUG segítségével végezzük el, nagyon ajánlatos az átalakítandó szöveget egy üres hajlékonylemezre felírni; semmiképpen se a merevlemezre készítsük el az itt ismertetendő műveleteket, mert azokkal esetleg tönkretethetnénk a merevlemez tartalmát.

Legyen az átalakítandó szöveg neve a hajlékonylemezre LEVEL.TXT. Erről is készítsünk másolatot az átalakítás előtt. A listán lévő programot egy szövegszerkesztővel kell először megírunk, aminek adjuk mondjuk a KONVERT.ASM nevet. Ezt a programot a MASM.EXE program segítségével alakíthatjuk át tárgykóddá — ez lesz a KONVERT.OBJ —, amit a LINK programmal tehetünk futtatható gépi kóddá: KONVERT.EXE.

A DEBUG segítségével közvetlenül is a gép memóriájába vihetjük a lista bal oldalánál található csupaszigépi kódot is, amit a memóriából a következő utasításokkal írhatunk fel hajlékonylemezre. Ehhez azonban előbb a CX tárolóba helyezzük el az átalakítandó fájl bajtjainak a számát.

Tételezzük fel, hogy a BX és a CX tartalma 0-val egyenlő:

```
— R cx
```

```
CX 0
```

```
:082
```

Ezek után foghatunk hozzá a konvertáló program lemezre írásához:

```
— N a:KONVERT.EXE
```

```
— W cs:100
```

ahol az első utasítással beállítottuk a DEBUG számára a fájl nevét, majd a második utasítás kiírta azt a memóriából egy cserélhető hajlékonylemezre.

Amennyiben még nincs a programunk a gép memóriájában, akkor a

Microsoft (R) Macro Assembler Version 5.00
FILE KONVERTALASA EPSON RX-80-ra

1/1/80 00:19:24
Page 1-1

page 70,80
title FILE KONVERTALASA EPSON RX-80-ra

comment / A kepernyon megjelenő ekezetes betűk ASCII kódja nem egyezik meg az EPSON RX-80 ekezetes betűinek ASCII kódjaival. Ez a program kikülvassa egy ASCII szövegből az ekezetes betűk kódjait és a nyomtató által megkívánt kóddal helyettesíti azt. A KEPKOD táblázat tartalmazza a kepernyon megjelenő ekezetes betűk kódszámait, a NYOMKOD táblázat pedig ugyanezen betűk kinyomtatásához szükséges kódszámokat tartalmazza. /

.list

```

0000      F0F0B      segment
          assume cs:F0F0B,ds:F0F0B,ss:F0F0B
          ;
0100      org 100h
0100      01 82 BF 90 94      KEPKOD      db 81h,82h,8fh,90h,94h
0105      99 9A A0 A1 A2 A3      db 99h,9ah,0a0h,0a1h,0a2h,0a3h,21 dup(?)
          0015[
          ??
          ]
0120      F6 EC E0 E1 EF      NYOMKOD      db 06fh,0ech,0e0h,0e1h,0efh
0125      E4 EB ED EE EF      db 0e4h,0e6h,0ebh,0edh,0eeh,0ef3h,21 dup(?)
          0015[
          ??
          ]
0140      000B      BETU      dw 000bh
          ;
0142      1E          START:      push ds          ; ds-t es cx-et
0143      51          push cx          ; elmentjük
0144      8C C8      mov ax,cs       ; cs lesz az új ds
0146      8E D8      mov ds,ax       ; majd pedig az
0148      33 C0      xor ax,ax       ; index regisztereket
014A      8B F0      mov si,ax       ; nullulazzuk.
014C      8B F8      mov di,ax
014E      8B 0400     mov bx,400h     ; az átalakító
          ; szöveg elejét itt tesszük bele
          ; a bx regiszterbe.
          ; cx-et a DEBUG-ból kell a program
          ; futtatása előtt beállítani.
0151      8A 00      KERES:      mov al,[bx][si]
0153      3C 81      cmp al,81h
0155      72 B1      jc UJBETU
0157      3C A3      cmp al,0a3h
0159      77 1D      ja UJBETU
015B      51          push cx
015C      8B 0E 0140 R  mov cx,[betu]   ; 11 konvertálható betű
0160      8B F9      mov di,cx       ; van a KEPKOD-ban
    
```

```

0162      4F          dec di          ; di (0 - A)-ig terjed
    
```

Microsoft (R) Macro Assembler Version 5.00
FILE KONVERTALASA EPSON RX-80-ra

1/1/80 00:19:24
Page 1-2

```

0163      8A A5 0100 R      UJKEPKOD:      mov ah,ds:KEPKOD[di]
0167      3A C4      cmp al,ah       ; KEPKOD-on belüli ciklus
0169      74 06      je BETUCSERE    ; a konvertálható betűt
016B      4F          dec di          ; keresi ki belőle
          loop UJKEPKOD ; ha nincs meg, akkor
          cll          ; vissza UJKEPKOD-ra, különben
016E      FB          jnc NINCSBETU ; betucseré NYOMKOD-ból
016F      73 06      jnc NINCSBETU
0171      8A B5 0120 R      BETUCSERE:      mov al,NYOMKOD[di]
0175      8B 00      mov [bx][si],al
0177      59          NINCSBETU:      pop cx
0178      46          UJBETU:         inc si
0179      E2 D6      loop KERES
017B      59          pop cx          ; vége a szövegresznek
017C      1F          pop ds          ; regi cx-et es ds-t
017D      84 4C      mov ah,4ch      ; visszarjuk a veremől
017F      CD 21      int 21h         ; visszatérés a DOS-hoz
0181          F0F0B      ends
          end START
    
```

0 Warning Errors
0 Severe Errors

DEBUG segítségével először töltsük be a számítógépbe a KONVERT.EXE programunkat.

—N KONVERT.EXE

—L 100

E két utasítás után már a gépi kódú programunkat a DEBUG az aktív alkönyvtárból a gép memóriájába tölti. Ezután meg kell adnunk azt a fájlnév, amit át akarunk alakítani, nálunk a LEVEL.TXT-t, az előzőkhez teljesen hasonló módon:

—N a:LEVEL.TXT

—L cs:400

Ezzel az utasítással a szöveg tartalmát a gépi kód által megkívánt területre töltöttük be. A betöltés után a CX tartalma megegyezik a szöveg bájtjainak a számával. Most már nincs más teendőnk, mint működésbe hozni programunkat a következő utasítással:

—G = 142

Ellenőrizhetjük a DEBUG —R utasításával, hogy a CX tároló tartalmát változatlan maradt-e. Amennyiben igen, akkor most már visszairhatjuk az átalakított szöveges fájlt a hajlékonylemezre. A konvertáló program működésének ideje alatt nem rontja el a CX korábbi tartalmát. A CX tartalma csak akkor változott meg, ha a vizsgálódásunk közben mi magunk változtattuk meg véletlenül a DEBUG-gal:

—W cs:400

Figyelem! Ez az utasítás a korábbi fájlt felülírja a lemezen, így azt elveszítjük! Amennyiben korábban nem készítettünk másolatot róla, de még szükség lenne az eredeti szövegre, akkor a fenti utasítás előtt adjunk új nevet a fájlnak, például az

—N a:LEVEL.KNV

utasítással, és csak ekkor adjuk ki a

—W cs:400

parancsot.

Tovább már nincs szükség a DEBUG programra (—q utasítással visszatérhetünk a DOS-hoz), és a szöveges fájlnak akár a DOS, akár egy szövegyszerkesztő PRINT parancsával kinyomtathatjuk.

Mire lehet jó még ezenkívül ez a konvertáló program? Elképzelhető, hogy titkosíthatjuk vele a programunkat: az átalakított programot tárolnánk a lemezen, és egy betöltő program gondoskodikna a program vissza-konvertálásáról és elindításáról, letiltva egyúttal a program futását leállító CTRL BREAK gombok működését is.

Szabó Péter Pál

Programmódosítás futás közben

Az elmúlt időkben csaknem évente új személyszámítógép-típus jött divatba. Ezek BASIC-jei — ha már a BASIC-nél maradunk — egymástól bizonyos tulajdonságokban eltérők. Jó lenne egy olyan „öszvér BASIC”, ami mindenből a legrtakosabbat tartalmazná.

Az Enterprise komfortos BASIC-je igazán nem lehet panasz. Van ugyan egy-két szokatlan furcsasága, de ez is inkább csak azokat zavarhatja, akik egy másik számítógép BASIC tájékoztatást szövegeket meg. Amit minden Sinclair-en nevelkedett gépfelhasználó hiányol a gépek kulcsszószövegéből — így az Enterprise-éből is —, az a VAL függvény Sinclair-i értelmezése. Ott ugyanis az interpreter programfutás közben a bemenetként beírt FS sztring értékét úgy határozza meg, hogy elvégzi a műveleteket, és behelyettesíti a változó(k) aktuális értékét is. Ez a lehetőség kényelmes megoldás a különböző függvények kiértékeléséhez, ábrázolásához,

hiszen nem kell a programsorokat újra beírni vagy módosítani.

A különböző személyi számítógépekhez olyan segédrutinokat készítették (több megjelent a Mikroszámítógép Magazinban is), amelyek programfutás közben felülírják az értékelendő utasítást tartalmazó sort a bemenetként beírt függvénnyel. A felülírás természetesen a tokenizálás szabályait figyelembe véve ad megfelelő eredményt.

Akik az Enterprise programsortárolását már tanulmányozták, azok láthatták, hogy elég összetett szabályok alkalmazásával kerül a program a memóriában a végleges helyére. Ha nem akarunk a gép lelkivilágába beavatkozni, akkor természetesen semmi közünk, semmi tennivalónk nincs, hiszen a gép rendszerprogramja elvégzi a megfelelő lépéseket. Lépéseket, hiszen a beírt programsorként értelmezhető szöveg csak több rutinnal tárolható. Ezek a rutinok az 5. szegmensen találhatók. Így például a reláció- és műveleti jelek átálakítását, tokenizálását (az adott szegmsten a 3. lapon elkezelve) a C536H—C61CH címen tárolhatjuk.

Érdekes beírni egy programsort:
 10 IF A <= 0 AND B = 1 < AND C > < 3
 THEN GOTO 10

már csak azért is, hogy listázás után lássuk, a gép ekkor már öntörvényei szerint írja ki a relációjeleket a neki tetsző sorrendben, hiszen a két karakter <= tokenként 16H értékkel tárolódik, a programsor beírásánál követett sorrendtől függetlenül.

Amit ezek a tokenizáló rutinok minden lehetséges — és lehetetlen — este pillanatok alatt elkészítenek, azt mi is megtehetjük a géppel, akár BASIC programmal is.

Az itt közzét programmal a programfutás közben sztringként beírhatunk egy tetszőleges függvénykapcsolatot, amit azután a program kiértékel. Minthogy a program futás közben felülírja önmagát, és a változótablett is használja, csak alapértelmezésű — bekapcsolás utáni — számítógép-állapotban használható sikeresen.

A függvény a 110-es sorba kerül a ! jel helyett kezdődő memóriacímre, ezért az első két sor megválasztása, elírása esetében a 4845 dec címet meg kell változtatni. A 130-as soron bemenetként beírt függvénykapcsolat tetszőleges műveleteket és a 750—760-as sorban felsorolt függvényeket tartalmazhatja, függetlenül változóként pedig az X-et.

Ez a BASIC program utánozza a rendszerprogram tokenizálási folyamatát. A 400-as sorban kezdődő rutin azonosítja és tokenizálja a záró és műveleti jeleket. Az 500-as soron kezdődő programozás felismeri a függvényeket, és tárolja azokat.

Amennyiben a tokenizálás közben számot talál a program, a számot olyan formára alakítja, hogy az programsorban tárolható legyen. Ezt a munkát a 600-as soron induló program végzi el, minthogy az Enterprise a 10 000-nél kisebb egész számokat és a valós számokat másként tárolja, valamint a szám típusának jelzésére szolgáló érték beírása más alprogramot vesz igénybe.

Az egészeket a 640—670-es sorok kódolják, a valósakat pedig, amit az LJ segédváltozóban a változóértelre tárolunk, a 680—730-as sorok visszatöltik onnan a programba.

Az ennyi tortúrán keresztülment FS helyett most már előállította a program azt a bájtsorozatot, amely már majdnem olyan, mint egy értékelendő programsor. Akkor nincs más hátra, mint hogy ezt a PS-nek nevezett adatsort a 110-es sorba betöltsse a program. Ezt végzi el a 300—370-es sorok közötti rutin. Ezután a függvény értelmezési tartományának ismeretében a képernyőre kerül a függvénykapcsolat érték-táblázata. Az újabb függvények itt most addig közelíthetők a programmal, amíg le nem állítjuk annak futását, mert a 250-es sor végtelenségig ismétli a programot.

A vállalkozó kedvűek természetesen tovább finomíthatják a programot, és az egyszerű érték-táblázat-készítés helyett más feladatokra is felhasználhatják.

Dr. Sz. Lukács János

```

100 GOTO 130
110 LET Y= ! ha jot akarsz az első...4
sort így írd be!!!
120 RETURN
130 LET LJ=2,2:PRINT "y=";INPUT FS
140 LET FS=LCASE$(FS)
150 GOSUB 400
160 GOSUB 300
170 PRINT "Min=";INPUT X0
180 PRINT "Max=";INPUT X1
190 PRINT "Ispes=";INPUT L
200 FOR I=X0 TO X1 STEP L
210 GOSUB 110
220 PRINT X,Y
230 NEXT
240 PRINT
250 GOTO 130
300 ! sor beiras
310 FOR I=1 TO LEN(PS)
320 LET S$=P$(I):LET C=4845+I
330 POKE C,ORD(S$)
340 IF S$="^" THEN POKE C,27
350 NEXT
360 POKE I+4845,1
370 RETURN
400 ! kvazi tokenizalas
410 LET D=LEN(F$):LET P$=""
420 FOR V=1 TO D
430 LET S$=F$(V:V)
440 IF S$="A" THEN GOSUB 500
450 IF S$="(" THEN S$=")" THEN LET P$=CHR$(ORD(S$)-32)
460 IF S$="^" THEN LET P$=P$S$
470 IF VAL(S$)>0 THEN GOSUB 600
480 NEXT
490 RETURN
500 ! függvény kereses
510 RESTORE 750
520 READ D
530 FOR V=1 TO D
540 READ B$
550 IF F$(V:V+2)=B$ THEN LET P$=P$B$
560 NEXT
570 LET F$(V:V)="X" THEN LET P$=P$B$!"X"
580 RETURN
600 ! szam atalakitas
610 LET LJ=VAL(F$(V:V))
620 LET H=LEN(STR$(LJ))-1
630 IF INT(LJ)<>LJ OR LJ>9999 THEN 600
640 LET E$=CHR$(MOD(LJ,256))
650 LET P$=CHR$(INT(LJ/256))
660 LET P$=P$CHR$(INT(LJ/65536))
670 GOTO 750
680 LET P$=P$CHR$(190)
690 LET K=PEEK(566)+PEEK(567)*256+6
700 FOR R=K TO K+5
710 LET P$=P$CHR$(PEEK(R))
720 NEXT
730 LET V=V+H
740 RETURN
750 DATA 10,ABS,ATN,COS,COT,EXP
760 DATA LOG,SGN,SIN,SGR,TAN
    
```

Rekurzív növény

Ez a programcska az Enterprise számítógép rekurzív eljárás-hívására mutat példát.

A képernyő közepére rajzol egy növényt, melynek leghosszabb szakaszát a MERET nevű változó, elágazásainak számát a SZINT nevű változó, elágazásainak szögét pedig a SZOG nevű változó határozza meg.

Növényünk színe a SZOG-tól függ.

Fehérvári Ferenc

```

100 PROGRAM "FA"
110 DEF FA(N,SZINT,SZOG)
120 IF SZINT>0 THEN
130 PLOT FORWARD N:
140 PLOT LEFT SZOG:
150 CALL FA(N-N/4,SZINT-1,SZOG)
160 PLOT RIGHT 2*SZOG:
170 CALL FA(N-N/4,SZINT-1,SZOG)
180 PLOT LEFT SZOG:
190 PLOT BACK N
200 END IF
210 END DEF
220 GRAPHICS HIRES 4
230 SET PALETTE 0,2,3,7
240 OPTION ANGLE DEGREES
250 PLOT 600,10:
260 PLOT ANGLE 90:
270 PRINT AT 2,5:"FA meret,szint,szog :";
290 INPUT MERET,SZINT,SZOG
300 S=MOD(SZOG,4)
310 SET INX S
320 IF S=0 THEN S=9
330 CALL FA(MERET,SZINT,SZOG)
340 END
    
```


Nyirő Árpád, a Fehérvári ucai Gelka művezetője elsőként vásárolt Enterprise-t. Két gyerek aje, érhető hát, hogy eredetileg a gépet „családi gépnek” szánta. Ebből az elképzelésből annyi mindenesetre valóra vált, hogy tízenegy éves fiával szinte egymás kezéből kapkodják ki a számítógépet. A szándék és a való később kikerekedett.

M. M. Munkaköréből adódóan is jól ismeri az elterjedt kisgépeket, mint a Spectrumot, a Commodore családát, a Primót. Sőt IBM klónokon is dolgozott. Miért éppen az Enterprise mellett döntött?

Ny. Á. Elhatároztam, hogy veszek egy kisgépet a fiamnak, amit én is tudok használni a munkámhoz, amikor meghallottam a Centrum hirdetését. Úgy gondoltam, ha annak, ami elhangzott, csak a fele is igaz, akkor nekem egy ilyen gépre van szükségem. Nem tagadom, igen kellemesen csalódtam, mert az Enterprise még annál is sokkal többet tud, mint amit híreszteltek róla. C64-et azért nem vettem, mert olyan gépet akartam, amelyik játékra is alkalmas, de — mint említettem — a munkámban is hasznosíthatom. Ennek következtében tehát feltétel volt, hogy a gép olyan legyen, amely más típusú gépekkel is bizonyos szinten hardver- és szoftverkompatibilis. Igaz, az operációs rendszere nehezekebb, de igen fejlett. Főleg a CP/M alatt futtatható programok igen jók.

M. M. Én sok helyen kerestem üzletekben CP/M alatt futó programokat, de sajnos eredménytelenül.

Ny. Á. Még az első időszakban felkérték, hogy tervezek kontrollert. Amikor elkészültem, elkezdtem keresni a kollégáimmal olyan programokat, amelyek CP/M alatt futnak. Tény és való, hogy ilyen programok a kereskedelemben vagy már nincsenek, vagy még nincsenek. Az NSZK-ban például az Amstrad cégnél kaphatók Enterprise-ra adaptálható programok.

M. M. Ön elsősorban a CP/M alatt futtatható programok használatát szorgalmazná?

Ny. Á. Feltétlenül! Szerintem a Centrum elsősorban a játékgépprogramokat erőlteti. Ha más igényt is, mint például az iskolákét is figyelembe vettek volna, akkor ezek a programok létfontosságúak lennének.

M. M. Ön nagyon régóta használja az Enterprise-t. Véleménye szerint ennek a gépnek el kellene terjednie az iskolákban?

Ny. Á. Igen, nagyon sok szempont miatt. A hardverreze sokkal inkább megfelel a nemzetközi irányzatnak, mint a jelenleg elterjedt gépeké. A gépen futtatható szoftverek kiválóan hasznosíthatók a

számítástechnika oktatásában, de egyéb területeken is. Ne feledjük, a gép ára is kedvezőbb a többinél.

M. M. Megkérdeztük a TII-t az Enterprise iskolai alkalmazásáról. Véleményük szerint bevezetése megbontaná az iskolai gépek homogenitását; a másik kifogás, hogy nem magyar ábcével készül.

Ny. Á. Miért, most homogén géppark van? Igaz, hogy a TVC-vel ellentétben a

M. M. A gyerek fejlődése szempontjából lényeges volt, hogy milyen gépen tanult?

Ny. Á. Tulajdonképpen az alapok szempontjából ez érdektelen. Ha úgy dönt, hogy tovább akar számítástechnikával foglalkozni, akkor már egyáltalán nem mindegy. Ugyanis amint már mondtam, az Enterprise hardveralkitása és operációs rendszere nagyon hasonló az IBM-kompatibilis gépekéhez.

M. M. Az ön munkáját mennyire segítette az Enterprise?

Ny. Á. Nagyon sok kellemes meglepetésben volt részem vele kapcsolatban. Hamar rájöttem, hogy ez a gép több célra is

MEGKÉRDEZTÜK AZ ENTERPRISE -RŐL

billentyűzete nem magyar ékezetes, de kis átalakítással a képernyőn megjelenik a teljes magyar ábcé. Ezt nem a géptől kell megkövetelni, hanem a felhasználótól.

M. M. Mint gyakorló apa, észrevette-e, hogy a fia helyesírása romlott, amióta ezen a gépen programozik?

Ny. Á. Egyáltalán nem. Szerintem nem ezen múlik, hanem hogy mennyit olvas és ír. A gyerek nagyon jól megkülönbözteti, mikor dolgozik géppel és mikor kell magyarul írnia. Automatikusan átáll egyikről a másikra. A programnyelv magyarosítása pedig kifejezetten ártalmas.

M. M. Ha már így elkalandoztunk a pedagógiában, arra kérem, mondjon néhány szót a fia fejlődéséről.

Ny. Á. Fiamnak ez az első gépe. Eleinte nehezen ment neki a gép kezelése, programozása. Sokat kellett segítenie neki. Azután elkezdte magától kikínólni a feladatok megoldását, nekem már csak az eredményt mutatja meg. Ami a játékgépprogramokat illeti, egyre kevesebbet foglalkozik velük.

M. M. Kellett-e különösebben ösztönözní a számítógép megszerzésére, illetve hozzájárultak-e ehhez a játékok?

Ny. Á. Igen, a játék elősegítette a gép megkedvelését, de nem kellett őt különösen ösztönözni. Van azonban ellenpélda is! A lányom játszani szeret, de különösebben nem érdekli a számítógép. Az Enterprise-zal sikerült a fiamnál célt érnem, vagyis hogy megismerje a számítástechnika alapjait és a programozás logikáját. Ezt olyan gépen tanulta meg, amelynek ismerete alapján könnyebben elmélyedhet a profi szintű számítástechnikában is.

alkalmas. A Gelkánál már nyolc-tíz helyen alkalmazták munkalapok feldolgozására.

M. M. Ezt a rendszert ön vezette be?

Ny. Á. Igen. A Gelkának volt egy nagy számítógépe, amit idővel kivontak a munkából. Jelenleg a munkalapok adatait, amelyektől az emberek fizetése is függ, az egységek Enterprise gépeken vizik fel lemezekre. A lemezeket a központban dolgozók fel IBM-kompatibilis gépen. Az ilyen munkához azonban kiegészítő numerikus billentyűzet kell, amit 1989 első negyedévében már kapni lehet.

M. M. Az Enterprise teljes konfigurációval már drága, megközelíti az IBM klónok árát!

Ny. Á. Ezzel egyetértek, de az IBM-kompatibilis gépek árai nagyon ingadoznak. Enterprise telepítése a vállalatokhoz nemcsak ár kérdése, hanem az adottságtól is függ. Ahol nincs még IBM-kompatibilis gép, ott érdemes Enterprise-t venni.

M. M. Milyen hiányosságokat tud mondani az Enterprise-zal kapcsolatban?

Ny. Á. Még mindig kevés a szakirodalom, és ami van, az hibás, mint például az EXOS 2.1. Furcsállom, hogy nem jelenik meg a kiegészítés. A legkomolyabb probléma azonban a kontrollerrészlettel van. Nagyon gyakori eset ugyanis, hogy nem, illetve hibásan illeszkedik.

Pinke György

Fedezzük fel együtt!

Mielőtt valaki hamar „hibázna”

A strukturális vagy moduláris programozás olyan programozási technika, amely részfeladatok megoldására szolgáló, logikailag jól elkülöníthető programrészek – struktúrák (szerkezetek), eljárások, modulok, szubrutinok, függvények – használatán alapul.

A BASIC nyelv hibájaként leggyöszörű az rójak fel, hogy nem támogatja ezt a programozási módszert, és emiatt rossz programozási gyakorlat alakul ki azokban, akik ezen a nyelven nevelkednek. Mielőtt bárki elhamarkodottan véleményét alkotta erről, ajánlom, induljon ki abból, hogy a hiba nem a BASIC nyelvben keresendő. Meggyőződésem ugyanis:

- a programozás alapjainak elsajátítása, egy programozási nyelv és a számítógép megismerése kis, apró feladatok megoldásával kezdődik, amelyeket nem lehet, illetve nem érdemes modulokká szedni;
- bármely programozási nyelven lehet áttekinthetetlen, felesleges részleteket tartalmazó programokat írni. Ha egy programozási nyelv tanulásán, tanításán nem az utasítások alkalmazásának megismerését, elsajátítását értjük, hanem a számítógépes problémamegoldást egy programozási nyelv segítségével, akkor ez a veszély nem áll fenn. Eb-

ből következik persze az is, hogy csakis átgondolt, tudatos munkával lehet egy összetettebb feladatot hatékonyan megoldani:

– a BASIC nyelven írt programjainkban is használhatunk modulokat, azaz szubrutinokat. Ezek alkalmazása egy kicsit több figyelmet kíván, mint más programozási nyelveknél, de megoldható.

Az Enterprise BASIC nyelvében a szubrutinok és a függvények lesznek az a modulok, amelyekből felépíthetjük programjainkat.

A SZUBRUTINOK

Két összetettebb, rokon feladat megoldásával mutatjuk be a szubrutinok, illetve a függvényblokkok – eljárásfüggvények – alkalmazását, működését. A függvények más típusaival csak után foglalkozunk.

A feladat: készítsünk programot, amivel a törtek szorzását gyakorolhatjuk!

A feladat elemzése. Azt bizonyára nem kell mondani, hogy ha papíron nem tudunk összeszorozni két törtet, akkor a számítógép számára sem tudjuk megfogalmazni, hogy hogyan csinálja. Az algoritmust tehát ismernünk kell.

```

1 REM --- 21. program ---
10 TEXT
20 GOSUB 400
30 PRINT VP,VQ
40 END
400 REM --- veliszam2 ---
410 REM kiseno parameterek:
420 " B: velistien egasz <10 ivp,vq
430 RANDOMIZE
440 LET VP=VQ*(9)+1
450 LET VQ=VQ*(9)+1
460 RETURN
    
```

21. lista

```

100 REM --- 22. program ---
110 !
120 SET CHARACTER 128,0,0,0,0,255,0,0,0,0
130 TEXT
140 INPUT PROMPT "szamalo,nevezoi:MP,MO
150 INPUT PROMPT "sor,oszlop:MS,MD
160 GOSUB 500
170 END
180 !
500 REM --- tortki szubrutin ---
510 REM beseno parameterek:
520 " sor,oszlop:ms,mc
530 " szamalo,nevezoi:mp,mq
540 PRINT AT MS,MO:MP
550 PRINT AT MS+1,MO+1:CHR$(128)
560 PRINT AT MS+2,MO+2:MD
570 RETURN
    
```

22. lista

```

1 REM --- 23. program ---
20 TEXT
30 SET CHARACTER 128,0,0,0,0,255,0,0,0,0
40 LET MS=10:LET MO=20
50 GOSUB 600
60 END
70 !
600 REM --- tortbe ---
610 REM beseno parameterek:
620 REM sor,oszlop:MS,MD
630 REM kiseno parameterek:
640 REM szamalo,nevezoi:SP,BQ
650 REM LET MS+1,MO+1:CHR$(128)
610 INPUT AT MS,MD,PROMPT "":BF
620 INPUT AT MS+2,MO,PROMPT "":BD
630 RETURN
    
```

23. lista

```

1 REM --- 24. program ---
10 TEXT
20 INPUT PROMPT "ket szam:"A9,B9
30 GOSUB 700
40 PRINT "lnko=";A9
50 !
700 REM --- lnko ---
702 REM beseno parameterek:A9,B9
704 REM kiseno parameterek:A9
710 IF A9=B9 THEN 760
720 IF A9>B9 THEN 730
730 LET A9=A9-B9
740 ELSE
750 LET B9=B9-A9
760 END IF
770 GOTO 710
780 RETURN
    
```

24. lista

```

1 REM --- 25. program ---
5 REM --- tortek szorzasa ---
10 TEXT
15 SET CHARACTER 128,0,0,0,0,255,0,0,0,0
20 GOSUB 400:REM-->TORT
25 LET SP=VP:LET SQ=VQ
30 LET MP=VP:LET MQ=VQ
35 LET MS=10:LET MO=4
40 GOSUB 500:REM-->TORT KI
45 PRINT AT MS+1,MO+2: " *
50 LET MO=MO+4
55 GOSUB 400:REM-->TORT
60 LET SP=SP+VP:LET SQ=SQ+VQ
65 LET MP=VP:LET MQ=VQ
70 GOSUB 500:REM-->TORT KI
75 PRINT AT MS+1,MO+2: " *
80 LET MO=MO+5
85 GOSUB 600:REM-->OSSZEG BE
90 LET A9=BP:LET B9=BQ
95 GOSUB 700:REM-->LNKO
100 LET B9=A9
110 GOSUB 700:REM-->LNKO
115 LET SD=A9
120 !
125 REM --- valasz vizsgalata ---
130 PRINT AT 16,5:CHR$(161)
135 IF SP/SD<>BP/BD OR SQ/SD<>BQ/BD TH
EN
140 PRINT AT 16,5:"Nem sikerult! Pro
bald meg ujbol!"
145 GOTO 85
150 END IF
155 IF BD=1 THEN
160 PRINT AT 16,5:"Helyes!"
165 END IF
170 END IF
175 PRINT AT 16,5:"Egyzerusitid!"
180 PRINT AT MS+1,MO+3: " = "
185 LET MO=MO+5
190 GOTO 85
321 ! -----
322 !
325 " A 25. program eddig tart
324 " A szubrutinokat nem kell
kiirni
325 ! -----
400 REM --- veliszam2 ---
410 RANDOMIZE
420 LET VP=VQ*(9)+1
430 LET VQ=VQ*(9)+1
440 RETURN
500 REM --- tortki ---
510 SET CHARACTER 128,0,0,0,0,255,0,0,0,0
520 PRINT AT MS,MO:MP
530 PRINT AT MS+1,MO+1:CHR$(128)
540 PRINT AT MS+2,MO:MQ
550 RETURN
600 REM --- tortbe ---
610 PRINT AT MS+1,MO:CHR$(128):CHR$(12
0)
620 INPUT AT MS,MD,PROMPT "":BP
630 INPUT AT MS+2,MO,PROMPT "":BQ
640 RETURN
700 REM --- lnko ---
710 IF A9=B9 THEN 780
720 IF A9>B9 THEN 730
730 LET A9=A9-B9
740 ELSE
750 LET B9=B9-A9
760 END IF
770 GOTO 710
780 RETURN
    
```

25. lista

```

1 REM --- 26. program ---
10 REM --- tortek osszezege ---
20 TEXT
30 SET CHARACTER 128,0,0,0,0,255,0,0,0,0
40 NUMERIC P,Q,R,S,E,D
50 CALL VELISZAM2(P,Q)
60 LET SP=P:LET SQ=Q
70 LET I=10:LET J=4
80 CALL TORTKI(I,J,P,Q)
90 PRINT AT I+1,J+2: " +
100 LET J=J+4
110 CALL VELISZAM2(P,Q)
120 LET SP=SP+P:LET SQ=SQ+Q
130 CALL TORTKI(I,J,P,Q)
140 PRINT AT I+1,J+2: " = "
150 LET J=J+5:LET W=0:LET T=0
160 CALL TORTBE(I,J,R,S)
170 CALL LNKO(R,S,E)
180 CALL LNKO(SP,SQ,D)
184 !
185 REM --- valasz vizsgalata ---
190 PRINT AT 16,5:CHR$(161)
200 IF SP/SD<>P/Q OR SQ/SD<>Q/S THEN
210 PRINT AT 16,5:"Nem sikerult! Pro
bald meg ujbol!"LET W=1
220 GOTO 160
230 END IF
240 IF E=1 THEN
250 PRINT AT 16,5:"Helyes!"
260 END IF
270 PRINT AT 16,5:"Egyzerusitid!"
290 PRINT AT I+1,J+3: " = "
300 LET J=J+5:LET T=1
310 GOTO 160
398 ! -----
399 !
400 DEF VELISZAM2(REF P,REF Q)
410 RANDOMIZE
420 LET P=RND(9)+1
430 LET Q=RND(9)+1
440 END DEF
499 ! -----
500 DEF TORTKI(MS,MO,KP,KQ)
520 PRINT AT MS,MO:KP
530 PRINT AT MS+1,MO+1:CHR$(128)
540 PRINT AT MS+2,MO:KQ
550 END DEF
599 ! -----
600 DEF TORTBE(I,J,REF BP,REF BQ)
605 PRINT AT I+1,J:CHR$(128):CHR$(12
0)
610 IF W=0 OR T=0 THEN
615 INPUT AT I+2,J,PROMPT "":BQ
620 INPUT AT I,J,PROMPT "":BP
625 LET W=1
630 ELSE
635 INPUT AT I,J,PROMPT "":BP
640 INPUT AT I+2,J,PROMPT "":BQ
645 END IF
650 END DEF
699 ! -----
700 DEF LNKO(A9,B9,REF D9)
710 DO UNTIL A9=B9
720 IF A9>B9 THEN
730 LET A9=A9-B9
740 ELSE
750 LET B9=B9-A9
760 END IF
770 LOOP
780 LET D9=A9
790 END DEF
    
```

26. lista

Milyen részekből áll össze a program?

1. Véletlenszámokkal előállítunk két törtet és azokat tört alakban megjelenítjük a képernyőn.
2. Kérjük az eredményt. Ezt is tört alakban.
3. Elvégezzük az összehasonlítást. A válasz akkor hibás, ha a gép által előállított és a választék begépelte tört egyszerűsített alakban nem azonos.

Most kell eldöntenünk, hogy mit tudjon egy szubrutin, mi legyen a kezdő sorszáma és milyen változókat használjunk.

VELSZAM2 szubrutin — két 10-nél kisebb pozitív véletlenszámot állít elő

Kezdő sorszám: 400
Kimenő paraméterek: VP, VQ
Ezek azok a változók, amelyeknek a szubrutin értékeket ad, azaz ezeknek az értékei adják a két véletlenszámot.

TORTKI szubrutin — kiírja a törtet
Kezdő sorszám: 500
Bemenő paraméterek: MP — számláló, MQ — nevező, MS — sor, MO — oszlop

Ezeknek a változóknak kell értéket adnunk, mielőtt a szubrutint hívánk a **GOSUB** utasítással

TORTBE szubrutin
Kezdő sorszám: 600
Bemenő paraméterek: MS — sor, MO — oszlop
Kimenő paraméterek: BP — számláló, BQ — nevező
LKNO szubrutin — két szám legnagyobb közös osztóját határozza meg

Bemenő paraméterek: A9, B9 — a két szám
Kimenő paraméter: A9 — a két szám legnagyobb közös osztója.

A PROGRAM MEGÍRÁSA

Először megírjuk a szubrutinokat. A szubrutinok elé rövid kis főprogramot írunk, hogy kipróbálhassuk.

Nem biztos, hogy a program tervezésekor mindent figyelembe tudunk venni és mindent gondolunk. A szubrutinok megírásakor is mérünk fel újabb szempontokat, amik módosíthatják eredeti elképzelésünket. Több feltételt legtöbbször csak a szubrutin kipróbálása után adhatunk meg. Például a **TORTKI** és a **TORTBE** szubrutinokban $1 \leq MS \leq 18$; $1 \leq MO \leq 38$ lehet. A feltételeket a szubrutinban is megadhatjuk. Ez azzal az előnnyel jár, hogy a szubrutin alkalmazásakor nem kell a dokumentációk között keresgény. Miután összeállítottuk a programot és az működik, ezeket a megjegyzéseket törölhetjük, hogy a program rövidebb legyen és gyorsabban fusson.

A bemenő paramétereket a szubrutinban is megvizsgálhatjuk. Ez azzal az előnnyel jár, hogy a hibakeresés egyszerűbb lesz. Amennyiben nincs ellenőrzés, jobban kell támaszkodnunk a szubrutinok leírására.

A 21—24. listából töröljük ki a főprogramokat, és csak a szubrutinokat mentjük ki. Ezek után a **MERGE** paranccsal egymás után beolvasuk a szubrutinokat, és az így összeállított program elé beírjuk a főprogramot (25. lista). A főprogram megírásakor arra ügyeljünk, hogy a szubrutinok hívása előtt a bemenő paramétereknek értéket kell adni. A szubrutin változóit a bemenő paraméterekben nem tanácsos máshoz használni. Ezért választunk szokatlanabb neveket a szubrutin változóknak.

A FÜGGVÉNYEK

Amint azt korábban már említettük, most csak a függvényblokkokról lesz szó. A függvényblokk olyan programrészlet, amelyik a **DEF** név paraméterek utasítással kezdődik és az **END DEF** utasítással végződik. Hívása a **CALL** név paraméterek utasítással lehetséges.

A 26. listán látható program függvényeit az előző program szubrutinjaihoz készítettük. A név helyett, azaz a **REM** sorokba beírtuk a **DEF**, a **RETURN** helyére az **END DEF** utasításokat. A függvényeknél a sorszámoknak már nincs szerepük, azokat bárhol elhelyezhetjük a programban, a függvényeket nevükkel azonosítjuk. A **DEF** szó után adjuk meg a függvény nevét, amelyre a **CALL** utasítással hivatkozhatunk. A függvények kimenő paramétereit elé a **REF** szót kell írni.

A paraméterek elnevezését több helyen megváltoztattuk, hogy bemutassuk a globális, azaz a programba bárhol, és a lokális, csak a függvényben használt változók közötti különbséget.

A globális változókat a főprogramban definiáljuk a **LET**, **INPUT**, **READ**, **GET** vagy a **NUMERIC** és a **STRING** utasításokkal. A lokális változókat a **DEF** utasításban kell megadni, és azokat csak a függvényben belül használhatjuk. A lokális változókat a programban: **MS**, **MO**, **KP**, **KQ**, **BP**, **BQ**, **A9**, **B9**, **D9**.

A **CALL** utasításban fel kell sorolni a be- és kimenő paramétereit abban a sorrendben, ahogyan azok a **DEF** utasításban szerepelnek. Itt természetesen csak globális változókat használhatunk, hiszen a főprogramban adunk értékeket ezeknek a változóknak, és ott használjuk fel a függvények kimenő paramétereit is.

Lokális változót nem használhatunk ugyanabban a függvényben be- és kimenő paraméterként. Ezért kellett az **LKNO** függvényben a **D9** változót beírni. A **TORTBE** függvény is eltér a szubrutinoktól. A program futtatásakor a kiegészítés miattjére is választ kapunk.

Dusza Árpád

Mi a manó?

Újabb **Enterprise**-titok. A gép forgalmazása körüli rejtélyek újabb adalékkal gyarapodtak. A Novotrade 2C számítástechnikai áruházában az **Enterprise 128** tizenegyezer-kilencszáz forintért kapható. OTP-hitelre ugyan nem vásárolható, de a hétezer forinttal alacsonyabb ár felbolygatta a piacot. Hogyan lehetséges ez az alacsony ár és hogyan adhatnak rá jótállást, amikor a **Professional** szervizzel a **Centrum** kötött szerződést? — Ez titok. Igaz, Rényi Gábor vezérigazgató a vele korábban készített interjúban kijelentette: egy cég üzletpolitikája nem tartozik a nyilvánosságra.

kerülnek a **Zzzzip** toplistájára, amit az **Enterprise** rovat rendszeresen közöl majd.

Racsó Tamás lelkes olvasónk küldött néhány tanácsot az **Enterprise**-tulajdonosoknak. A kétnyelvű gépeken vagy egy nem ismertett **EXOS**-változó, a kódja 90h (144). Értéke 0, ha német, 255, ha angol módban van a gép.

Az **EXOS**-leírás nem ismerteti, hogyan kell saját editor-csatornát definiálni. Csatornamegnyitásként a **VID EDIT** (1Dh=29) **EXOS**-változóba egy létező videocsatorna kódját kell írni. Ennek természetesen szöveges lapnak kell lennie. Az **X** irányú méret legalább 4, az **Y** méret legalább 3 karakternyi legyen. A **KEY EDIT?** (1Eh=30) változóknak a billentyűzetcsatorna kódját kell tartalmaznia. A **BUF EDIT** (1Fh=31) változót szükség szerint be kell állítani, aminek maximális értéke 63 lehet. Ha ennél nagyobb értéket írunk bele, nem lesz hibajelzés, a rendszer csak az alsó 7 bitet veszi figyelembe (modulo 64). Ezután megnyithatjuk az editor-csatornát. Ha a csatornakód nem 0, akkor nem íródik ki promt a bemenetnél, de helyettesíthetjük az editorcsatornára író **PRINT** vagy **PRINT AT** utasítással, majd közvetlenül utána egy **INPUT**-tal. A hatás ugyanaz.

Zzzzip competition. Az **Enterprise BASIC Compiler** minden rég elfeledett, a fiókok mélyén porosodó, lassúságuk miatt mellőzött programot újjávarázsol.

Szerkesztőségünk az „a” **STUDIO**-val karöltve felajánl száz darab mintapéldányt ebből a szinte nélkülözhetetlen programcsomagból, azoknak a lelkes programalkotóknak, akik részt vesznek játékos pályázatunkon. Eszerint: készítsen **BASIC** nyelvű játék- vagy demoprogramot és ne zavarja, hogy lassú. Ahhoz, hogy minél előbb **Zzzzip**-tulajdonos lehessen, programját kezettől rövid leírással, valamint neve és pontos laccime feltüntetésével küldje el a szerkesztőségünk címére: Mikroszámítógép Magazin 1371 Budapest Pf. 433. A programokat a **Magazin** szakmai zsűrije értékeli és a legsikeresebb pályaművek fel-



Gaetsch Günter né rajza

A Scanntronik nem ült a babérjain

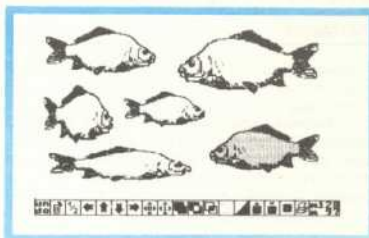
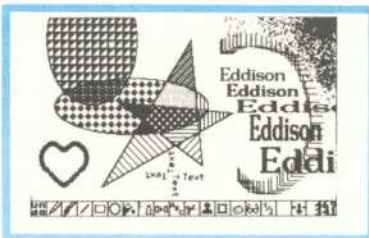
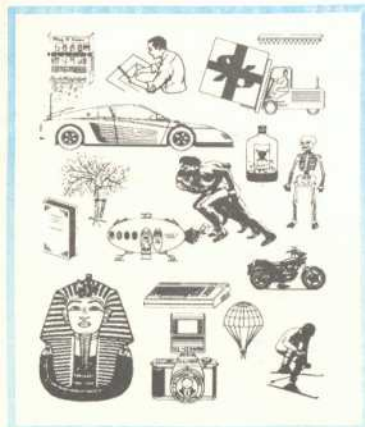
A nyugatnémet Scanntronik Mugrauer GmbH nem üldögtelt az év szoftverének kikiáltott DIP programja miatt babérjain, hanem felvette az ugyancsak NSZK-beli Markt & Technik kiadó által dobott kesztyűt. Mint Mugrauer úr egyik budapesti látogatásakor hangsúlyozta, cége a professzionális felhasználási célokra jobban megfelelő nagy felbontású, HiRes grafikát rajzoló szoftvereket fejleszt. Az M&T multicolor üzemmódban működő Amica Paintjére és a Giga Paintjére a Scanntronik válasza a karácsonyi piacra kihozott Eddison és Eddifox volt.

Az 58 márkáért árusított Eddison lényegében a sokak által jól ismert Hi-Eddi +, illetve a Printfox beépített rajzolóprogramjának továbbfejlesztett változata. A szerencsés Pagefox-modul tulajdonosok után most már másoknak is hozzáférhető lemezen a szuper DIP program rajzoló része. Sikerült a teljesen menüvezérelt programba beépíteni a fokozatmentes nagyítási és kicsinyítési lehetőséget is. (Ezt a Printfoxnál a „64-er” magazinban nyáron megjelent beírható Lupe-Fox adja.)

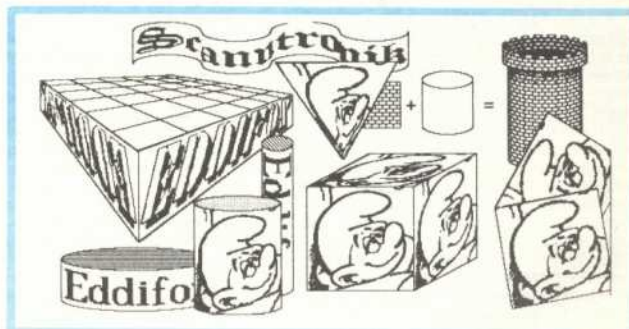
A C64 új „szeme”, a Superscanner III már a fényképeket is a képernyőre varázsolja



Részlet a legújabb grafikai gyűjteményből, ami a Printfoxhoz használható



Az Eddison menürendszere kísértetiesen hasonlít a Pagefoxéra



Az Eddifox már mindent tud, ami egy C64-ből kihozható

Az ugyancsak lemezen forgalomba hozott Eddifox 88 márkáért már csak a Pagefox-modul tulajdonosokat teszi boldoggá. Ugyanis a nagyobb számítógépeknél, mint a PC-knél, az Atari ST-nél vagy az Amiganál ismert rajzolófunkciókat, torzító lehetőségeket biztosítja a C64-nek a modul által „feltuningolt”, 100 kb-át többletmemória kihasználásával. A nagy teljesítmény és a komfortos használat mellett az Eddifox a következőket nyújtja:

— A 8 képernyőnek megfelelő A/4 oldal (640 x 800 pont) egyidejűleg ott van a grafikai memóriában. Az egész oldal egyszerre forgatható (90 fokos szögben), menthető lemeze, tölthető vissza onnan és kinyomtatható. Az A/4 oldal teljes egészében fokozatmentes kicsinyítésnek és nagyításának nem szab határt a látható képernyő.

— Az Eddifox a grafikai képek a modifikálására, illetve torzítására is teremt eszközöket. Ezekben rejlik a program különlegessége: a fokozatmentes forgatás, a térbe döntés, az elmosás, az enyészpontos térbeli ábrázolási mód, négyzög háromszögre alakítása, egy adott rajz sík, illetve hajlított felületre transzformálása (például kockára vagy hengere). Nehéz e lehetőségeket szavakkal szemléltetni, néhány példa inkább érzékelteti a mellékelt illusztrációk.

— A beépített „gyalu” és „ráspoly” a nagytáznál, a torzításoknál és a forgatásnál előforduló egyenletlenségeket csökkenti, illetve elsimítja.

— A vonalak húzását, a négyszögek, körök rajzolását nem korlátozza a látható képernyő. Tehát az A/4-es lapra teljes szélességben rajzolhatunk például egy nagy kört.

Sajnos, ez a csodaprogram csak a már említett Pagefox DIP-modullal fut együtt, viszont a kiadványszerkesztőbe a grafika elviszése nélkül átterhelhetünk, és ott tovább használhatjuk azt. Így az Eddifox a Pagefox kibővített grafikai szerkesztőjévé válhat.

További újdonsága a Scanntroniknak a „Tippek és trükkök a Pagefoxhoz” című, 150 oldalas, 78 márkáért kapható könyve, amelyhez mellékelt lemez nemcsak oldalterveket, hanem egy grafikai könyvtárat is tartalmaz. Magából a könyvből nemcsak a profik, de a kezdő kiadványtervezők is jó tanácsokat kaphatnak a tipográfia, az olvashatóság, az esztétikus felületfelosztás, a layout-készítés, az aranyretetszés csinájához-binjához. Ha a Printfox-tulajdonosoknak csak a digitalizált rajzokra van szükségük, akkor megvehetik 38 márkáért a lemezt a könyv nélkül „Grafikai gyűjtemény a Printfoxhoz” címmel. (Ez nem azonos a 78 márkás ismertebb Printfox — Basarral!)

Továbbfejlesztették a képletapogató (digitálizáló) berendezésüket is: a Superscanner III — igaz, elég borsos áron, 398 márkáért — már nem csak vonalas rajzokat, de öt szerkesztési fokozatban árnyalatos fotókat is képes a képernyőre varázsolni. A Pagefox-modul (ha van) itt is sokat segíthet azzal, hogy vele egész A/4-es oldal feldolgozása lehetséges. A Superscanner III nagyobb teljesítményét bizonyítja az is, hogy 30 százaléktól 300 százalékgig fokozat nélküli zoomlehetősége is van az x és az y értéke elkülönítetten. Ezzel az utólagos nagyítással és kicsinyítéssel jelentkező minőségromlás kiküszöbölhető.

Huppán Béla

Hardver

Perifériák

A következőkben megpróbálunk összefoglalni néhány — elsődlegesen kis mikroprocesszoros rendszerben alkalmazott perifériával kapcsolatos — legfontosabb áramkört ismereteket. Ezeknek a perifériáknak az áramköri kialakítása viszonylag egyszerű. Kellőképpen működtető-kezelő programjukkal, elterjedt kifejezéssel: henderlőjűkkel működtethetők.

Billentyűzetek

A számítógép—ember kapcsolat klaszikus és talán ma is legfontosabb eszköze a billentyűzet. A legegyszerűbb megjelenési formája egy kétállású kapcsoló vagy nyomógomb, ami a számítógép egyik bemeneti kapujára csatlakozik. Az érintkezők azonban több záródás és szétnyílás — visszapattanás — után záródnak. Úgy mondják, hogy az érintkezőknek pergése (prellje) van. Ez azért okozhat problémát, mert a mikroprocesszoros rendszer lekérdezési sebessége olyan nagy, hogy az esetleges, kézzel, egyszerű billentyűnyomást a pergés miatt gombnyomássorozatnak képes észlelni. Ezért bizonyos esetekben az érintkezőket pergesmentesíteni kell, ami megvalósítható mind hardverrel, mind programmal. A programmal való pergesmentesítésnél az első záródás észlelése után bizonyos idő múlva (a gyakorlatban 10-20 ms) ismét beolvassuk az észlelt érintkező állapotát, ami már a stabil állapotot adja.

A pergesmentesítés egyik áramkörti megoldása, hogy az első jelszintváltás után RC taggal késleltetjük a kimeneti jelszint változását. Az 1. ábrán foglaltuk össze a szokásos megoldásokat.

A prellmentesítésnek az a legmegnyugtatóbb megoldása, ha nem mechanikus, hanem prellmentes, elektronikus, például Hall-elemes kapcsolót alkalmazunk. Ennél a megoldásnál a téglalap alakú, lapos kivételeket tartalmazó Hall-elem a hozzá közel kerülő kis mágneslap hatására megváltoztatja kimeneti logikai jelszintjét, amit a bemeneti kapura kötünk. A nyomógombba beépített kis mágneset a gomb

A sorozat alapgondolata — azon a régi felismerésen túl, hogy az elektronika és a számítástechnika elválaszthatatlan egymástól — a következő tapasztalatot summázza. A szoftver — a programok — jelentősége egyre nő, de az is tény, hogy az igazán jó (az adott számítógép nyújtotta lehetőségeket maximálisan kihasználó) programok megírásához a

megnyomásával juttatjuk a Hall-elem közelébe.

Amennyiben több billentyűt akarunk alkalmazni, az 1. ábrán bemutatott megoldás többszörösén kívül egy másikat, gazdaságosabbat is választathatunk. A billentyűzet ilyenkor áramkörileg egy mátrixalakra felbontott érintkezőrendszer, ahol az egyes érintkezők egyik pontja a mátrix adott sorára, a másik pontja a mátrix adott oszlopára kapcsolódik. A sor-, illetve oszlopkivezetésekhez kapcsolódó elektronika megállapítja, hogy melyik sor és oszlopkivezetés között van kontaktus, amelyhez tartozó sor-oszlop metszéspontban lévő billentyűt nyomtuk meg. A mátrixtasztatúra elektronikájában általában a soros lekérdezés elvét alkalmazzák. Ennek lényege, hogy az egyik vonal-sorozatra, például a függőlegesre sorban, egymás után logikai L impulzusokat adunk, és „figyeljük”, hogy melyik vízszintes soron jelenik meg az L szint. Ebből egyértelműen kiderül, hogy melyik billentyűt nyomtuk meg. Egy lehetséges, 16, mátrixalakra elrendezett billentyűzetkezelő elektronika látható a 2. ábrán.

Az elektronika a bemeneteket kapcsolt 3–10 kHz-es órajellel dolgozik. A 74LS76-os IC mint kétbites számláló működik, és a 15. és 11. kimenete folyamatosan címezi a 74LS139-es demultiplexert. Ennek Y0...Y3 kimenetei a számláló kimeneteitől függően folyamatosan, egymás után L szintre kerülnek mindaddig, amíg nyomógomb nincs lenyomva. A billentyűzetmátrix sorai ennek folytán sorban egymás után L szintre kerülnek. Ha egy nyomógombot megnyomunk, akkor a gyors letapogatás miatt a demultiplexer megfelelő kimenetén megjelenő L szintet a lenyomott billentyű a D0 és D1 jeleket előállító kapukra kapcsolja. A lenyomott billentyű okozta L szintet a C és D jelű kapukra jutva, a közös kimenetüket L szintre húzza, letiltja a kétbites számlálót. Ilyen módon a leállított számláló pillanatbeli áll-

programozónak rendelkeznie kell alapfokú áramköri hardverismerettel is. Megerősíti ezt, hogy szaporodik az olyan berendezések, mikroprocesszort alkalmazó rendszerek száma, amelyek programvezérelten működnek. Az ilyen rendszerek tervezőinek és fejlesztőinek is szükségük van integrált hardver- és szoftverismeretekre.

lapota adja a D2 és D3 bitek értékét. A számlálót leállító jel egyúttal a billentyűnyomást jelző STROBE jel is. Ekképpen a lenyomott billentyű négybites kódjához jutunk, és ez mindössze négy bemeneten olvasható be az eredeti 16 helyett.

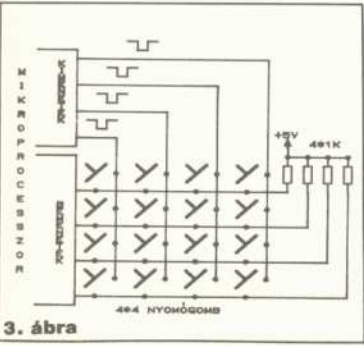
Ilyen fajta — de nagyobb billentyűmátrixot lekérdező — áramkör van például a kalkulátorokban és az egyéb, tasztatúrát használó LSI áramkörökben. Egy mikroprocesszoros rendszerben néhány be-kimeneti vonal felhasználásával maga a processzor is elvégezheti a billentyűzet kezelését, ahogy ezt a 3. ábra mutatja.

Természetesen ilyenkor a processzor idejének nagy részét a billentyűzet letapogatásával tölti. Ez a megoldás csak akkor használható eredményesen, ha a feladat olyan, hogy egy billentyűnyomást mindig a hozzárendelt feladat végrehajtása követ, majd utána ismét lehet a billentyűnyomásokat figyelni.

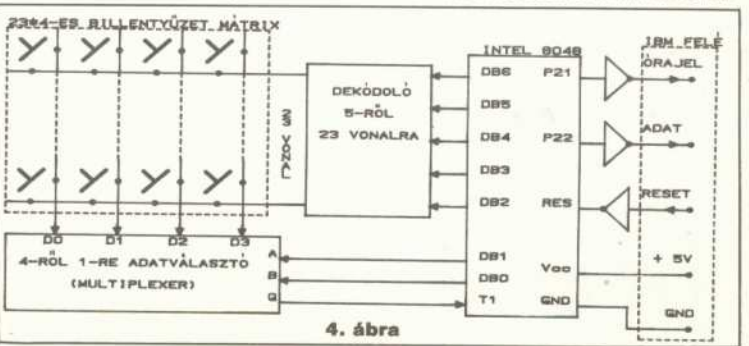
A modern billentyűzetekben — amelyek önállóan, a számítógéphez külön vezetékekkel kapcsolódva jelennek meg — már önálló, egytökos mikroszámlítógép végzi el az összes funkciót: a billentyűzet letapogatását, a billentyű megnyomásakor és elengedésekor önálló kód generálását és a kódok átvitelét a számítógép felé. A 4. ábrán az IBM PC/XT billentyűzetben használt áramkört mutatjuk be változatlan.

A billentyűzet speciális, „intelligens”, amit a beépített MCS—8048-as maszk-programozott mikroszámlítógép tesz lehetővé. A billentyűzetet alkotó kapcsolók egy 23×4-es mátrixba vannak rendezve. A számítógép felé a billentyűzetről az adatok sorosan érkeznek, és az itt nem részletezett soros-párhuzamos átalakító áramkörön keresztül jutnak egy bemeneti pontra. A mikroszámlítógép feladatai a következők.

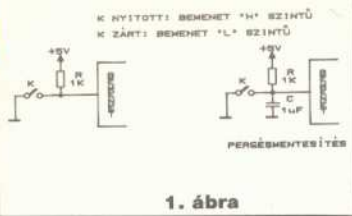
1. A 8048 öt kimeneti biteje egy dekódoló áramkörön keresztül a mátrix 23 vonala



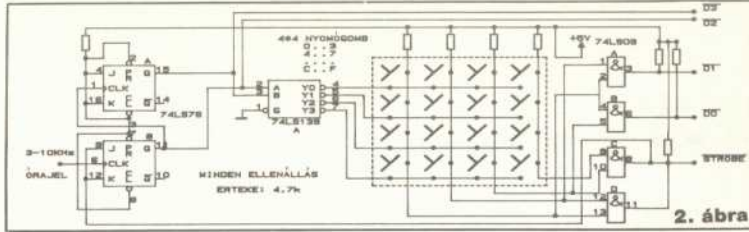
3. ábra



4. ábra



1. ábra



2. ábra

közül egyet mindig a nullára húz le. A mátrix négy kimeneti vonalat egy multiplexer megcímzésével sorba letapogat, és ha T1 kivezetés alacsony szintű, akkor a mátrixpontban lévő billentyű(k) le van(nak) nyomva. A 8048 folyamatosan pásztázza a mátrixpontokat, és ha zárt állapotú talál, vár néhány milliszekundumot (prellmentés).

2. A lenyomott billentyűhöz tartozó kódot a memóriájában tárolja (make code). A billentyű elengedésekor is egy külön kódot generál (break code), amit szintén tárol.

3. A tárolt kódot sorossá alakítva kiküldi a soros vonalra.

4. Ha egy billentyű huzamosan, több mint fél másodpercig le van nyomva, a kódot automatikusan ismétli (auto repeat funkció).

A billentyűzet kialakítása lehetővé teszi, hogy 18 billentyűnyomást/elengedést tároljunk, amit időben eltolva, egymás után küld el a soros vonalra a számítógéphez.

sen használhatók. Például amelyek extrém kis fogyasztásúak, külső fénynél olvashatók — folyadékkristályos vagy LCD kijelzők —, vagy amelyek nagyméretűek lehetnek (mágneses elvű, sokelemes megjelenítők). Sok helyen a számok megjelenítésén kívül alfanumerikus karakterek, ábrák megjelenítésére is szükség van. Erre a célra nagy választékban készülnek eszközök: az egyszerű, LED-eket tartalmazó mátrixkijelzőktől kezdve a gáziklésű, illetve az LCD-s kijelzőkön át a ma még legjobban elterjedt, kitűnő megjelenítésű katódugárcsővegek.

A legegyszerűbb esetekben a számok kijelzésére a hétszempenes LED kijelzőt használjuk, mert ehhez kell a lehető legkevesebb képelem, bár esztétikuma vitatható.

A szokásos kialakítást a szegmensek betűjelével a 6. ábra mutatja. A táblázatból megállapítható, hogy egy-egy szegmensre melyik szám kialakításához van szükség.

Egy-egy szegmensben legtöbbször két, vonal formájúra kialakított LED van sorba kötve. A hét szegmensben kívül a kijelzőben általában tizedespont is van, amit egy pont formájú LED reprezentál. A hét szegmens és a tizedespont diódáinak elektródáit nem vezetik ki egyenként: vagy az anódjaikat, vagy a katódjaikat a kijelzőtökon belül összekötik, és ezt egy kivezetésként hozzák ki (közös katódú, illetve közös anódú kialakítás). Mivel a számokat a számítógép BCD kódban ábrázolja, ezért az ilyen LED kijelzők olyan meghajtott áramkörökkel csatlakoznak a mikroszámítógépek kimeneti kapura, amelyek bemenetükön a szám négy bites BCD kódját fogadják, és a kimenetükön az adott szám hétszempenes megjelenítéséhez szükséges jelszintek állnak elő.

Kazettasmagnetofon-illesztők

A mikroszámítógépeknél a programok és adatok tárolására — mint olcsó és általában meglévő eszköz — gyakran használunk kazettás magnetofonokat. A magnetofon és a mikroszámítógép között a soros adatátvitelt valósítunk meg, és az információt reprezentáló L és H szintű bit-sorozatát hangfrekvenciás jellel — például 1200 Hz-es és 2400 Hz-es frekvenciájúvá — alakítjuk. Azt az áramkört, amelyek ezt elvégzi, modulátornak nevezzük. Ezeket a jeleket már a magnetofon képes rögzíteni. A visszaalakítás az előbbi folyamat fordítottja, amit a demodulátor elnevezésű áramkör lát el. A 7. ábra ennek a folyamatnak a blokkvázlatát mutatja.

Egyszerűbb kialakítás is alkalmas a jelátvitel megvalósítására, de a digitális jeleket, azaz a váltakozó L és H szinteket tartalmazó négyszögjelsorozatokat a magnetofon számára egy kissé át kell alakítani, a visszaolvasott jeleket pedig négyszögösíteni kell. Egy nagyon egyszerű megoldást mutat a 8. ábra.

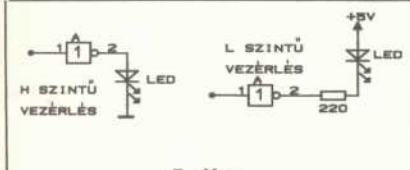
A soros adatkimenet négyzet alakú jele az R1—R2—C2 elemeken keresztüljutva lassúbb fel- és lefutást lesz. A C1 kondenzátoron keresztül csak a jel váltakozó része megy át, ami a magnetofon bemenetére jut és a szalagon rögzítődik. A visszajátszóskor a magnetofon kimenetén megjelenő jel a tranzisztort kapcsolgatja, és a számítógép bemenetén négyszögjel jelenik meg: a T1 vezet-bemeneten L szint, a T1 nem vezet-bemeneten H szint. A magnetofon illesztésének igen lényeges része az a program, amely a számítógépben 8 bites bajtok formájában a kifelé történő sorosítást és befelé történő párhuzamosítást végzi. Ilyen jellegű feladatot elvégző programot a soros adatátvitellel foglalkozó részben már bemutatunk.

Dr. Kónya László

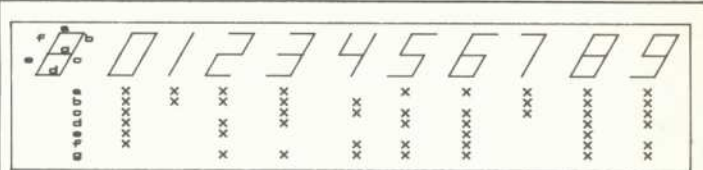
Megjelenítők

Az adatok, eredmények vizuális megjelenítésére igen sokféle eszközt használnak a mikroszámítógépes rendszerekben. Kétállapotú jelek világító LED diódákkal jeleníthetők meg. Ezek a diódák a rajtuk átfolyó 5...20 mA-es áram hatására, a típusától függően vörös, zöld vagy sárga fényrel világítanak. Ezért akár közvetlenül a mikroszámítógép kimeneti kapujára köthetők, ahogy ez az 5. ábrán látható.

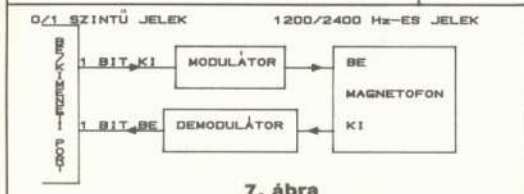
Számok, szimbólumok megjelenítésére is ma a rendkívül igénytelen és üzembiztos szilárdtest-kijelzők, a LED-ek terjedtek el, valamint azok a típusok, amelyek egy adott helyen adott feladatra előnyö-



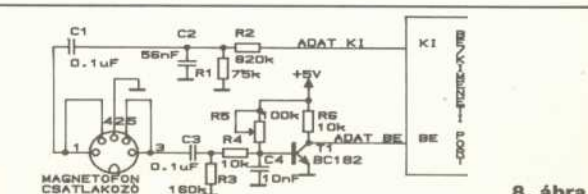
5. ábra



6. ábra



7. ábra



8. ábra

BÖRZE.



COBRA
ELEKTRONIKAI ÉS SZOLGÁLTATÓ KISSZÖVETKEZET
1097 Budapest, IX. Ilatos út 7. Telefon: 476-160/388

SZÁMLAKÉSZÍTÉSTŐL A KÖNYVELÉSIG

- COBRA—CONTO ügyviteli programrendszer moduljai
- Számlakészítő, Számlanyilvántartó, Bér- és jövedelemszámfejtő, Főkönyvi könyvelő, Anyagnyilvántartó, Általános könyvelési

PROGRAMOK!
Árak: 19 000—49 000 Ft-ig
KÉRJEN RÉSZLETES TÁJÉKOZTATÓT!

TUTTI

ELECTROCOOP
KISSZÖVETKEZET

- IBM PC kompatibilis gépek
- HARDVERTELEPÍTÉS
- SZERVIZ ÉS GARANCIA
- SZOFTVERES TÁMOGATÁS
- RÖVID HATÁRIDŐ

Cím: Bp., Üllői út 81. 1091
Tel.: 334-354



ENET

Lokális hálózat
IBM kompatibilis gépekre
Külön hardver
nem szükséges
Ára gépenként:
19 900 Ft + áfa
ECOSOFT KISSZÖVETKEZET
Tel.: 863-677



IRODA:
VI., Nagymező u. 51.
TEL.: 325-768

ADATÁTVITELI
RENDSZEREK
HARDVER/SOFTVER
GRAFIKUS MUNKAHELY
ADATRÖGZÍTÉS
SZERVIZ

ASY

Telefon:
415-166

Kereskedelmi és Szoftveriroda
1061 Bp., Liszt F. tér 10.
Telex: 22-4378

- **ASY-16 szupermikro számítógép**
 - 12 terminál
 - VME busz
 - UNIX

— **CRT TERMINÁLOK**
VT—52, QUT—102, Siemens
8160

- **BILLENTYŰZETEK**
- **MONITOROK**
- **INTEGRÁLT VÁLLALATI INFORMÁCIÓS RENDSZER**

UNIX környezetben üzemeltethető.



PERIFÉRIA
Elektronikai
Fejlesztő
és Szolgáltató
Kisszövetkezet
Bp. VII., Peterdy u. 30.
Telefon: 213-588

AJÁNLATA:

- P—XT: 140 E Ft-tól + áfa
 - P—AT: 200 E Ft-tól + áfa
- igény szerinti konfigurációk
- FX—1000 Printer: 75 E Ft + áfa
 - 40 MB-os WINCHESTER: 86 E Ft + áfa

procontrol



Kisszövetkezet
megnyitotta

COMPUTER SZAKÜZLETÉT

- hardver, szoftver
- elektronikai elemek
- integrált áramkörök
- számítógépek, perifériák
- blokkolóórák
- vonalkódoszközök, biztonsági rendszerek

SZEGED
Kazinczy u. 8. Tel.: 62/12-259
Bersenyi u. 2. Tx.: 82-726

2,2 m-es parabola-
antenna

TEL.: 323-332
TT—2164

FALIVEZETÉK-KUTATÓ

TEL.: 314-575

ERIKA
ÍRÓGÉP

Bp. VI.,

Népköztársaság útja 2.



1146 Bp., AJTÓSI DÜRER
SOR 10.
Levél cím:
1393 Pf.: 319.
Telefon: 421-974
Telex: 22-6544

március havi kínálata

a 80 386-os
mikroprocesszorra
alapozott, multiuser
üzemmódú

ACER SYS 32

rendszer, amely
széleskörűen,
változtatható kiépítésben,
alap és alkalmazói
szoftverekkel együtt
kapható.

Sorozatunkban azokat az új hardver- és szoftvertermékeket ismertetjük, amelyek várhatóan általánosan elterjednek, és meghatározó szerepük lesz a fejlődés irányainak kialakításában.

Merre tart a világ?

Hogyan tovább?

A COMDEX/FALL '88 megnyitó előadását a számítástechnikai ipar hat élen álló cégének egy-egy vezető szakembere tartotta arról, hogy mi várható a 90-es években.

Rod Canion, a Compaq Computer Corp. elnöke javasolta, hogy szabványosítsák az IC-eket, a buszokat és az operációs rendszereket. Szerinte ennek az lenne az eredménye, hogy „a számítástechnikai ipar olyan gyorsan nőne, amilyen gyorsan csak képességeiből telik, és ez a növekedés a jövőben is folytatódna”. Ennek érdekében a Compaq száz céggel együtt a COMDEX alatt javaslatot tett az ún. EISA szabvány elfogadására. Úgy vélték, ha ezt elfogadják, akkor a „vásárlók képei lesznek előre látni azt, hogy mi várható, és azt is, hogy beruházásaikkal mi történik”. Ezt feltehetően azért tartották megemlítenedőnek, mert jelenleg, ha valaki valamilyen géptípust, operációs rendszert választ, nem tudhatja, mikor tér át az ipar valami egészen másra. Ez pedig azt eredményezheti, hogy vehet új gépet, szoftvert, és kilóldhat azzal, hogyan tudja az újra átvinni eredményeit, adatait.

Bill Gates, a Microsoft Corp. elnöke a jövő szempontjából az ún. grafikus illesztőt — erről a cég egyik vezető szakemberével készített interjúban már írtunk — és a hálózatosítás általános elterjedését tartotta meghatározónak. Úgy vélte, hogy három éven belül minden DOS gép, Macintosh típusú gép grafikus illesztőt fog használni. A hálózatosítás következő lépése pedig a szabványosítás.

Philippe Kahn, a Borland International elnöke az alkalmazási szoftvergyártásban az általánosítás egy magasabb szintjét ígerte. Azt mondta: „Úgy érzem, a mi termékeink ezen az úton haladnak.”

Terry Lautenbach, az IBM ügyvezető alelnöke szerint az elmúlt tíz év rohamos fejlődése folytatódni fog, mert az ipar „rendelkezik a legkreatívabb elmékkel és a legjobb üzletemberekkel”.

Robert Kavner, az AT and T Data Systems elnöke a vásárlók igényeinek kielégítésével kapcsolatban azt mondta, hogy ha azokat teljesíte-

nék, akkor az olyan lenne, „mintha egy új autót vásárlásakor egy slussz-kulcsot és egy 500 oldalas dokumentációt várna a vevő”. Az egyszerűsítés pedig véleménye szerint ott tart, „mintha az autók zömét kormánykerékkel szállítanák, de néme-lyiket botkormányral, sőt egérral”. (Az utolsónak felszólaló elnök, egy nálunk nem ismert terjesztő hálózat vezetője, saját problémáik megoldásáról szólt.)

És minélünk? Nincs tudomásom arról, hogy mi az EISA—MCA szabványháborúban valamelyik oldalon állnánk. A Műszertechnika Szervezetet mindenestre az utóbbihoz — az IBM által ajánlott szabványhoz — készít kártyákat. A hálózatosításban — ahogy eddig a számítástechnika minden területén — haladunk affelé, hogy nálunk a típusválasztékból minden megtalálható legyen, ami csak létezik.

Külön kiemelendők tartom a szakemberhelyzetet. Nálunk az értelmiség — különösképpen a műszaki és természettudományi — szomorú helyzete miatt nemcsak az nem igaz, hogy „a legkreatívabb elmékkel rendelkezünk”, hanem, ha az utánpótlást nézem — márpedig a jövő szempontjából csak azt nézhetem —, lassan az is kétséges lesz, hogy a kreatív elmék közül bárki is a számítástechnikai iparba törekszik. Ennek pedig csak az lehet a következménye, hogy az elmaradásunk katasztrófálissá válik.

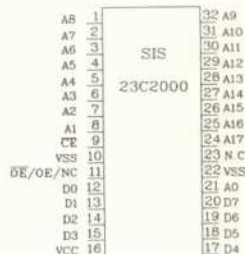
Integrált áramkörök

Az elmúlt egy év alatt a hardver legfontosabb termékcsoportjait áttekintettük. Ezt most kisebb ciklusidővel újra kezdjük. (Megjegyzem, hogy egy-egy érdekesebb újdonsággal igyekszem a kört kiegészíteni.)

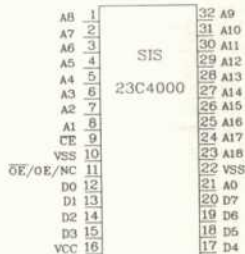
Először, kiterjesztve az első témát, az integrált áramkörök újdonságaival foglalkozom.

CMOS ROM-ok

Kétirányú fejlődés tapasztalható, mindkettő az eddigi irány folytatá-



1. ábra



2. ábra

3. ábra



RGB analóg kimenete van. A mikroprocesszorok párhuzamos csatornájáról közvetlenül vezérelhető.

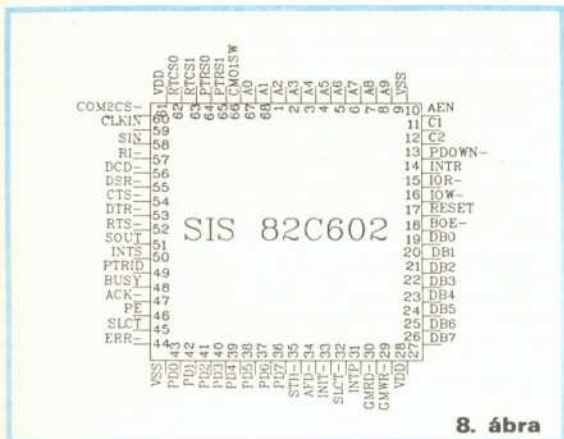
Perifériakezelők

Ezek mindegyike az AT típusokhoz készült. Az első (7. ábra), a 82C451, ellátja a nyomtatónak, a párhuzamos és soros csatornáknak, valamint a modemnek a kezelését. A 82C602-nek (8. ábra) ezenfelül egy valósidejű órafunkciója is van. A 83747 (9. ábra) a közismert 8250 típusok megkettőzése.

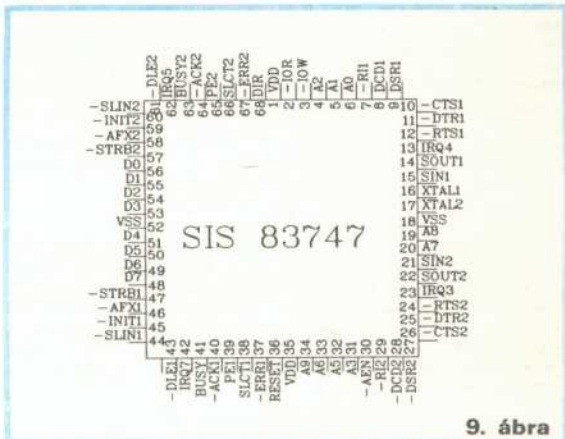
A perifériakezelők gyártásában az általános tendencia, hogy a különböző cégek más-más funkciókkal igyekeznek kiegészíteni a megszokottakat. Ekként a felhasználók sok speciális alkalmazáshoz kész



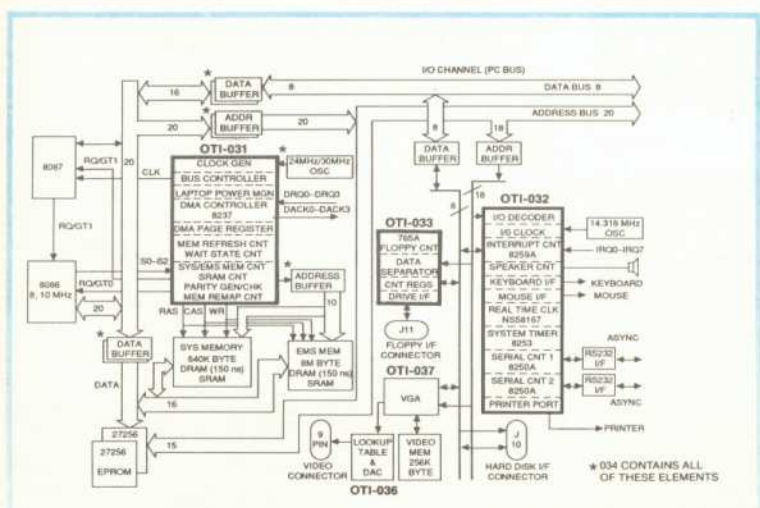
7. ábra



8. ábra



9. ábra



10. ábra

megoldást kaphatnak, ha ismerik az igen széles választékot.

A 10. ábrán egy, az OAK TECHNOLOGY által készített, PS/2 Model 30 kompatibilis kártya kapcsolási rajza látható. Az OTI feliratú IC-eket gyártja a cég. Az egyes alkatrészek blokkjába beleírták az általa megvalósított funkciókat. A cég szerint ezeknek az alkalmazásával az IBM PS/2 Model 30 alkatrészeinek több mint a fele helyettesíthető.

Simonyi

MÁRCIUS 5-én este 6-kor
jótékony célú
DIXIELAND-TALÁLKOZÓ
a Semmelweis
Orvostudományi Egyetemen
Bp. VIII., Nagyvárud tér 4.
**BENKŐ
OLDSMOBILE
LUCKY BOYS**
Jegyek a helyszínen kaphatók
Kedden 11⁰⁰ – 19 óráig és az előadás előtt.
Egyetemisták és Főiskolások
Rákellenes Egyesülete

SYSTEMEK '88

– München

A két évvel ezelőtti első Systek kiállításon még a számítógéppel segített technológiák álltak a középpontban. Az októberben Münchenben megrendezett II. Nemzetközi CIM kiállítás már a gyakorlatban bevált CIM-technológiák seregszemléje volt. A tizenhat csarnokban a 140 kiállító bemutatott termékei nemcsak viselték a CIM feliratot, de a számítógépek és problémamegoldások mellé helyezett demonstrációk — megmunkáló központok, robotok — működésével azt is szemléltették, hogy — képtelenesen szólván — leomlóban van a tiszta iroda és a piszkos termelőüzem közötti fal is. Érthető, hogy nagy volt az érdeklődés.

A vásár irányadó információközponttá vált a közepes és nagyvállalatok számára a CIM megvalósítására tett erőfeszítések koordinálásához.

Fő tendenciák

Az árcsökkenés folytatódik

A hardvergyártók közötti versenyben a legnagyobb profitot — „hagyományosan” — most a sebességnöveléssel lehet elérni. Miközben ez a törekvés érvényesül, a 2-3 éves technológiák már elavulóban vannak, és az ilyen gépek mintegy évi 20-30 százalékos arányban nyomják lefelé az árakat. A PC és a workstation — munkaállomás — eddigi megkülönböztető jegyei a teljesítménynövekedés és az árcsökkenés ollójában eltűnőben vannak. Helyenként zavaró volt a vásáron, hogy fontos termékek lista szerinti árát nem közölték. Ilyenkor — például az IBM CADAM-nál — felmerülhet a vásárló gyanúja, hogy az árkalkuláció nem a szoftvercsomag összeállításával van kapcsolatban.

Előretört a UNIX

A mintegy ötven „nagy” mérnöki programcsomag közül már 55 százalék UNIX-környezetben fut, a hagyományosan vezető VAX/VMS operációs rendszer — bár a tervező programcsomagok száma ott a legmagasabb — a második helyre szorult, 34 százalékkal. Ez a tendencia arra utal, hogy a CAD/CAM-szoftverházak a közös környezetre a UNIX-ot tekintik. Az ANVIL—5000 például fut UNIX alatt Sun és Apollo, VMS VAX, és AIX alatt is IBM munkaállomásokon.

Grafikus szuperminió számítógépek

Az olcsó szuperminió számítógépek fejlesztésére fordított amerikai erőfeszítések első eredményei — az ALLIANT, CONVEX — megjelentek a piacon. A skalár- és vektormódban dolgozó, 4—64 párhuzamosan működő processzorokkal, 40—200 MIPS sebességgel és a UNIX alatt futó hatalmas szoftverválasztékkal, valamint elképesztően alacsony, 80—200 000 USD árukkal új minőség teremtettek.

Érett hálózati megoldások

Gátszakadászerűen fejlődött be a szigetként működő különböző számítógépek és más CIM eszközök integrálása a hálózatba kapcsolással. Az érett LAN-megoldások után a Wide Area Network (WAN) technológia is a gyors fejlesztés szakaszába ért. Demonstrációk mutatták be, hogy a Münchenben elvégzett rajzmódosításokat a wolsburgi gyárban működő megmunkáló központ azonnal végrehajtja a munkadarabon. Önálló, PC/AT-n működő CAD/CAM-megoldást nem láttunk.

Az élvonalbeli gyártók eredményei

Egyértelmű, hogy a CIM akkora piaccá válhat a következő évben, mint ma a 27 egész számítógépipar. Ezért nem meglepő, hogy a legnagyobb cégek óriási versenyben mutatják be elért eredményeiket. A legfontosabbaknál a következőket láttuk:

A DEC partnervállalatokkal közösen, a már bevált VAX hardverbázison mutatta be új eredményeit, melyeket a saját szoftvereszközeinek és alkalmazásainak integrálására hoztak létre. Egy integrált folyamat a következő elemekből állt:

- A VAX-Profi-ra épülő kereskedelmi rendszer fogadja a megrendeléseket.

- A MEDUSA tervezőrendszerrel kommunikálva kereshető ki a termék, vagy ha szükséges, vele tervezhető meg az új elemek.

- A MEDUSA-NC állítja elő az NC programot, amit „az üzemi igénye szerint” az EXAPT-NC vezérléshez generáltak.

- Az AZR nevű termelésirányítási programcsomag fogadja a heti tervezésű termelési tervet, amit napi melynél bont le.

- A szerszámok, készülékek előkészítésében és összegyűjtésében a TOMS nevű programcsomag vezérli a raktári tevékenységeket.

- Ezt követően a ROYAL/VARISSET adja át a szerszámbeállítás adatokat a gépkezelőnek, aki a megmunkáló központra terminalon keresztül kapja az információkat, a palétán odaszállított munkadarabok megmunkálásához. A gépkezelő a DNC vezérlőből hívja le a megfelelő NC programot.

- A megmunkálás után az AZR-en keresztül jelenti a gépkezelő a munka befejezését, és kéri a következő feladatot.

Mindezek mögött Ethernet hálózat és a MAP protokoll következések alkalmazása áll.

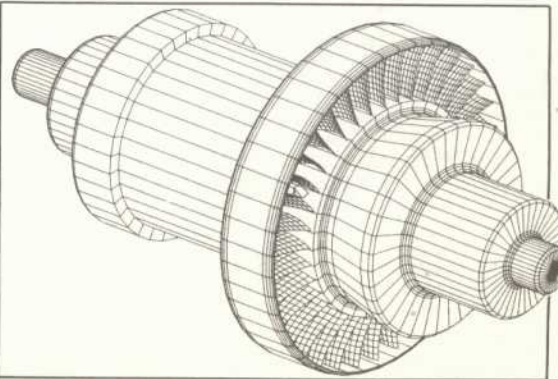
A DEC az ULTRIX—32 3. bemutatásával kihangsúlyozta elkötelezettségét a UNIX mellett. Ez a verzió teljesen kompatibilis a POSIX, a 0. szintű OSF, a Berkley 4. 3, az X—Window és a FIPS — gyakorlatilag az összes fontos — szabvánnyal, és sok új, a heterogén hálózatokat is támogató eszközt integrál.

A PRIME + COMPUTERVERSION a két évvel ezelőtti egybeolvadása óta, a még szeleesebb termék- és alkalmazási spektrumban elért eredményeit demonstrálta különböző — PRIME, SUN, DEC, Silicon Graphics — gépeken. Különös hangsúlyt kapott a munkaállomás-konceptió, és az integrálhatóságon kívül a szabványok bemutatása is. Standjukon a következőket láttuk:

- 1988 végére egységes CAD-szoftvercsomag fejlesztést jelentettek be, amely a CIS 7. 0 és a PRIME 5. 1 verziójú MEDUSA legjobb tulajdonságait egyesíti. A rendszer VAX, SUN, PRIME gépeken futhat.

- Acélszerkezetek tervezéséhez kibővítették a CADD5 programcsomagot egy PKS nevű, hídserkezetek tervezésére optimalizált felhasználói interfésszel.

- Az NSZK-ban már ezerkét százzal installált CADD5 4 x rendszer új, 4. verzióját mutatták be, amely már a NURBS (Non Uniform Rational B—Splines) nevű matematikai módszeren alapuló felülettervező funkciókat is tartalmazza.



A McDonnell-Douglas a UNIGRAPHICS — II CAD/CAM rendszer új, ergonomiailag fejlesztett verzióját DEC/VMS gépeken mutatta be. E csúcstermék integrátságát az alábbi ábra szemlélteti:

GRIP	Felhasználói programok	Felhasználói funkciók
3D Adatbázis		
UNISOLIDS 30 modell	20/30 Rajzoló-modul	Szerelés GSM NC-tervező
		GFEM véges-elemes analízis
		Szerszám-tervezés
		Poszt processzor
		Mechanizmusok
Robot-szimuláció	Rajzológép	Alkat-rész-beépítési lista
		Szer-szám-gép-vezérlés
		Kinematika

A **Hewlett—Packard** bemutatójának középpontjában a gépesített tervezőeszközök (CAD), a minőségbiztosítás (CAQ), és a gyártás (CAM) HP-Advance-Net hálózaton keresztül integrálása állt.

A már ismert HP 9000-es UNIX munkaállomáson futó ME10 tervezőrendszer — paraméteres variánsképző modulál kiegészítve —, megjelent, a 80386-os processzorra épülő VECTRA-nál nagyobb gépen, MS—DOS alatt is. Újdonság az SQL-bázisú, e rendszerhez integrált adatkezelő rendszer, mellyel a fejlesztési környezetben szokásos műszaki leírások, dokumentációk kezelhetők. Mivel a Hewlett—Packard német nyelvterületen régóta vezető pozícióban a CAD rendszereivel, a bevált gyakorlati példákat bemutató társvállalatok eredményei meggyőzőek voltak a nagyszámú érdeklődő számára.

Az **IBM** standja közepes vállalatokra specializált anyagot mutatott be. Részleteket átfogó megoldásként látható volt a professzionális CADAM új NC modulál, a CAEDS és CATIA az IBM 6150 munkaállomáson, és a PS/2, AS/400-as néhány rendszerrel. A CADAM sikerét jelzi, hogy 1986 óta 2600, MICRO—CADAM verzióját 10 000-nél több helyen adták el.

Az **Apollo** a DN3000 sorozatú munkaállomását (68020-as processzor, 68881-es társprocesszor, 4 Mbájt memória, 1,5 MIPS sebesség) fix lemez nélkül, de 1024 x 800-as felbontású képernyővel, integrált hálózati csatolóval, UNIX alatt ablakkezeléssel, és más szoftverekkel 10 000 DEM-ért kínálja, azzal a céllal, hogy árban is komoly alternatívát nyújtson a PC-vásárlóknak. A gép vagy PC emulátorral futtat MS—DOS programokat az egyik ablakban, vagy egy AT-processzoros, 1 Mbájtos kártyával egészíthető ki az MS—DOS-környezet biztosítása céljából.

Újdonság a DS3500/DS4500 Apollo gépsorozat, 4 Mbájtos tárral, 4/8 MIPS teljesítménnyel, 19 inches színes monitorral 30 000—53 000 DEM-ért kapható.

A **Mentor Graphics** egy Apollo DN10000 gépen mutatta be, három demonstrációban, elektronikai tervezőrendszerét. Az első rész a kapcsolási vázlat elkészítését, a digitális és analog szimulációt és annak dokumentálását mutatta. A gép fantasztikus teljesítményét a tervezés rajzgenerálási fázisa szemléltette. A harmadik ciklus a hőterhelés elemzésétől a háromdimenziós konstrukcióig fogja át a feladatokat.

Az **AUTODESK** az új AUTOCAD Release 10 verziót és a AUTOSolid testmodellezőt mutatta be. A legelterjedtebb CAD-szoftver már régóta fészegeti a PC és az MS—DOS körletait, és így nem igazán meglepetést megjelenése a Macintosh II, a DEC/VMS, és a SUN/UNIX, Apollo/UNIX munkaállomásokon.

A **TEKTRONIX** termékváltási szakaszban van. A cég forgalma '87-ben stagnált, de az új 4300-as, színes grafikus munkaállomás családától gyors felújjászt várna. Mindegyikben a UNIX-szerű UTEK operációs rendszer alatt fut az ANVIL5000, Preview, Model, Image, valamint a háromdimenziós animációt és fotorealista megjelenítést lehetővé tevő Medit szoftvercsomag. A megjelenítés kitűnő minőségét például a TEK 4319 képernyője

1280 x 1024 képponttal, 256 színnel szolgáltatja. Mögötte 68020-as processzossal 4 Mbájtos tárral és 86 Mbájt lemeztárral kiépített számítógép dolgozik. Hozzávetőlegesen 30 000 DEM-ért.

Érdekességek

A müncheni székhelyű Micron 10 MIPS-es 4 Mbájtos UNIX 5.3 (C, FORTRAN, PASCAL) operációs rendszerrel futó gépet kínált ANSI CGI és X—WINDOWS implementációjú képernyőkezeléssel, 1280 x 1024 pixel felbontású színes monitoron — 19 500 DEM-ért.

Egy 80386-os, 25 MHz-es munkaállomás 2 Mbájtos tárral, egy 70 Mbájtos fekete-fehér képernyővel 16 200 DEM-ért kapható.

A bemutatott két grafikus szuperminimális számítógép sok érdeklődőt vonzott. Az ALLIANT/FX sorozatú gépeivel moduláris, 16 processzoros konfigurációval a 4—230 MIPS-es teljesítménytartományban —, a VAX gépcsaládnak kíván versenytársra lenni a tudományos-műszaki grafika területén.

Pomper János

SQLxSTAR

Bár a SYSTEK elsősorban a CIM-termékek kialakítása, az adatbázis-kezelő szoftverek gyártói is fontosnak érezték, hogy az utóbbi években nagy sikereket elért RDBMS rendszerek sokoldalúságát itt is bemutassák.

Központi helyen állított ki — nem véletlenül — az elmúlt év legnagyobb piaci sikerét elkönnyelődő ORACLE Corp., amely új termékét, az SQLxSTAR-t mutatta be.

Az SQLxSTAR nyitott rendszer, amely a földrajzilag osztott erőforrásokat — mint például inkompatibilis számítógépeket, heterogén hálózatokat, valamint a legelterjedtebb operációs rendszerek (az MVS, a VMS, az OS/2, a UNIX) alatt futó különböző SQL-szerű adatbázisokat — egyetlen információs rendszerrel integrálja. Az elosztott adatbázisokhoz például DB2, ORACLE vagy SQL/OS alatt — úgy lehet hozzáférni, mintha azok egyetlen adatbázis részei lennének egy számítógépen, még akkor is, ha igazából egy hálózat különböző gépein vannak.

AZ SQLxSTAR HÁROM ORACLE TERMÉKBŐL ÁLL:

Elosztott ORACLE RDBMS

Az elosztott ORACLE RDBMS a felhasználó számára lokális rendszernek látszik. Nincs olyan központi hely, amely ha kiesik, az egész rendszert leblokkolná, és a felhasználónak az adatok fizikai helyét sem kell tudnia. Ez a földrajzi függetlenség azt is jelenti, hogy a ráépített alkalmazási rendszer független a hálózat aktuális konfigurációjától.

Több tranzakció párhuzamos futásáról megfelelő biztonsági mechanizmusok gondoskodnak. Az osztott tranzakciókezelést úgy valósítja meg, hogy a hozzáférési útvonalakat optimalizálja.

SQLxNet

Az SQLxNet kulcsselem az SQLxSTAR koncepciójában: ez a kommunikációs és interfész-eszköz az osztott rendszerben. Az SQLxNet használatával tudnak a távoli gépek egymással kommunikálni. Az ún. kliens lekérdezi és kezeli, az ún. server szállítja az adatokat. Minden többfelhasználós operációs rendszerű gép futhatja az ORACLE kernelt, és így képes servergépként viselkedni. A tapasztalat szerint a DBMS erőforrás-gép központi egységének terhelése 30-70 százalékban olcsó munkaállomásokra — például PC-re — helyezhető át.

Nyitott rendszer, ezért sok LAN/WAN protokollal támogat. Például az OS/2 PCLAN-oknál az aszinkron összeköttetést, a DECnet-et, az SNA-t, az IBM 3270-est, az Ethernetet, a TOKEN RING-et, a NOVELL-t és a MAP-et.

SQLxConnect — hid az idegen rendszerekhez

Az ORACLE nyílt rendszerként az SQLxConnect segítségével idegen, de SQL-szerű adatbázisokat integrál. Például az ORACLE eszközök kommunikálhatnak az IBM SQL szabványának megfelelő DB2-vel vagy adatbázis-kezelőkkel. Ez a rendszer a SQLQMX, SQL Calc és SQL Plus szoftverekkel ugyanúgy dolgozik, mint egy eredeti ORACLE adatbázissal — akár lokálisan, akár SQLxNet hálózatban.

Nagyobb karakterméret!

C16

Az országban elterjedt számítógép a "magyar-tudása" nem az igazi. Ezzel már a Magazin is sokat foglalkozott. Röviden arról van szó, hogy a 8x8-as karaktermérettel a magyar ékezetek nem helyezhetők el esztétikusan és olvashatóan, valamint az írógépektől eltérő, logikátlan billentyűzet-ellenrendezés miatt nehézkes a gépelés. Az újságokban jelentek már meg különböző mikroszámítógépekre magyarosított programok, de ezek eddig nem javítottak az imént vázolt helyzeten. Mivel magam is hiányt éreztem egy megfelelő karakterkészletnek, nekiláttam olyan segédprogram írásának C16-os számítógépeken, amely nagyobb karakterméretet tesz lehetővé.

Az elkészült program tulajdonságai:

- 12 x 8-as karakterméret,
- 17 soros képernyő,
- a billentyűzetellenrendezés szabadon változtatható.

Fontos, hogy a program a számítógép semmilyen funkcióját ne korlátozza. Memóriatakarékosági okokból azonban csak egy karakterkészletet raktam bele, mégpedig a kisbetűs változatot. Természetesen a karakterkészlet át is tervezhető a mellékelt program segítségével.

Ezek után nézzük, hogyan kell begépelni a programot. Magától értetődik, hogy a memórialistákat nagy figyelemmel kell be-

gépelni. Először is lépünk át a monitorba, majd sorban adjuk ki a következő param-csokat:

F 1000 17FF 00

T D400 D700

> 1600 1E 30 3C 66 66 3C 0C 78

T 1400 16FC 1004

E négy parancs begépelése után pötyög-jük be a memórialistákat (1. lista) 12D8-tól 13FF-ig és (2. lista) 16D8-tól 1A77-ig. Ha ezzel is készen vagyunk, a biztonság kedvéért érdemes kimenteni az 1000—1A77 területet.

A következő beadandó parancs a G 1A61. Ennek hatására a gép visszajut BASIC-be, és SYNTAX ERROR üzenetet ad. A NEW parancs kiadása után gépeljük be a BASIC nyelven írt segédprogramot (3. lista), majd futtassuk. Amikor véget ér a programfutás, NEW-val töröljük ki, és lépünk át monitorba ismét. Mentünk ki az 1000—1A77 memóriaterületet — ez már a végleges program. Rögötn vagy beolvasás után aktiválható a következő módon: G 1A10, majd NEW. Az adattárolóról — szagraolvasást feltélenül monitor módban csináljuk, hogy a program ugyanoda kerüljön vissza, ahonnan kimentettük.

Az aktivizálás után a gép kikapcsolásig vagy resetig használhatjuk az újfajta karak-

```

>1000 20 BF CF 20 CD CE E5 FB
>1008 48 F9 00 85 FB 09 5B 20
>1010 11 0B 26 68 95 FB 60 2C
>1018 09 07 10 0E 01 F0 20
>1020 04 07 10 0E 20 95 EA 20
>1028 58 EA 20 EA E3 60 EA EA
>1038 20 36 14 4E BE FC 0D 89
>1048 FF 29 82 60 85 FF F8 D7
>1044 0D 00 FF 4C 10 EA EA
>1048 EA EA EA EA EA EA EA EA
>1058 18 69 03 80 0E FF 38 0D
>1068 10 FF E9 04 0A 0D 1F FF
>1060 E9 04 0A 09 15 8E 10 FF
>1068 0C 1F FF 03 13 FF A9 0A
>1070 04 44 18 0D 06 FF C9 9A
>1078 08 01 60 09 02 80 8E FF
>1088 09 C5 8D 10 FF 0D 12 FF
>1088 29 FB 80 12 FF 20 00 18
>1090 09 18 0D E5 07 80 03 8D
>1098 E5 07 60 EA EA EA EA EA
>10A0 09 11 8D 13 FF 0D 06 FF
>10A8 18 69 05 80 0E FF 09 4C
>10B0 0D 44 18 60 EA EA EA EA EA
>10B8 E9 EA EA EA EA EA EA EA EA
>10C0 EA EA EA EA EA EA EA EA EA
>10C8 EA EA EA EA EA EA EA EA EA
>10D0 EA EA EA EA EA EA EA EA EA
>10D8 EA EA EA EA EA EA EA EA EA
>10E0 00 3E FF 04 8A 4E 96 49
>10E8 8A 8D 04 01 29 10 F8 83
>10F0 4C 83 86 36 5F 8E 8E 8E
>10F8 FF 60 69 69 68 40 8E
>10A0 43 05 C9 03 0D 19 0D
>10B0 47 05 30 84 44 05 06
>10C0 1F 13 FF 49 04 8D 13
>10D0 F9 A9 08 44 05 06 20
>10E0 C9 08 90 10 A9 0E
>10F0 07 08 07 06 0E 08
>10A0 03 8E 60 60 6A 49
>10B0 19 85 CE 44 19 85 ED

```

1. lista

```

>1205 00 00 00 00 7E 30 18 00
>1209 00 00 00 00 00 00 00 FF
>120E 00 00 00 00 18 10 18 00
>1215 00 00 00 00 00 33 FE 00
>121F 00 00 00 00 00 04 06 00
>1300 00 00 00 00 00 00 00 00
>1309 00 00 00 18 10 00 3C 00
>1310 00 00 00 18 10 00 3C 00
>1318 00 00 00 18 10 00 1B 00
>1320 00 00 00 18 10 00 3C 00
>1328 00 00 00 66 00 3C 00
>1330 00 00 00 66 44 00 3C 00
>1338 00 00 00 18 10 00 66 00
>1340 00 00 00 66 00 66 00 00
>1348 00 00 00 66 44 00 66 00
>1350 00 00 00 66 00 3C 00
>1358 00 00 00 00 1F 1F 1F 1F
>1360 00 00 00 00 00 F8 F8 18
>1368 00 18 18 18 18 F8 F8 00
>1370 18 18 18 18 1F 1F 00
>1378 18 18 18 18 18 18 18 18
>1380 18 18 18 18 FF FF 18
>1388 00 18 18 18 3C 66 00
>1390 00 18 10 00 7E 60 60 00
>1398 00 18 10 00 3C 19 10 00
>13A0 00 18 10 00 3C 66 66 00
>13A8 00 66 00 3C 66 66 00
>13B0 00 66 44 3C 66 66 00
>13B8 00 18 10 00 66 66 66 00
>13C0 00 00 66 00 66 66 66 00
>13C8 00 66 44 00 66 66 66 00
>13D0 00 18 10 00 3C 66 66 00
>13D8 00 00 00 3C 19 10 00
>13E0 18 18 18 18 F8 F8 18
>13E8 18 18 18 18 1F FF 00
>13F0 18 18 18 18 1F 1F 18
>13F8 00 00 00 00 FF FF 00

```

2. lista

```

>1608 7E 30 18 0C 18 30 7E 00
>1609 00 00 00 00 00 00 00 FF
>160E 18 18 18 18 7E 3C 18 00
>1618 00 33 FE 66 66 66 66 00
>161F 00 04 06 7F 7F 04 04 00
>1700 00 00 00 00 00 00 00 00
>1709 10 00 3C 06 3E 66 3E 00
>1710 10 00 3C 66 7E 60 3E 00
>1718 10 00 18 10 18 18 00
>1720 10 00 3C 66 66 66 30 00
>1728 66 00 3C 66 66 66 3C 00
>1730 44 00 3C 66 66 66 3C 00
>1738 10 00 66 66 66 66 3C 00
>1740 66 00 66 66 66 66 3C 00
>1748 44 00 66 66 66 66 3C 00
>1750 66 00 3C 06 3E 66 3E 00
>1758 00 1F 1F 18 18 18 18
>1760 00 F8 F8 18 18 18 18
>1768 18 F8 F8 00 00 00 00
>1770 1F 1F 00 00 00 00 00
>1778 18 18 18 18 18 18 18
>1780 1F FF FF 18 18 18 18
>1788 18 3C 66 7E 66 66 66 00
>1790 7E 60 68 78 60 60 7E 00
>1798 3C 18 18 18 18 3C 00
>17A0 3C 66 66 66 66 66 3C 00
>17A8 3C 66 66 66 66 66 3C 00
>17B0 3C 66 66 66 66 66 3C 00
>17B8 66 66 66 66 66 66 3C 00
>17C0 66 66 66 66 66 66 3C 00
>17C8 66 66 66 66 66 66 3C 00
>17D0 18 3C 66 7E 66 66 66 00
>17D8 00 FF FF 18 18 18 18
>17E0 18 F8 F8 18 18 18 18
>17E8 18 FF FF 00 00 00 00
>17F0 1F 1F 18 18 18 18 18
>17F8 00 FF FF 00 00 00 00

```

```

>1940 4C 8A 0B 4B 19 8C 19 CD
>1948 19 8E EA 14 0D 86 8C 85
>1950 00 85 83 3F 34 36 3F
>1958 53 45 01 25 52 44 36 43
>1960 46 54 59 37 54 47 38 42
>1968 49 55 56 39 49 4A 30 41
>1970 4B 4F 4E 11 50 4C 91 2E
>1978 82 87 2C 90 8A 81 1D 1B
>1980 89 8A 2F 31 13 84 8D 20
>1988 8C 91 8B 85 23 D7 C1
>1998 79 C3 05 01 25 02 C4 26
>19A0 C3 C6 D4 08 27 79 C7 28
>19A8 C2 C8 D5 D6 29 C9 6A
>19B0 C0 CB CF CE 11 D8 C0 C1
>19B8 3E 82 87 3C 9D 8B 81 1D
>19C0 18 89 84 31 23 84 22
>19C8 7F 02 87 8A 87 84 85 86
>19D0 88 8A 97 8B 49 8C 8A
>19D8 97 8E AC 8D 81 98 8B 8F
>19E0 9F 8F 80 74 8D 9A 87 6F
>19E8 9E 62 68 75 76 29 83 6A
>19F0 6D 6A 8A 83 6E 7D 7E 8A
>19F8 5E 5D 3E 2D 5B 9F 2B 38
>1A00 7F 1B 02 8A 8B 8C 8D
>1A08 95 7C 0E 83 8F FF FF
>1A10 78 85 8D 84 18 83 89 18
>1A18 6D 15 03 89 87 8D 8E FF
>1A20 8D 12 FF 29 FF 8D 12 FF
>1A28 A9 50 8D 44 18 20 8D 80
>1A30 A9 18 8D 46 85 A9 8D 80
>1A38 45 85 82 8D 80 8D C9
>1A40 7F F8 04 C9 8D 04 85 8C
>1A48 EA 9D 96 8A 8E 8D 80
>1A50 E8 8D 3F FF 89 DC 8D FE
>1A58 FF A9 18 8D FF FF 8D 80
>1A60 FF 89 71 85 2B 89 1A 85
>1A68 2C 58 66 00 00 00 00
>1A70 00 00 00 00 00 00 00
>1A78 00 00 00 00 00 00 00

```


terméretet. Monitor reset után az újraindításhoz elég a G 1A10 parancs, de ha sima resetet használunk, ez után a BASIC program visszahozásához ki kell adni egy RENUMBER parancsot. Ha a tárban lévő BASIC programra nincs szükség, akkor a NEW parancsot adjuk ki. Helytakarékosság miatt itt nem közlöm azt, hogy milyen karakterkódhoz milyen kép tartozik, de a

```
FOR R=0 TO 255:POKE 3072+R,R:
NEXT R
```

parancssor kiírja a képernyőre a teljes karakterkészletet az inverzekkel együtt. Az ASCII kódokra ebből már következtetni lehet (javasolom a kísérletezést).

Biztosan sokan akarják majd megváltoztatni a karakterkészletet vagy a billentyűzetkiosztást. Nekik segítség a mellékelt két BASIC program. A KARAKTERSZERK. nevű program (4. lista), mint a neve is mutatja a karakterek áttervezését segíti. A program először megkérdezi, melyik kódot akarjuk áttervezni. A válasz 0 és 127 között legyen. Mozogni a kurzorbillentyűkkel, pontot törölni a CLR, kifesteni a HOME lenyomásával lehet. A szerkesztésből az ESC gombbal léphetünk ki. Ha a hova? kérdésre 127-nél nagyobb választ adunk, akkor nem tárolja el sehova a tervezett képet. A tovább? kérdésre az I vagy az N betűvel válaszolhatunk.

A BILLENTYÜZETKÓDOLÁS nevű programmal (5. lista) megváltoztathatjuk a billentyűzet elrendezését. Itt a melyik kódot? kérdésre természetesen nem a képernyőkódot, hanem az ASCII kódot kell megadni. A hova? kérdés megjelenése után le kell nyomni azt a billentyűt, ahova az adott kódot kívánjuk helyezni. A SHIFT vagy a COMMODORE billentyűvel együtt is lenyomhatjuk a billentyűt, de a CTRL-lel nem, mert annak a táblázatát a ROM-ban hagyta meg, memóriatakarékosság miatt.

Aki csak használni akarja a programot, annak úgy vélem, ennyi információ elég. Azoknak, akik meg szeretnék ismerni a működés hogyanját, fontos lehet, hogy a hosszú kódon belül mi mire szolgál.

A karaktergenerátor 1000 és 1800 között van. 1800 és 1900 között található a tulajdonképpeni programunk, amely a megszakításkezelés rutinjába épül be. A program működési elve az, hogy hat rasztersoroként megszakítást generál, és a TED chip regisztereinek átállításával — aktuális raszteror, karakteren belüli raszteror, karaktergenerátor helye — mintegy becsapja a TED-et. Az 1000 és 1400 között levő karakterkészlet a 12 pont magas karakterek felső 7 és legalsó sorát, az 1400 és 1800 közötti karakterkészlet pedig az alsó 8 sorát tartalmazza.

Az 1900 és 1A10 közötti rész a billentyűzetkezelést szolgálja. Nagy része a billentyűzetdekódolás táblázatait tartalmazza. Végül az 1A10 címen kezdődő rutin kapcsolja be a 12x8-as karakterméretet.

A programot most már mindenki tetszése szerint használhatja. Abban is biztos vagyok, hogy a sok C64-tulajdonos gondol

majd arra, ez a program esetleg az ő gépére is átirtható. Sajnos, nincs a TED regiszterek mindegyikének megfelelője a C64 VIC chipjében, melyeket a program felhasznál. Ezért ezzel a módszerrel nem lehetünk eredményesek, ami persze nem zárja ki, hogy más módon ne lehetne megoldani a problémát C64-re is.

Horváth Géza

4. lista

```
10 REM *** KARAKTERSZERK. ***
20 DIMV<11>
30 INPUT "Q          KEPERNYOKOD ";KO
40 PRINT "Q"; ; IFKO>127ORKO<0THEN30
50 FORI=4096+8*KOTO4096+8*KO+7
60 N=PEEK<I>
70 FORR=1TO8
80 IFN<128THENPRINT". "; ;GOTO100
90 PRINT"*"; ;N=N-128
100 N=2*N:NEXTR:PRINT:NEXTJ
110 PRINT "X00000";
120 FORI=4096+1024+8*KOTO4096+1024+8*KO+7
130 N=PEEK<I>
140 FORR=1TO8
150 IFN<128THENPRINT". "; ;GOTO170
160 PRINT"*"; ;N=N-128
170 N=2*N:NEXTR:PRINT:NEXTI
180 Q=3*1024:X=0:Y=0
190 POKEQ+X+40*Y,PEEK<Q+X+40*Y>+128
200 GETKEYA#
210 POKEQ+X+40*Y,PEEK<Q+X+40*Y>-128
220 IFA#=#"ANDX<>0THENX=X-1:GOTO190
230 IFA#=#"ANDX<>7THENX=X+1:GOTO190
240 IFA#="J"ANDY<>0THENY=Y-1:GOTO190
250 IFA#="I"ANDY<>11THENY=Y+1:GOTO190
260 IFA#="J"THENPOKEQ+X+40*Y,42
270 IFA#="I"THENPOKEQ+X+40*Y,46
280 IFA#<>CHR$(27)THEN190
290 PRINT"          KODOLAS..."
300 FORR=0TO11:V<R>=0
310 FORI=0TO7
320 IFPEEK<Q+40*R+I><>42THEN340
330 V<R>=V<R>+2+(-1)
340 NEXTI:NEXTR
350 INPUT "XHOVA";HO
360 IFHO>127ORHO<0THEN410
370 FORR=0TO7
380 POKE4096+8*HO+R,V<R>
390 POKE4096+1024+8*HO+R,V<R+4>
400 NEXTR:POKE4096+8*HO+7,V<11>
410 PRINT"XTOVABB?"
420 GETA#:IFA#="I"THEN30:ELSEIFA#<>"N"THEN420
```

5. lista

```
10 REM *** BILLENTYUZETKODOLAS ***
20 PRINT"Q          BILLENTYUZETKODOLAS"
30 INPUT "XMELYIK KODOT ";KO
40 IFKO>255ORKO<0THEN20
50 IFKO=0THENEND
60 PRINT"HOVA?"
70 B=PEEK<198>:SH=PEEK<134>
80 IFB=10ORB=64THEN70
90 IFSH=0THENKE=6475:GOTO130
100 IFSH=1THENKE=6540:GOTO130
110 IFSH=2THENKE=6605:GOTO130
120 GOTO70
130 POKEKE+B,KO:GOTO20
```

```
10 REM *** SEGEDPROGRAM ***
20 FORR=0TO90:K=4096+8*R
30 FORQ=0TO3:POKEK+Q,0:NEXTQ
40 POKE K+7,PEEK<K+1031>
50 NEXTR
```

3. lista

A Tasword módosítása

A ZX-Spectrumot kedvelők táborában közismert Tasword 2 és Tasword 3 programok a beírt szöveg 64 karakteres kiírásakor a kisbetűket úgy jelenítik meg, hogy azok elég nehezen olvashatók. Ezen a több mint szépséghibán segít az alábbi kis BASIC program, amely a kisbetűt mintegy 70 százalékat újradefiniálja. A régi és az új betűkkel — kétszeres nagyításban — írt szöveg az *ábrán* látható.

A Tasword Two programot betöltjük, majd a Symbol Shift + A (STOP utasítás) billentyűk lenyomásával egy menüt kapunk. Ekkor BREAK-kel megszakítjuk a Tasword Twot, és beírjuk a kis BASIC programot, majd RUN 9000-rel futtatjuk. A 9500-zal kezdődő sorokat nem kell futtatni! Ezután a Tasword Twot RUN-nal indítjuk, és egy szöveg beírásával ellenőrizzük az eredményt. Ha megfelelően működik, az előzőekben leírt módon szakítsuk meg a Tasword Two futását (STOP és BREAK), töröljük ki a 9000-től kezdődő so-

rokat, és a módosított programot mentjük ki kazettára (RUN — STOP — T).

Ha a Tasword Two/H 2.0 (MIKROTEAM) verziót használjuk, az ékezetes betűket a RUN 9500 utasítással kell módosítani.

A Tasword 3 máshol helyezkedik el a memóriában, ezért úgy kell futtatni, hogy a BASIC programunkban előforduló abszolút címekből 22748-at levonunk.

Egyszerű fogással a Tasword Two-nál a látható szövegrészről 64 karakteres hardcopyt is tudunk készíteni. A STOP és BREAK utasítással megszakítjuk a programot, majd újraindítjuk RANDOMIZEUSR 64330-cal. Ekkor a Tasword Two „látzólag” rendben elindul. Kikeressük a nyomtatni kívánt szövegrészt. A STOP utasítás hatására nem a szokásos menü jön elő, hanem a program BASIC-be lép, és a képernyőn ott marad a kiválasztott szövegrész, amelyet a COPY utasítással kinyomtathatunk.

Vadász László

```

9000>POKE 61769,0:POKE 61777,0:PO
OKE 61783,3:POKE 61862,3:POKE 61
843,4
9100 RESTORE 9120: FOR n=0 TO 27
9110 READ a: POKE 61706+a,7: NEX
T n
9120 DATA 0,2,8,12,16,20,24,28,3
2,34,36,48,51,53,56,92,112,116,1
20,123,128,131,136,144,146,148,1
95,197
9200 RESTORE 9220: FOR n=0 TO 5
9210 READ a: POKE 61715+a,5: NEX
T n
9220 DATA 0,16,48,90,168,171
9300 RESTORE 9320: FOR n=0 TO 5
9310 READ a: POKE 61768+a,2: NEX
T n
9320 DATA 0,3,26,27,28,29
9400 POKE 61745,6: POKE 61770,6:
POKE 61776,1: POKE 61778,1: POK
E 61793,6
9450 STOP
9500 RESTORE 9520: FOR n=0 TO 9
9510 READ a: POKE 61667+a,7: NEX
T n
9520 DATA 0,3,31,35,247,249,251,
255,259,271
9530 POKE 61666,0
    
```

Ezek a Tasword Two eredeti karakterei.

a á b c d e é f g h i j k l m n o ó ö ő p q r s t u ü v w x

A szöveg a Tasword Two/H 2.0 változattal készült!

Ezek a Tasword Two módosított karakterei.

a á b c d e é f g h i j k l m n o ó ö ő p q r s t u ü v w x

A szöveg a Tasword Two/H 2.0 változattal készült!

Sok programozó nagy gondot fordít arra, hogy programját az avatatlan szemek elől elrejtse. Nos, ennek semmi akadályja, ha valaki az alábbi, TV-Computerre készült programot a kellektárából alkalmanként előveszi.

A program megértéséhez ismerni kell a TVC IS — BASIC programtárolási módszerét. A BASIC számára fenntartott tárterület alapállapotban a 6639 (dec) címtől kezdődik.

Például az

```

1 REMx
sort a következőképpen tárolja:
6639: 6 következő sor távolsága (eltolási érték)
6640: 1 sorszám INTEL formátumban
6641: 0
6642: 252 REM kódja
6643: 88 x
6644: 255 sorvégjelző
6645: 0 programvégjelző (0 sorhossz)
6646:
    
```

A BASIC szöveg végét a sorhossz 0 értéke jelzi.

A titkosításra az ad lehetőséget, hogy az interpreter a program listázásánál és futtatáskor más-más módon tér rá a következő sor kiírásá-

ra, végrehajtására. Futtatáskor az aktuális sor kezdőcíméhez hozzáadja az eltolási értéket.

Listázásnál viszont csak azt vizsgálja, hogy az eltolási érték nulla vagy nem nulla. Ha nem nulla, kiírja a sort a sorvégjelző 255-ig, majd ha az ezt követő bajt nem nulla, folytatja a tevékenységét. A megoldás ezek után kézenfekvő: a program elején létre kell hozni egy olyan sort, amelynek jó az eltolási értéke, tehát futás-képes, viszont tartalmaz a sor végi 255 után egy „látzólagos programvéget” jelző 0 bajtot, megévezte így a LIST utasítást. Nézzünk egy példát.

```

6639: 6 eltolási érték (következő
sor = 6639 + 6 = 6645)
6640: 1 sorszám INTEL formátumban
6641: 0
6642: 252 REM kódja
6643: 255 sorvégjelző
6644: 0 látszólagos programvég
6645: x következő sor
    
```

Még egy apróság: az első sornak 0-ás sorszá-mának kell lennie, hogy ne lehessen javítani. Ugyanis módosításkor az egész programot új-ralancolja a rendszer, eltüntetve „bűvészkedé-

Listázás elleni védelem

sünk” nyomait: a teljes program újra listázhatóvá válik. Egyébként ez a módszer csak a LIST ellen nyújt védelmet, a LISTI-parancs-sal továbbra is kiírható a teljes lista. Ezek után nézzük a programot!

```

1 REMx
3 LM = PEEK(5922) + 256 * PEEK(5923)
4 POKE LM + 1,0:POKE LM + 4,255:POKE LM + 5,0
5 REM *** A program ***
    
```

A négy sor egyébként csak a teljesen kész programunkba tegyük bele, lefuttatása után mentjük ki! Ha a programunk tartalmaz LO-MEM utasítást, akkor azt a 2-es sorba tegyük be!

Nagy István

Adom a magyarázatot!

A Magazin 1988/10. számában a Ki ad magyarázatot? című cikkben felsorolt ismeretlen 6500-as kódokat megfejtettük. Jelöléseink:

- * — a címzési módból kapott cím felső bájta+1 (például \$6000-nél \$6001)
- A — akkumulátor
- X — regiszter
- SR — státuszregiszter
- SP — stack pointer (veremmutató)

Az utasításneveket a műveleteknek megfelelően választottuk.

- 1. \$93 SXH (Zp),Y: (cim) = A AND X AND X
SR: —
Egyetlen regiszter értéke sem változik!
- 2. \$0B AND # } egyenértékű az \$29 kódú utasítással
- 3. \$2B AND # }
- 4. \$4B ALR # : A AND # ; LSR A
SR: Z, N, C
- 5. \$6B ARR # ; A AND # ; RORA
SR: Z, V, N, C
C bit: az A 6-os bitje a művelet után
V bit: 1, ha az A 6-os bitje megváltozott, egyébként nem változik
- 6. \$8B ANX # : A = A AND X AND *
SR: N, Z
- 7. \$9B SXS absz,Y: SP = A AND X ; (cim) = A AND X AND *
SR: N, Z
- 8. \$9C HAY absz,X: (cim) = * AND Y
SR: —
- 9. \$9E HAX absz,Y: (cim) = * AND X
SR: —

- 10. \$9F SXH absz,Y: (cim) = A AND X AND *
SR: —
Egyetlen regiszter értéke sem változik
- 11. \$AB ATX # : A AND # ; TAX
SR: N, Z
- 12. \$BB PAX absz,Y: SP = A = X = SP AND (cim)
SR: N, Z
- 13. \$CB XAS # : X = X AND A ; SEC ; X = X SBC #
SR: N, Z, C
SR bitjeinek a változása X regiszter változásait mutatja
- 14. \$EB SBC # : azonosan egyenlő a \$E9 kódú utasítással
Mivel ezeket a kódokat C Plus/4 számítógéppel fejtettük meg, és a Magazin 1986/8. számában közölt táblázat addig ismert utasításai is megegyeznek a C Plus/4-ével (ellenőriztük!), ezáltal azt a következtetést vontuk le, hogy a 85XX, 75XX, 85XX típusú processzorokkal működő gépeken is ugyanezeket a hatásokat fejtenék ki az új utasítások.

Macska József és társai

A rovat 1989/1. számában ígértem, hogy a következő számban folytatom az egyetlenrendszer megoldó program javítását. Ez elmaradt az amerikai utam miatt, de az áprilisi számtól kezdődően visszaterék rá.

Gondolom, az olvasók egy részében felmerül az a kérdés, hogy miért foglalkozom ezzel olyan sokat? Egyszerűen azért, mert a feladat sokszor fordul elő (akkor persze elég lenne közölnöm a jó megoldást és mellőzni a részletes magyarázatot!), de elsősorban nem ez az igazi ok, hanem az, hogy a programmal kapcsolatos problémák általában sokak, és a gépi számítások pontosságáig mindenkinek érdekes lehet.

S. E.

Egy- és kétsorosok

A RAINBOW nevű amerikai szaklap minden számában közli az olvasók által beküldött legérdekesebb egy- és kétsoros programokat. Ezek a programozók valódi ügyességpróbái, hiszen legfeljebb 255 karakternyi helyet kell a lehető legjobban kihasználniuk (a lap a Tandy cég CoCo típusú számítógépeivel foglalkozik, és az ezekre írt Microsoft BASIC változat ennyit karaktert fogad el egy sorban). Az már szinte ráadás, hogy ezek a programok gyakran még jól használhatók is.

A sjátétek igen elterjedt játékpogramok. Közülük is a legismertebbek a lesiklájjátékok. A Magazinban aránylag kevés játékpogramot közölnek. Miért kell akkor egy ilyen? Mert ez mindössze két sor, és mégis benne van minden lényeges eleme az ilyen játékoknak. Nagyon ötletes

például az, hogy a sielőt botokkal egy nagy H betű adja.

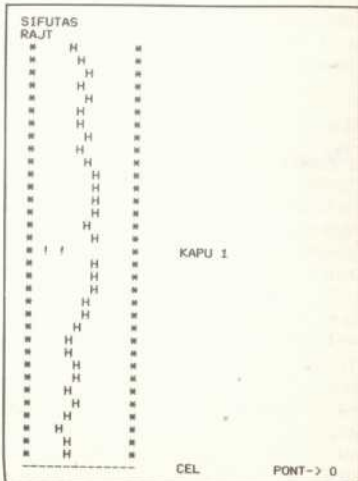
Az eredeti program az itthon kevésbé elterjedt analóg botkormányt használta. Ezt megváltoztatva most a J és B feliratú gombokkal lehet a sielőt jobbra-balra kitéríteni.

A listában látható program más gépekre átirásához tudni kell, hogy a CLS a képernyőt törli, a POKE 1024 a képernyő bal felső sarkának megfelelő tárolóhely — ahová az ASCII 72 értéknek megfelelő H betűt kell beírni, illetve oda, ahová a sielőt kitérítjük —, a STRING\$(14,"—") pedig 14 darab kötőjel írását jelenti.

A listán kívül kinyomtattam az ábrán az induló és a befejező részt.

A cím — tévedésből sifutas — kiírása után a program kiírja a "rajt" szót, és némi időt hagy a felkészülésre. Az ábrákon, az igazítól eltérően, minden sorban látható a sielő, a pálya azon pozíciójában, amelybe kormányoztuk. A kapukra a kapukat jelentő két ! jelen kívül a „kapu” és annak sorszáma figyelmeztet. Ezek 14 pozícióval előre láthatók, van tehát időnk a kormányzásra. A húsz kapuból álló pályán végighaladva elérjük a célt, ahol is megjelenik a pontszámunk. Ez kapunként 5 lehet. A pályát nem ismerhetjük ki, mert a kapuk véletlenszerűen kiválasztott helyekre kerülnek.

LESIKLÁS



```

10 CLS:PRINT "SIFUTAS":FOR I=1 TO 999:NEXT I:P
=>PRINT "RAJT":FOR I=1 TO 999:NEXT I:CLS:FOR I
=1 TO 2:FOR J=1 TO 14:FOR K=1 TO 14:FOR L=1 TO 14:
TOSO:INEXT L:PRINT " *TAB(14)*" *POKE 1024+P
,72:Y=INKEY$:IF Y="J" THEN P=P+1:IF P=13
THEN P=12:ELSE IF Y="B" THEN P=P-1:IF P
=2 THEN P=3
20 FOR J=1 TO 500:INEXT L:INEXT K:IF I=2 THEN IF G=
P THEN SC=S+PRINT "STRING$(14,"-")TAB
(14,"-") *TAB(20)*CEL", "POINT->"SC ELSE PRINT
" *TAB(14)*" *TAB(14)*" *TAB(20)*"KAPU
1":IF G=P THEN SC=S+5:INEXT L:INEXT K

```

Simonyi Zsuzsa

A The Newsroom magyar változata I. rész

A Newsroom programmal kisebb intézmények, iskolák, vállalatok saját belső újságját lehet megszerkeszteni. Magyarországi elterjedésének mindmáig akadálya volt, hogy a program nem írt magyar betűket, ugyanis ilyen feladatra nem készítették fel. E probléma „feloldása” azért érdekelt, mert csupán egyetlen programot ismerek ezen kívül, amelyik „profi” betűtípust ír: ez a Geos.

De lássuk a Newsroom lehetőségeit. A Newsroom alkalmazásáról az 1001 játék című könyvben már jelent meg leírás.

Az alábbiakban röviden ismertetem a program üzemmódjait, hogy azután a későbbiek érthetőek legyenek.

A Newsroom jelentése: szerkesztőség. A program részei a következők.

Banner – főcím készítése

Lehetőségei: Clip art ábra beolvasása, grafika alkalmazása, szöveg írása.

A Bannert a gép grafikusán tárolja, ezért nagy a helyigénye. Ezzel szemben tetszőlegesen keverhetők a betűtípusok, aminek a jelentőségét később ismertetem. A Banner a nyomtatóipar teljes szélességét kihasználja.

Photo Lab

A Photo Lab nyújtja a legtöbbet, gyakorlatilag a Newsroom valamennyi lehetőségét: Clip art ábra beolvasása, grafikus ábrakészítés, szövegírás, előzőleg már elkészített fotó beolvasása (beragasztása).

Az 1. ábrán szereplő cirill betűs szöveget fotóként is beragaszthatjuk a magyar betűs szövegbe.

A Photo Lab grafikusán tárolja az adatokat, ezért igen nagy a tárolókapacitás-igénye. Viszont ha egy fotót tervezünk valahova, annak bármilyen kicsiny részét kiválaszthatjuk („take a photo”). Ehhez a baloldalt, alul látható kis fényképezőgéphez vezetjük a kurzort, a tüzgombbal kiválasztjuk, majd a kijelölni kívánt részlet két áttelnes sarkát is épp így jelöljük.

A Photo Lab és a Banner üzemmóddal bármely képrészlet kinagyítható, ami azt jelenti, hogy (mint a Geosban) gyakorlatilag pixelenként, nagy felbontású üzemmódban rajzolhatunk.

Copy desk

A Copy desk nem más, mint az újságírás technikai eszköze. Szimbóluma az írógép: ez a szöveg is a Copy desk üzemmódban készült. Lehetőségei: szövegírás, fotó beragasztása.

A Copy deskben írott szöveg tárolási helyfoglalása igen kicsi, mert a program a betűk ASCII kódját tárolja. Csak két betűtípus használható: praktikusan az összeálló kis- és nagybetűméret.

Layout

A layout magyarul a tördelés műveletét jelenti. Ebben az üzemmódban tervezhetjük meg az újság egy szövegoldalát. A szövegoldal legfelül egy Bannert (főcímet) tartalmazhat, alatta két olyan hasábot, amelyek a papír méretétől függően három vagy négy szövegrészt (panelt) foglalnak magukba. A tördelt oldal (page) lemezen rögzíthető. Helyfoglalása minimális, mert csak az oldalra betördelt állományok névt kell tárolnunk.

Ez a nyomda, itt készülnek el a nyomtatott szövegeink. Figyelemre méltó, hányféle nyomtató működtetésére készítették fel a Newsroomot! Én az MPS-801-et használom.

A Newsroom fontos sajátossága, hogy teljes egészében grafikus üzemmódban nyomtat. Ezért ön maga határozza meg a betűk írásképét. Tehát nem a felemás Commodore-betűket nyomtatjuk, amelyek között például a kilences és a kis g betű összetéveszthető. A kisméretű betűk 8x8 pixel, a nagyméretűek 16x16 pixel méretűek lehetnek. Banner üzemmódban a betűméreteket a program megkétszerezi.

Wire Service

Talán telexszobának lehetne fordítani. Ezzel a Newsroom egyedülálló a szövegszerkesztő típusú programok között: telefonon létrehozott vezetékcsatlakozással szövegeket képes közvetíteni. Az erre szolgáló menüből ötféle interfész típus választható.

Clip art

Clip art a nevük a szoftvergyártó cég által előre elkészített kis ábráknak. Ezek három teljes lemezoldalt töltenek ki. Jellemüket tekintve főként karikatúrák, de vannak köztük igen jó minőségű térképek is.

Cikkemben erre nem térek ki bővebben, mert célom a betűformákkal kapcsolatos tudnivalók ismertetése.

Karakterkészletek

A Newsroom ötféle nevet ír ki a képernyőre, ha betűtípust választunk. A két kisméretű betűtípus a Serif small és a Sans Serif small. Ezek a Newsroom betöltésekor betöltődnek. Betűtípus váltásakor csak a választás céljait szolgáló menü töltődik be a lemezről. Ha kijelöljük az általunk használni kívánt betűtípust, utána lemezművelet nincs. Banner és Photo Lab üzemmódban az újonnan választott betűtípussal folytathatjuk a szövegírást. Copy desk üzemmódban valami furcsa történik: a gép a teljes szöveget átalakítja az újonnan választott betűtípusra.

Ennek a tulajdonságnak az a következménye, hogy ha magyar betűs programot cirill betűs szöveggel akartam folytatni, az egész Newsroom programot újra be kellett töltenem a gépbe.

1. ábra

Привет читающим!

2. ábra

Καλησπέρα. Καλώς ήρθες.

A nagybetűknél más a helyzet. A Newsroom nagyméretű betűtípusai a Serif, a Sans Serif és az English. Ezek a kisméretű betűkhöz hasonlóan menüből választhatók. Minthogy terjedelmesek, a gép minden választás után lemezről tölti be az új betűtípust. Ezek tehát keverhetők.

A magyar betűk elkészítésénél azt az eljárást követtem, hogy valamennyi betűtípust és betűméretet átírtam magyarra, mert nem láttam értelmét annak, hogy a betűválaszték csonka legyen. Ezzel szemben az idegen nyelvek betűkészleténél csak a Serifet írtam át. A magyarországi felhasználónak ez a praktikus, mert így egy lemezen van meg neki az idegen és a magyar betűtípus. Erre akkor van szüksége, ha egy hosszabb idegen nyelvű, összefüggő szövegben magyar szót vagy megjegyzést kell elhelyezni. Ezért például a görög betűs betűkészletem tartalmaz egy komplett kis- és egy komplett nagyméretű görög betűválasztékot, valamint egy kis- és két nagyméretű magyar betűválasztékot. Így lehet például a 2. ábrán lévőhöz hasonló szövegeket beszúrni a szövegbe. (Jó estét! Isten hozott!)

A billentyűzet kiosztása

A Newsroom összesen 90 billentyűválasztatot olvashat le. Ez elég kevés. A magyar betűknél az alábbiak szükségesek:

- 26 betű az angol ábécéből,
 - 9 magyar ékezetes betű,
 - kis- és nagybetűk, együtt 70 darab,
 - 10 számjegy,
 - 10 elengedhetetlen írásjel,
- összesen 90 darab.

Ennek az a magyarázata, hogy a Newsroom a Commodore billentyűt nem kódváltásra használja, hanem vezérlőbillentyűként. Ha ugyanis a programot botkormány nélkül használjuk, ez a billentyű tölti be a tűzgomb szerepét. Mindennek tetejében nem használhatunk minden billentyűt együtt a shift billentyűvel. Ezek a

- 67 shift csillag,
- 94 shift plusz,
- 96 shift mínusz
- 98 shift kukac (Klammeraffe).

Helyettük az alapbillentyűhöz tartozó betű jelenik meg, mintha a shiftet le sem nyomtuk volna.

A karakterkészlet első két eleme nincs értelmezve. A 3-as a space, a 4-es a felkiáltójel, és a betűsor a 88-asig folytatódik. Erre a helyre grafikus karaktert terveztek — kockát illetve háromszöget —, de, mint említettem, ezt a Newsroom már képtelen kiírni. Az utolsó kiírható billentyű a 97-es (shift nyíl), ide a nagy rövid Ű betűt helyeztem.

A billentyűzettervet később ismertetem. A lehetőségek szerint az Easy Scriptével megegyezik, illetve ahhoz közel áll.

Akít a program közelebről érdekel, az a Kertészeti és Élelmiszeripari Egyetemen, a Fizika Tanszéki csoportnál férhet hozzá: 1118 Budapest, Somlói út 14–16.

Dr. Zana János

Számítógépek és környezetük

A számítógépek leírása gyakran attól érthetetlen a kívülállóak számára, hogy az író valamilyen okból tárgyat *környezetéből kiszakítva*, önmagában ábrázolja. Pedig — ha ennek a kijelentésnek van egyáltalán értelme — *önmagában csak az univerzum áll.*

Gépeink nem *szorsszerűen*, hanem attól olyanok, amilyenek, hogy *illeszkedniük* kell környezetükhöz. A gépeket valamilyen szervezett formában, emberek közreműködésével vagy (valahova: repülőgépekbe, űrhajókba stb. beépítve) anélkül **munkát** kell végeznünk. Ahogy egy jól tervezett balta nyelének, a ma használatos gépkocsik műszerfalának stb. jól kell illeszkedniük a használati módhoz, úgy kell, hogy a gép parancsnyelve, megszabási rendszere, állománykezelése stb. megfeleljen annak a tipikus — mondjuk így — *alkalmazási technológiának*, üzem-módnak, melynek keretében használják.

Az alkalmazási környezet, a munka természete igen eltérő lehet. Dolgozhatunk gépeinkkel úgy, hogy

- az olcsó (**személyi**) számítógép egészen addig „ott áll a sarokban, mint egy írógép”, ami tulajdonosa be nem kapcsolja, hogy aztán dolga végeztével ismét otthagya. Ilyenkor a tulajdonos kezeli az adathordozókat, vigyáz arra, nehogy elveszzenek, összekeveredjenek, még mindig átmásolta („elmentette”) volna a rajtuk lévő értékes adatokat stb.

A skála másik végén többek között azok a **nagy teljesítményű**, biztonsággal örzött, „**közüzemi**” jellegű és szolgálatot nyújtó számítógéppontok gépei vannak, amelyek üzemeltetésénél mindenekelőtt két dolgot kell figyelembe venni:

- a nagyszámú, a gépet működtetőket szempontjából lényegében ismeretlen felhasználó گردگیének kiszolgálását. Mindegyik érezze úgy, a gép csak neki dolgozik, a gép választásai legyenek rövidek stb.;
- a gépet minél jobban kihasználják, minél több pénz termeljen stb.

E két szélsőség között (mellett?) a tipikus környezetek széles választékát találjuk. Néhányat — most már csak jelzésszerűen — még felsorolunk:

- más gépekbe **beépítésre szánt** (vezérlő) számítógépek, nagy teljesítményű, de nem „sook kis és tipikusan ismeretlen felhasználónak”, hanem egy-egy nagyvállalatnak dolgozó (például ún. **batch-orientált**) gépek környezete;

- egy-egy kisebb szervezeti egység (például egy hivatal, egy vállalat egy-egy **osztálya**, **főosztálya**) részére dolgozó gépek környezete, ahol egymást jól ismerő, egymás munkájának eredményeire támaszkodó, azokra gondosan vigyázó munkatársak dolgoznak.

A gép és az elvégzendő munka

A számítógépek és környezetük kapcsolata — a gépek körüli **munkaszervezés** — a számítástechnika egyik legkevésbé kiforrott területe. Ez tükröződik az elnevezések, a fogalmak bizonytalanságában, a problémákör strukturalásának gyakori módosulásában, sokféleségében. Emiatt itt még nehezebb a tudós továbbadása, mint például a programozás technikájában vagy az elektronikus, digitális áramkörök tervezésében. Ennélfogva nehéz ma a számítógépes **rendszerlemezést**, rendszer-szervezést tanítani. Sokakban még az a kérdés is felmerül: van-e egyáltalán ilyen diszciplína?

A bevezetőben „könnyű kézzel” arra utaltunk, hogy a számítógépeket tipikusan **munkavégzésre** használják. Úgy látszik, a „munka” szó sajátos jelentést kapott a magyar számítástechnikai terminológiában: a számítógépek, a számítógéppontok éves, heti, napi munkavégzésének folyamataiban **munkának** hívjuk azt a jól definiált „tevékenység-, feladatcsomagot”, amelyet a gép — és mi, akik a gépet használjuk — egy egységben látunk. A „munka” kifejezés tehát (az angol „job” fordításaként) egy — eredeti jelentésétől nem távolí, attól nem eltérő, de mégis csak *speciális* — szűkebb jelentést is nyert. Elnevezhetjük volna ezt az egységben látott valamit „feladatnak” is, ahogy az angol terminológiában is van némi bizonytalanság a „**job**” és a „**task**” (a munka és a feladat) értelmezése között.

Mivel jellemezhető egy-egy ilyen számítógépes **munka**? A teljeség igénye nélkül mondhatjuk, hogy

- van *megrendelője*, aki többek között jogosult a gép szolgáltatásainak igénybevételeire stb.;
- van *erőforrásigénye*, például bizonyos perifériákat, adathordozókat, **programokat** stb. igényel;
- esetenként meg van határozva a munka *felső költséghatára*, töltetesen már csak azért is, hogy a munka végrehajtásához igénye programban benne maradt hibák — ún. végtelen ciklusok — miatt a nem igazán gondos megrendelőnek ne kelljen „ingét-gatyáját” ráfizetnie a munkára;

- a megrendelőnek meg kell adnia, hogy a munkáját elvégző gép honnan kapja a bemeneti adatokat, hová adja a gép az eredményt stb.;
- van-e a megrendelt munkának *prioritása* a többi munkához képest;

- kell-e, és ha igen, hova kell elszállítani például a munka eredményként létrejövő nyomtatványokat;

- milyen a munka titkossági fokozata, tehát nyílt-e, bizalmas-e, „szolgálati használatra” minősítésű-e stb.

Kezddőnek a legtöbb gondot ebben az összefüggésben az szokta okozni, hogy nehezen fogják fel: a **programok is erőforrások**. Ennek az az oka, hogy a számítástechnikával ismerkedők többsége a mai PC-s világban vagy „*játszik*” a géppel, vagy programokat „*fabrikál*”. Ezért számításmunka tipikusan csak **egy** program látszik: amivel játszanak, vagy amit éppen készítenek. A profi számítógéppontokban dolgozók jellemző szemlélete szerint a „**gép**” feldolgozó egységek, csatornák, perifériák, futásra kész **programok**, tehát a legkülönbözőbb erőforrások összessége. A programok tipikusan úgy vannak megírva, hogy azokat „**egy időben**” akár több felhasználó is használhatja. Olyanok ezek, mint valamilyen közös konyha falára felfüggesztett receptek, amelyeket egyszerre több szakács is követhet úgy, hogy nem is kell egyszerre elkezdeniük a főzést. Sőt: a szakácsok cserélődhetnek a konyhában — a receptek maradhatnak. A konyha előtt áll *egy konyhafőnök* is, aki mielőtt beengedné a felelőkény szakácsot a közös konyhába, tisztázza, van-e még szabad tűzhely, ki vannak-e fűggesztve a szakácsnak szükséges receptek (közösen használható programok). Ha nincsenek, akkor mielőtt beengedné a szakácsot, gondoskodik arról, hogy a pincéből (a háttértárról) felhozzák neki a receptet stb. E kissé gyermeketnek tűnő hasonlatban a „**munka**” egy-egy étel elkészítése, a falakra kifüggesztett receptek pedig az operatív tábla beolvasott programok, a munkák elvégzésének fontos „**erőforrásai**”.

Programok, mint erőforrások

Az előbbiekből talán kiderült, hogy a profi — pénzes — számítástechnika kicsit másképpen tekint a programokra, mint az érdeklődő, a gépekkel most ismerkedő laikus. Ha el akarjuk érni, hogy ezeket a számítástechnikai „recepteket” több számítástechnikai „szakács” is egyidejűleg felhasználhassa, akkor — bármennyire is büszkének vagyunk Neumann János felismerésére — olyan programokat kell írunk, amelyek futás közben nem **módosítják magukat**. Ilyen *közösen használt* programokat okos és szigorú *önkorlátozással* különválasztjuk azokról az adatokról, melyeket azok „*megmunkálnak*”. A „megmunkálendő”, feldolgozandó adatok **munkáról munkára** változhatnak. A **munka** megrendelője a „rendelés feladásakor” külön meg kell, hogy adja, honnan kell majd venni a feldolgozandó adatokat. A kezdők kis, személyi számítógépeikhez szánt programjaikba önmaguk építik be ezt az információt!

A **munka** — mint szervezési egység — fogalmát tehát akkor tudjuk igazán jól megérteni, ha megértettük, hogy nagy, drága, sok felhasználóval kiszolgálására szánt gépeinkbe a *közös felhasználásra szánt* programokat **más elvek alapján** készítik, mint a kezdők szokták BASIC programjait készíteni.

Ezek a profi programok olyanok, hogy többek között nem bennük (a programban magában) van megköthető, pontosan milyen perifériáról — terminálokról, kártyaolvasóról, operátorok konzolról — származnak a feldolgozandó adatok; hova adja (ki) a gép az eredményt: például arra a terminálra, munkaalomlásra-e, amelyikről a feladatot „megrendelték” — elindították. Ez azt jelenti, hogy például a programban nem a futtatásokról

most meghívjuk Önöket, hogy elmélkedjünk közösen a számítógépeknek a környezetünkkel való kapcsolatáról. Gondoljuk át, hogyan is használják a gépeket munkavégzésre. A gép és a munka fogalma hagyományos gépeinknél is szorosan összefonódott. Amikor a számítógépek — a gépeknek ez az új fajtája — kapcsán beszélünk munkáról, akkor nem feltétlenül csak a fizikában meghonosodott munkafogalmat értjük rajta, hanem elsősorban azt a fajta munkát, amit ma még többségében rutin jellegű szellemi munkának tartunk. Például adatok célszerű rendezését, számítási feladatok megoldását stb.

ténylegesen felhasznált be- és kimeneti perifériák, állományok (fájlok) stb. nevei vannak beírva, hanem valamilyen „helyettesítő nevek”, melyek a program futásakor összekapcsolódnak a megrendelt munka megrendelője által megadott, specifikált perifériaállomány- stb. nevekkal.

Számítógépek — gazdaságos működés

Írtuk, ha egy olcsó PC „gazdára vár”, nem baj, hadd várjon, a „gazda” ideje a drágább. Nagy gépekkel nem ez a helyzet. Ezeknek egy-egy órányi ideje akkor is akár több tízezer forintba kerülhet, ha csak ott állnak a gépteremben. A cél tehát: minél jobban kihasználni a gépet, minél több munkát elvégeztetni rajta.

A gép éves, havi, heti, napi stb. „tevékenysége” (használok ezt a szót, mert a munkát már lefoglaltuk a számítástechnikában másra) munkák — felhasználói megrendelések — tömegéből áll. A jellegzetes alaphelyzetek:

- a számítóközpont „bejártatánál” igen sok megrendelés (munka) gyűlik össze. A gép úgy működik, mint egy „nagy daráló”. Beledobálják a megrendeléseket — a munkákat, job-okat — és a gép a munkák specifikációi, azaz erőforrásigényei, prioritásai, sorrendi kötöttségei stb. alapján MEGSZERVEZLI azok legelőszérűbb, folyamatos végrehajtását. A „daralót” állandóan „táplálják” az újabb és újabb befutó job-okkal, az eredményeket pedig folyamatosan a rendeltetési helyükre továbbítják;

- a számítóközpontot telefonon bárki bármikor elérheti, közölheti vele a „megrendelését”, s ennek során — ha kívánja — állandó (párbeszéd) kapcsolatban maradhat a géppel. Ma már vannak olyan fejlett rendszerek is, ahol egy-egy távoli terminálról egyszerre akár több munkát is meg lehet rendelni, és a gép nem keveri össze a különböző munkák be- és kimeneti adatait.

Természetesen nemcsak ez a két számítástechnikai „szinpadkép” (forgatókönyv, scenárió) képzelhető el, de a többit most már az olvasó képzeletére bizzuk és visszatérünk a bevezető gondolatra: miért is olyanok nagy teljesítményű gépeink, mint amilyenek.

Tipikus szervezési módok

A tipikus felhasználási körülményektől függően számos megoldás — stratégia, operációs rendszer, architektúra stb. — alakult ki a drága gépek jobb kihasználására. Hadd jegyezzük itt meg: a gépek jó kihasználása csak jó *összmunkával* érhető el. Eszerint:

- már a **hardver** tervezésekor figyelembe kell venni az üzemeltetési igényeket, mint a megszkizitási lehetőségeket, a tárfelosztást, az ún. „laptchnikát”, a tárvédelmet stb.;

- az **operációs rendszer** lehetővé kell hogy tegye a jó összekötést a hardver és a működtetők között;

- a gép egész **környezetének** (a géptermi rendnek, a számítóközpont adatrédéli rendjének, számlázási rendjének, a rendszer ún. „hangolásának” stb.) mind-mind összhangban kell lennie a felhasználók igényeivel, az operációs rendszerrel és a hardverben rejlő lehetőségekkel.

Ismét csak érzékeltetésül, milyen megoldásokra gondolunk.

- Tipikus — régebben általános — volt a **kötegel (batch)** jelle-

gű szervezés. Ennél a számítóközpontba érkező megrendeléseket — **munkákat, job-okat** — azonnal beolvastatják a gépbe. (A „köteg” — angolul: batch, németül: Stapel — elnevezés arra utal, hogy régebben a **munkák** leírását egy köteg, sorozat lyukkártyán rögzítette a megrendelő, aki ezt a **köteget** „nyújtotta be” a számítóközpontba.) Az operátor, illetőleg később már maga a gép az így benyújtott megrendeléseket osztályozta.

Az okos géptermi személyzet természetesen ügyelhet arra is, hogy a gépnek mindig legyen olyan viszonylag hosszú határidős (alacsony prioritású) háttér munkája is, amit a gép akkor vehet elő, amikor a sürgős feladatok száma csökkent.

Egészen más körülmények között működnek az ún. **beépített számítógépek** (angolul: embedded), amek a gép készülék, hogy más gépekbe „alkatrészként” beépítsék őket. Ezek tipikusan *rejtve maradnak* a felhasználó — a befogadó gép kezelője, a repülőgép pilótája stb. — előtt. Az ilyen gépek elsődleges felhasználójának leginkább a **másik** (a befogadó) gép tervezője (tehát egy másik *szakember*) tekinthetünk. Ő általában szívesen vállal egy kis tervezési többletnehézséget, ha jobb feladatmegoldást találhat. Neki általában nem okoz gondot a tisztán *műszaki* jellegű csatlakoztatási, illesztési, beszerelési előírás sem. Az ilyen beépített számítógépeknek rendszerint a befogadó gép folyamatait — általában többet egyszerre — kell figyelniük, és ha szükséges, azonnal be is kell avatkoznuk a figyelt folyamatokba.

Az információ az ilyen gépekhez jellegzetesen vezetékeken, elektro-mo jelekként érkezik. A közvetlen emberi, például irodai alkalmazásra szánt számítógépek szokásos perifériái a beépítésre szánt gépeknél szinte teljesen hiányoznak. Az ilyen gépek specifikációja arra van „kihegyezve”, hogy a gép gyorsan átváltszon egyik feladatról a másikra, hatékonyan tudjon „párhuzamosan”, egyszerre reagálni több feladatra, például tudjon több figyelést is ellátni. Ezért az ilyen gépek ún. megszkizitási mechanizmusra igen fejlett, operációs rendszerük „eseményvezérelt”, nem ún. „politika-stratégia” szerint működ.

A fejlődés irányai

Úgy tűnik, napjainkban a számítástechnika fejlődése kétpólusú:

- a személyi számítógépek ára és teljesítménye egyrészt szükségessé, másrészt lehetővé teszi a nagygépek körüli munkaszervezés egyre több elemének átvételét. Ugyanakkor — már csak a képzetlenebb felhasználók miatt is — ezeket a megoldásokat áttekinthetőbb, könnyebben megvalósítható rendszerbe kell foglalni, mint ami a nagygépeknél szokásos, másrészt a nagygépek körüli szakember-utánpótlás megköveteli a személyi számítógépek kapcsán bevezetett kényelem — jól struktúrált parancsnyelvek stb. — mind nagyobb részének átvételét. Ezek természetesen a korábbi nagy beruházások miatt csak lassan, fokozatosan vezethetők be.

Az operációs rendszerekkel párbeszédre alkalmas parancsnyelvek a „történelem folyamán” televíziószerűen fejlődtek, nem jól strukturáltak megtenulással, használatuk igen nehézkes. Itt furcsa jelenségnek találjuk magunkat szembe. A számítástechnikai szakemberek nemzedékei tanulták meg *fáradtságos munkával* ezeket az igen sokat tudó parancsnyelveket. A tudás tehát nem csupán a kenyérgereszt, hanem talán a konzervatívizmus forrásává is vált. Választóvonalá a laikusok és a beavatottak között. A személyi számítógépek korszak beköszöntével megjelentek a *laikus* felhasználók igényeivel sokkal jobban alkalmazkodó, de — kezdetben — sokkal kevesebbet tudó parancsnyelvek. Az a területen végbemenő fejlődés egyik érdekessége, hogy a személyi számítógépek fejlesztői — mint erre a korábbiakban már utaltunk — *sorra „újából felfedezik”* mindazokat a fontos tulajdonságokat, amelyeknek egy korszerű parancsnyelvben (operációs rendszerben) lenniük kell, és ezeket — de most már ismervé a nyelvekre vonatkozó áttitkos szabályokat is — a korábbiaknál sokkal jobban strukturáltan építik be a parancsnyelvekbe, az operációs rendszerekbe.

Befejezésül szeretnénk ismét felhívni olvasóink figyelmét arra, hogy amikor a számítógépeket ismerkednek, akkor ne csak a *gépekre* figyeljenek, hanem arra a *környezetre* is, amelyben a gép tipikusan működik. A gépek — az operációs rendszerek — működését, tulajdonságait, szolgáltatásait csak így érthetik meg igazán.

FEJLŐDÉS

Az előző alkalommal tárgyalt centrumérték-számításon kívül nagy jelentősége van a fejlődés értékének is a sakk-hadállások értékelésénél. A fejlődés szerepe a megnyitás kezdeti stádiumában a leglényegesebb. A megnyitásban kell a figuráknak kifejlődni. Az a cél, hogy az itt elfoglalt pozíciójukkal a közéjátékban minél aktívabb szerephez jussanak. A játszmában az van kedvezőbb helyzetben, akinek tempóelőnye van, mert ez a fél diktálja a játszma további menetét. Ezért nagyon fontos, hogy figuráinkkal mielőbb kifejlődjünk, és ennek érdekében a játszma elején ne lépünk többször azonos bábuval, hanem próbáljuk meg a teljes haderót mozgósítani.

A fejlődés és a korábban már tárgyalt mozgékonyági érték szorosan összefügg. Amelyik félnek fejlettebb a hadereje, annak a mozgékonyági tényezője is nagyobb. A fejlettség értékelésében sokféle tényezőt érdemes megvizsgálni.

Először is meg kell állapítani, hogy mely figurák hagyták el eredeti helyüket. Ennek figyelembevételével elsőként a huszárak és a futók kifejlésése, majd a sáncolás, és végül a nehéz tiszték aktivizálása a helyes sorrend.

A fejlettség másik, kevésbé ismert szempontja, hogy ameddig vezér van a táblán, addig a sáncolás létfontosságú. Ennek több oka van. A legtöbb megnyitásban a középvonalak jóval hamarabb kinyílnak, mint a szélsők. A centrumban bekövetkező cserék során ezek a vonalak sokszor viszonylag korán szabadabbá válnak a bástyák vagy a vezér számára. Így érthető, hogy a közepen maradt királyra ezeken a vonalakon több veszély leselkedik, mint sáncolás után, amikor az eredeti helyükön lévő f, g és h gyalogok — vagy hosszú sáncnál az a, b és c gyalogok — közvetlenül védik királyukat. A közepen rekedt király pedig könnyen okozhatja az egyensúly megbillenését is, ugyanis megszakítja a két szárny — a király- és a vezérszárny — közötti összeköttetést. Ekként tehát közlekedési akadályt hoz létre és kettéosztja saját haderejét, meggátolva annak ésszerű elhelyezését és átcsoportosítását. Ha viszont a vezérek lecserélődtek — legalábbis az ellenfél vezére nincs a táblán —, akkor már nem olyan alapvető a sáncolás.

David Levy, a kiváló sakkozó és sakkprogramkészítő a következő képletet használta programjában a fejlődés figyelembevételére:

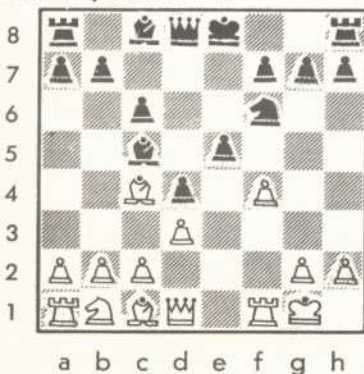
$$F = D/3 - U/4 - k \cdot C$$

ahol

- D = azoknak a könnyű tiszteknek a száma, amelyek nincsenek a helyükön;
- U = 0, ha a vezér még nem mozdult vagy leüttették, és a fejletlen könnyű és nehéz tiszték száma, ha a vezér már elmozdult, de még nem üttették le;
- C = 2, ha az ellenfél vezére még a táblán van, és 1 - P/4, ha az ellenfél vezére már nincs a táblán;
- P = a kölcsönösen leüttött bástyák és könnyű tiszték száma;
- k = 0, ha a játékos elsáncolt, és 1/3, ha a játékos elvesztette a vezérszárny sáncolás jogát, 2/3, ha a játékos elvesztette a királysárny sáncolás jogát, 1, ha a játékos mindkét oldalon való sáncolás jogát elvesztette.

Kasparov—Roizman, Minszk, 1978.

A 9. féllépés utáni állás



1. ábra

Dap Hartman elemző algoritmusában ezt a képletet a következőképpen alakította át:

$$F = 20 \cdot d - 15 \cdot u - (k \cdot c)$$

ahol

- d = a lépett könnyű tiszték száma;
- u = 0, ha a vezér nem lépett és nem üttették le, a fejletlen könnyű és nehéz tiszték száma, ha a vezér már elmozdult, de még nem üttették le;
- c = 8, ha az ellenfél vezére még a táblán van, 6 - p, ha az ellenfél vezére már nincs a táblán;
- p = a leüttött könnyű tiszték és nehéz tiszték száma;
- k = 0, ha a játékos elsáncolt, 3, ha a játékos nem sáncolt el, de mindkét oldali sáncolási jogát megvan, 5, ha elvesztette a vezérszárny sáncolás jogát, 10, ha elvesztette a királysárny sáncolás jogát, 15, ha mindkét oldali sáncolási jogát megszűnt.

Ha a megadott változók értékei a következő intervallumon kívülre esnek, akkor azt az adott változót az algoritmus automatikusan zérusnak tekintti.

$$0 \leq d \leq 4 \quad 0 \leq u \leq 7 \quad 0 \leq c \leq 8 \quad 0 \leq k \leq 15$$

Ezt figyelembe véve $F_{\min} = -210$ és $F_{\max} = 80$.

Dap Hartman képlete alapján a Kasparov—Roizman játszma tizennyolcadik féllépésében (lásd az 1. ábrát) a következő értékeket kapjuk világos és sötét számára:

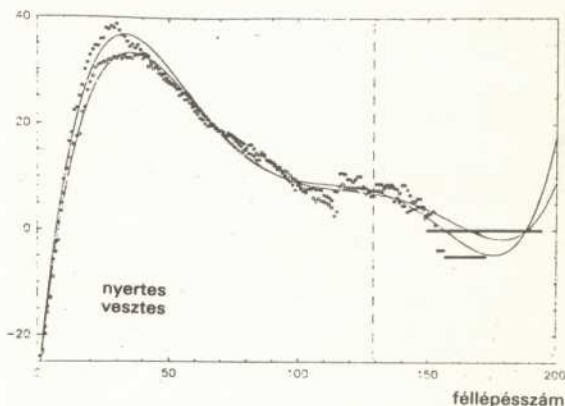
$$F(\text{világos}) = 20 \quad F(\text{sötét}) = 16$$

A 2. ábrán Dap Hartman elemző programjának eredményét láthatjuk, ahol a függvény ábrája jól szemlélteti a nyertes és a vesztes fejlődési értékét a féllépésszám függvényében.

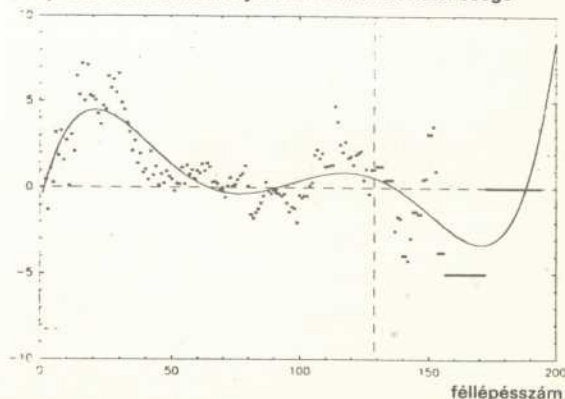
Kovács P. Attila

A nyertes és a vesztes fejlődési értéke

2/a ábra



A nyertes és a vesztes fejlődési értékének különbsége



2/b ábra



Ha IBM kompatibilis PC XT/AT gépre van szüksége vagy LAN hálózatot szeretne, készséggel állunk rugalmas lízing- vagy bérleti feltételeinkkel az ön segítségére is.

Ugyanilyen gépek beszerzésében is partnerei vagyunk. Minden IBM kompatibilis PC XT/AT gépre vagy hálózatra és perifériáira átalánydíjas karbantartási és javítási szerződés köthető.

CAD-szolgáltatásaink a műszaki élet minden területén segíthetik munkáját.

Betanítás, oktatás

Felhasználói szoftvereinkről szerénytelenség nélkül csak azt tudjuk leírni, hogy egyszerűen fantasztikusak.

Vagy léteznek, hogy ön ezt még nem tudja?



FÜTI Szervezéstechnikai Leányvállalat
1115 Budapest, Bánk bán u. 17/B

VÁRJUK ÖNT A VÁSÁRVÁROSBAN,



NJSZT

ahol 1989. március 17—22. között, a 25. pavilonban rendezzük az Országos Mikroszámítógépes Találkozót, a µ'89-et és a TOURCOMP kiállítást.



Tisztelet Adófizető Polgároink!

Az 1987. évi VI. törvény rendelkezik a személyi jövedelemadóról, amely az állampolgárok számára bejelentési, nyilvántartási, adóbevallási, adómegállapítási, adóelőlegfizetési, adófizetési, bizonylatmegőrzési, valamint vagyonynyilvántartás-tétellel kötelezettséget ír elő.

Az 1988. évi jövedelmek után legkésőbb 1989. március 20-ig kell az állampolgárok egy részének adóbevallást készítenie. Ennek segítségére szervezzük meg



nével új szolgáltatásunkat.

A szolgáltatás keretében

- megállapítjuk, hogy kell-e Önnek adóbevallást készítenie,
- feltárjuk az Önt megillető mentességeket és kedvezményeket,
- megvizsgáljuk az Ön számára legelőnyösebb változatokat,
- kérésre adatról kezelhető, rendszerezett nyilvántartást készítünk az Ön további személyes használatára
- az Ön által közből adottok és nyilatkozatok alapján elkészítjük az adóbevallást.

Öntől azt kérjük, hogy

- gondolja át az 1988. évben szerzett jövedelmét,
- gyűjtse össze az ezzel kapcsolatos dokumentumokat,
- ezekkel együtt keresse fel az ADÓGÁRAS IRODÁT.

Garantáljuk, hogy az elkészített adóbevallás — az Öntől kapott információk mértékéig — hibátlan.

Személyesleg fűződő információs jogait maradéktalanul tiszteljük.

Öntől semmilyen, Önt azonosító információt nem kérünk és nem őrzünk meg.

Ön időt, fáradságot és adót takarít meg.

Kérjük figyelemmel további tájékoztatónkat!

Adóbevallás elkészítése 750,- Ft
Ebből dírmegemly 1989. február 15-ig. 250,- Ft
Adóbevallás elkészítése nyugdíjraok részére 500,- Ft
A kiállított adóbevallás ellenőrzése 200,- Ft
Az önk az AFA-t tartalomozók.

A SZOLGÁLTATÓ IRODÁK CIMET
HIRDETÉSEKBE TESSZUK KOZZE

Az Olvasó írja

A rovatvezető bejegyzése miatt engem ért az a meg-tisztelő feladat, hogy válaszolhatok az olvasók kérdése-ire. Tallozzunk együtt az utóbbi hetek levelei között!

Mészár Zsolt, Orosháza

En a Commodore 64-es gépről szeret-nék levelezni. Cserébe egy érdekes fel-adat megoldását tartalmazó programlis-tát küldök.

```
5 REM == BORTONOR ==
10 DIM M(1000)
20 PRINT "♥"
40 FOR A=1 TO 1000
50 FOR B=1 TO 1000 STEP A
60 IF M(B)=0 THEN M(B)=1: GOTO 70
65 M(B)=0
70 NEXT B
80 NEXT A
90 FOR B=1 TO 1000
100 IF M(B)=1 THEN PRINT B, : S=S+1
105 NEXT B
110 PRINT: PRINT S; "RAB SZABAD"
120 END
```

A feladat: A király amnesztiát hirdet. A börtönben ezer rab van fogva tartva. A börtönőr ezerszer körbejár, és ellenőrzi a cellákat. Egyesével, kettesével, hármasával, négyesével stb. — egészen ezerig — vizsgálja a cellákat. Ha a cella zárva van, akkor kinyitja, ha pedig nyitva van, akkor bezárja. Amelyik rabnak a végén nyitva marad a cellája, az szabad.

Kérdés: Hány rab lesz szabad és hányas a cellájuk száma?

Ezenkívül a Newsroom nevű újságszerkesztő programmal kapcsolatos észrevételeimet közlöm. Bizonyára vannak olyanok, akiknek nincs botkormányuk, és a Newsroom programmal rajzot vagy fotót akarnak készíteni. A program leírása ugyan megtalálható az 1001/2 játék C64-re című könyvben, de, az ott leírtak ellenére, ha rajzolás közben a CTRL billentyű lenyomva tartjuk, a rajzolás éppen olyan kis léptékű, mintha botkormánnyal rajzoltuk volna.

Azt hiszem, ez a levél önmagáért beszél.

Pásztor Jánosné, Takácsi

1986-ban Thomson TO 7-70 számítógépet vásároltunk az NSZK-ban a lányunknak, akinek iskolai felkutatási keretében alkalmat nyílt foglalkozni a programozással. Mivel körüvesztetőként hivatalosan úton járunk kiinn, és kevés márkánk volt, játékprogramot nem tudtunk venni, és azóta sem tudunk beszerezni. A német nyelvű ismeretéből kiderül, hogy BASIC, LOGO, ASSEMBLY, FORTH nyelven ért a gép, grafika készíthető (van fényceruzája), négy oktávnyi zene szerezhető vele, ha lenne, aki segítené...

Ezúton kérjük azokat, akiknek Thomson TO 7-70 típusú gépük van, hogy jelentkezzenek a szerkesztőség címén. Bizunk abban, hogy akad olyan géptulajdonos, aki a fenti levélíróknak is tud segíteni.

Ifj. Fekete László, Budapest

Arra gondoltam, hogy alakítani kellene egy C64 levelező klubot. A tagok egymás között cserélhetnének tapasztalatokat, ötleteket és a saját maguk által megírt programjaikat. Mi erről a véleményük?

1989. január 1-jétől, DIG-MAG címmel, egyik barátommal újságot „indítunk”. Persze csak a haveri, esetleg iskolai közösgében. De tágabból is lehet szó. Az újságban főleg oktatósorozatok futnak majd, de programokat is tervezünk bele. Ez lehetne a levelező klub alapja is! Szeretném C64, C Plus/4 és ZX-Spectrum tulajdonosok segítségét kérni a programokra és ötletekre vonatkozóan. Akad azonban egy nagy problémánk! Hogyan szorosítsuk olcsón az újságunkat? Erre keressük most a választ.

Szeretném megtudni, hogyan lehetne értékesíteni a saját magunk írta programokat. Először egy karakterszerkesztőről lenne szó.

Nem értem, hogy a „Semmi sem lehetetlen!” című cikkükben irta boltot miért nem szüntettek meg egyszerűen. Ez a legnagyobb piszkosság, ami ott folyik! (Elnézést a kifejezésért.)

Mostanában a C64-hez egy új adatátviteli „rendszer” tervezek, s ehhez kell építenem egy C64 interfészt és egy mikrogepet. Hol tudom beszerezni a hozzá való alkatrészeket? A következőkellenének: 6510, 6526, 2764 (EPROM) és MK4802 (statikus RAM).

Miért szentelnek minden számból 3-4 oldalt az Enterprise-nak? Nem lehetne ugyanazt az Amiga 500-at megtenni?

A fent említett mikrogepekhez kisebb IC-k is kellenek. Ezért IC katalógust keresek, de nem találok.

A levelező klubot nagyon jó ötletnek tartjuk. A HCC-n belül már próbálkoztak egy ehhez hasonlóval, de nem lett belőle semmi, mert nem volt, aki vállalja a sok adminisztrációt. Mint dr. Simonyi Endre, a HCC Klub elnöke elmondta, nem adták fel a reményt, így ha azt valaki — akár ifjú olvasónk — elvállalja, a klubhálózatunk belül is működhetne a levelező klub.

Az újságkiadáshoz lapkiadási engedélyre van szükség. Olcsó sokszorosításra semmi esélyt nem látunk, mivel a nyomdaköltségek egyre magasabbak.

Saját készítésű programok értékesítését például a Novotrade is vállalja, ha az a program kiváló színvonalú, és a maga területén újdonságnak számít.

Karakterszerkesztő program elég sok van forgalomban, nem tudjuk, hogy az önk programja mennyivel tud többet, mint az eddig megjelentek. Ha kíváncsi a szakvéleményünkre, akkor küldje be a programot címünkre.

„Semmi sem lehetetlen” című cikkünkre nagyon sok olvasó reagált. Ezért itt ragadjuk meg az alkalmat, hogy elmondjuk, cikkünk megjelenésének hatására a bolt beszüntette a másolt programok árusítását.

Az alkatrészek beszerzése nem lesz egyszerű feladat. Elvéve kaphatók kiskereskedőknél vagy klubokban.

Hogy miért szentelünk négy oldalt az Enterprise-nak? Ez is visszatérő kérdés. Pedig egyszerű a magyarázat: mintegy 17 ezer van belőle az országban. Míg az Amiga 500-ból sokkal kevesebb. Mi is nagyon jó gépek tartjuk, de sajnos az ára miatt nem hiszünk, hogy igazán elterjedne hazánkban. Mindenesetre mi nem zárkózunk el az Amiga 500-za foglalkozó cikkeket vagy programok megjelenésétől sem. Az most már csak a géptulajdonosoktól függ, hogy kapunk-e tőlük használható anyagot.

IC katalógust sajnos nem lehet vásárolni, de folyamatosan jelennek meg katalógusok a Magyar Elektronika és a Rádiótechnika című lapokban. Az Elektromodul könyvtárban pedig valószínűleg megtalálja (Bp. XIII., Victor Hugo u. 11-15.)

Egy pécsi olvasónk

Felháborodással ragadtam tollat, és írtam meg ezt a levelet önöknek! Birkatúrelmem lejár — utoljára másfél (1) évvel ezelőtt írtam a szerkesztőségnek, hogy az 1986-ban kiírt játékprogram-pályázatra küldött két kazettámat szeretném vissza-kapni. Ez alatt az idő alatt módomban volt észrevenni, hogy maguk le sem ... engem. Se kazettáit, se választ nem kaptam levelemre.

Azért elgondolkodtató, hogy egy magára valamicskét adó lap ilyet megenged magának. Jó lenne, ha előkerítenék a kazettáimat!

Levélírónk nem kívánta, hogy felháborodását közzéadjuk. Anonimitásának megtartásával mi ezt mégis megteesszük, mert lám, sajnos nemcsak dicsérető levelek duzzasztják postáinkat, hanem előfordul ilyen is. Egyébként kazettáit időközben visszaküldtük.

De vigasztalódjunk kissé a következő levéllel:

Patek Alajos, Budapest

En csak 1988 eleje óta vagyok olvasója a Mikroszámítógép Magazinnak, de minden számban találtam valamit, ami érdekes volt számomra. Mint lapjuk decemberi száma mutatja, immár öt éves megjelenési évfordulójukat ünneplik. Ez alkalomból szeretném ünnepi hangulatukat azzal is emelni, hogy kifejezem: ez a lap igen jó példája a magazinoknak, amelyekből — sajnos — elég kevés van. Mondhatnám, hogy önök valóságos missziót teljesítenek a mikroszámítógépes amatőrök nem csekély tábora érdekében. Engem különösen az Enterprise-cikkeik érintenek, mert ilyen gépem van, de igen sok értékes információkat kapok más gépekkel foglalkozó leírásaikból is. Nem akarok túl sokat írni: a Mikroszámítógép Magazin kiválóan szerkesztett folyóirat. Jubileumuk alkalmából legyen önök az olvasók elismerő hangja is.

Szerkesztőségünk minden munkatársra névelen megköszönöm a jókívánásokat. Várjuk további leveleiket.

Tamásné Lakó Erika

Mit kell tudnunk az adóról?

Több olvasónk érdeklődött, hogy a különféle számítástechnikai szolgáltatásokra milyen adószabályok vonatkoznak. Ezért kérdéseinket az Adó- és Pénzügyi Ellenőrzési Hivatalhoz fordultunk, ahonnan az alábbi tájékoztatást kaptuk:

1. Válaszboríték ellenében való ingyenes tájékoztató küldésének adóvontata nincs, mert a hirdető bevételhez nem jutott.

2. Különbő programok adásvétele esetén az áfa adókulcs 25% (Termékszám: 97. Szoftvertermékek, 98. Számítástechnikai adattermékek).

3. Számítástechnikai szolgáltatások esetén:

a) Adómentes a magánszemély által nyújtott szerzői jogi védelem alatt álló szolgáltatás (1987. évi V. tv. 1/A melléklet 16. pont)

b) Ezen belül 0%-os kulcs alá esnek azok a számítástechnikai szolgáltatások, melyek az információellátással, kutatással, kísérleti fejlesztéssel kapcsolatosak. Ideértve statisztikai besorolástól függetlenül azokat a szá-

mitástechnikai szoftver szolgáltatásokat is, melyekért szerzői jogdíjat fizetnek (SZJT 922-71).

4. 25%-os adókulcs alá esnek az alábbi számítástechnika-alkalmazási szolgáltatások. Pl:

SZJT 710-11 Szoftverszolgáltatás

710-12 Szoftveres szellemi bér-munka

710-13 Szoftvertermékek átalakítása

710-21 Számítástechnikai eszközök szolgáltatása

710-24 Gépi adatfeldolgozási tanácsadás, véleményezés

710-31 Információs szolgáltatás

710-33 Adatállományok módosítása, aktualizálása

710-9 Egyéb adatfeldolgozási, számítástechnika-alkalmazási szolgáltatás.

Személyi jövedelemadó tekintetében az adókötelezettség kiterjed minden belföld-

ről származó jövedelemre (1987. évi VI. tv. SZJA 3. §).

Ezeknél a jövedelemnél a megszerzés érdekében felmerült és elismert költségekkel kell csökkenteni a bevételt (SZJA 13. §). Amennyiben a magánszemély saját találmányát értékesíti (átruházás, hasznosítás), úgy 500 000 Ft bevételig a bevétel 35%-át kell számításta venni (SZJA 6. § (1.) bekezdés). Ha a jövedelem a ténylegesen felmerült és igazolt költségek alapján a fentieknél kisebb, a jövedelmet az SZJA 13. § (2.) bekezdés alapján kell meghatározni (SZJA 6. § (2.) bekezdés).

Az adóalap megállapítása során mind az értékesítésből, mind a szolgáltatásból származó jövedelmeket, minden más jövedelemmel (kivéve az értékpapirokból, kis összegű kifizetésekből, külföldről származó bevételből és az 1987. december 31. előtt, a kifizetőt kötött szerződésből származó jövedelmeket) össze kell vonni és az adót az így összevont jövedelem után kell megállapítani.

ADOK—VESZÉK —CSERÉLEK

Ebben a rovatban rövid, szöveges, a mikroszámítógépekkel kapcsolatos hírdetéseket közlünk. A díjszabás: közületeknek gépelt soronként (60 karakter) 100,- Ft, magánszemélyeknek az első sor 50,- Ft, minden további sor 20,- Ft. Az N3SZT tagjainak az első három sor ingyenes. Hirdetéseiket a szerkesztőség címére várjuk.

ADOK

Atari program idegen nyelvek tanulásának megkönnyítéséhez. Válaszboríték ellenében ingyenes tájékoztató! Dr. Gerő László, Szeged, Budapesti krt. 4/b. VIII/31. 6723

Atari 800 XL (64 k) típusú alig használt számítógép programozási könyvvel eladó. Árajánlatokat kérek: Ujhelyi Ferenc, Gyöngyöstarján, Pozsonyi u. 12. 3036

C16 (új) magnóval, joystickkal, bő irodalommal és 100 db játékkal 15 000 Ft-ért eladó. Érdeklődni lehet levélben: Tóth Attila, Csongrád, Hársfa u. 61. IV/12. 6640

Commodore 16 mamóriabővítővel (64 k), magnóval, 2 db joystickkal, programokkal eladó. Érdeklődni: 17-19 óra között. Nagy Béla, Budapest, XXI. ker. Puli sétány 16. Fsz. 2.

C64-es floppyval, lemezekkel, datasettel, kazzettákkal, könyvekkel eladó. Sertény János, Békéscsámon, Vöröskatonák u. 97. 5946

C64C + 1541C kétszáz programmal, szakirodalommal eladó 40 000 Ft-ért. Érdeklődni: Kotroczó Balázs 655-110

C64-re új programokat és leírásokat adok, olcsón. Danka Ivadár, Nyírbátor, Váczi M. u. 9. Fsz. 2. 4300

Enterprise programokat kiváló minőségben adok, veszékek, cserélek. Benkő Sándor, Nyíregyháza, Rákóczi u. 23. I/5. 4400

EPSON kompatibilis mátrixprinter (Centronics); valamint Enterprise programokkal, dokumentációval eladó. Göbölös László, Kalocsa, Bársony u. 13. 6300

Junoszty televíziót videomonitorrá alakítok (kép+hang), az eredeti funkciók megtartása mellett. Minden videokamerát rendelkező számítógéphez alkalmazható, C128 esetén 80 oszlopos megjelenítésre is alkalmas. Páll Miklós, Budapest, Rákóczi F. u. 345. K 7-10 C lépcs. IV/9. 1214

Primo SSD-hez 32 k-s komplett felhasználói programcsomag égetése. Vidékre utárvétel. Előzetes megbeszélés: Horváth László 583-544 délelőtt

Legújabb Spectrum programok eladók! Katalógusért küldjön válaszborítékot! Frim András, Pécs, Zrínyi u. 1. 7621

ZX-Spectrum (48 k) gépetem több száz programmal, kézikönyvekkel, esetleg magnóval eladom. Irányár 16 000 Ft. Illés Tamás, Nyíregyháza, Fészek u. 200. 4400

ZX-Spectrum (48 k) 30 db játékprogrammal eladó. Irányár: 9000 Ft. Kovács Zoltán, Fehérgyarmat, Május 14. tér 17. 4900

ZX-Spectrum (48 k) + Kepston illesztő + botkormány + programok (100 db) + könyvek 10 000 Ft-ért eladó.

Enterprise (128 k) + magnó + programok + könyvek 18 000 Ft-ért eladó. Cím: Vágó István, Szigetszentmiklós, Jókai u. 27/B1. IV/17. 2310

ZX-Spectrum extrákkal (80 k, beépített MONITOR prg., RESET) 15 000 Ft. Adok hozzá több száz programot és könyveket. SPECCY DSZ 4s0 párhuzamos floppy illesztő 10 000 Ft. Lakatos Péter, Miskolc, Klapka Gy. 36. 3524, Tel.: 64-417

VESZÉK

C64-re keresek AMICA PAINT-hoz vagy ART STUDIO V1.2b-hoz screenloader-t (képtöltő prg.) lehetőleg BASIC nyelv-

ven. Reiner Péter, Pécs, Kertész u. 15/1. 7632. Tel.: 26-984

ZX-Spectrum 48-hoz billentyűzetet (főlia) vagy javítóceget keresek. Tapasztó Szabolcs, Kenderes, Lenin u. 98. 5331

CSERÉLEK

C16, Plus/4-re mindenféle programot cserélek. Listát kérek! Lakatos Balázs, Gyöngyös, Nagpuszta u. 14. 3200

Commodore 16-es, 64-es, 128-as géppel rendelkezőkkel kapcsolatban keresek, hétézer programom van. (Amigához 700 lemeznyi programom van.) Gyermán Sándor 23000 Zrenjanin Rade Koncara 23/V. Jugoszlávia

Plus/4-re színvonalas programokat (játék, felhasználói, zene, grafika) cserélek kazettán. Listát kérek! Rigó Zoltán, Kazincbarcika, Gyulai P. u. 3. I/1. 3700

C64 programokat cserélek kazettán. Listát kérek! Goda Gábor, Jászberény, Kosuth L. u. 39/A. 5100

Enterprise programok és tapasztalatcsere, fejlesztéshez, CP/M adaptáláshoz partnereket keresek. Horváth György, Budapest, Havanna u. 35. II/8. 1181. Tel.: 585-434 este

Enterprise programokat cserélek. Több mint 250 programom van. Minden levélre egy héten belül válaszolok. Listát kérek! Pavlovits Dávid, Szeged, Bécsei krt. 11/B. 6722

Spectrum programokat cserélek. Listát kérek! Ifj. Csutora László, Érd, Rábca u. 24. 2030

ZX-Spectrum programokat cserélek vagy eladok. Merényi Tamás, Pécs, Zsuzsanna u. 10. 7632

ADOK: új rádiós Walkmant (felvesz, lejászik) + értéklőnbözetet. KÉREK: ZX-Spectrumot és/vagy tartozékokat. Telefon 20 óráig: 156-868

Ez a rovatunk KODEX 2000 szövegszerkesztővel készült.

Plenge, Axel:
Grafika a Commodore 64-esen
 (Budapest, 1988.
 Data Becker — Novotrade,
 253 oldal. Ára: 240,— Ft)

A C64-es számítógép egyik fő erőssége a grafika, amit azonban a gyártó cég alaposan elrejtett. A kezdők csak csodálhatják a sok akciójátékban és kész programokban előforduló grafikákat, amelyek a megfelelő rutinnal — általában gépi nyelven — a készülő grafikai képességet teljes mértékben kihasználják. Ez a könyv lehetőséget kíván nyújtani minden C64-felhasználónak ahhoz, hogy saját programjaiban is kamatoztathassa számítógépe ez irányú tudását. A kötet három részből áll: hardver, programozási ismeretek és alkalmazás. Részletesen ír a szerző a VIC chip képernyővezérléséről, a kép szerkesztésének szabályairól, tisztázta a grafika fogalmát és azt, hogy mit jelent a jelkészlet, illetve a képernyőtároló eltolása. Vizsgálja a C64-es tárolójának felépítését és lehetőségeit. Megtámasztja, hogyan kell átállítani a grafikus üzemmódot és a legegyszerűbb ábrákat képernyőre rajzolni. Mindezekhez kényelmesen használható eszköztárcsoporthozat is közöl. Külön fejezet foglalkozik a korábban tanultak alkalmazásával és a különböző grafikai eljárások kombinációival.

Balogh Sándor — Berkes Jenő — Kovács László:
A számítógépes távközlés tematikai szolgáltatásai.
 Teletex, fakszimile, videotex
 (Budapest, 1988. LSI ATSZ,
 487 oldal. Ára: 463,— Ft.)

Az információ korát éljük. Ez egyben az információ általános elosztásának a kora is. A teleinformatika ma a világon az egyik legdinamikusabban fejlődő terület. Ez a fejlődés a távközlés és a számítástechnika integrálásának jegyében megy végbe. Új — az egész Földet behálózó — közmű, az információs közmű körvonalai kezdenek kibontakozni, melyet a szolgáltatások sokoldalúsága és integrációja jellemez.

A távbeszélőn, a telexen és más hagyományos távközlési szolgáltatásokon kívül az igényekhez jobban alkalmazkodó, mind sokoldalúbb, az információ összetett kezelését nyújtó, új, komplex távközlési szolgáltatások jelennek meg világsszerte. Egy részük nemzeti kereteken belül, más részük viszont nemzetközi szervezetek által szabványosított, az egész világra kiterjedő nyilvános szolgálat.

A modern távközlési technika elterjesztése fokozott felkészültséget igényel nemcsak a távközlési szolgáltatóktól és a velük együttműködő gyártói cégekől, de maguktól a felhasználóktól is. Ez alól hazánk sem kivétel. Ennek jegyében az új nyilvános távközlési szolgálatok, gyűjtőnéven a telematikai szolgálatok bevezetése a Magyar Postán napirenden van. A telematikai szolgálatok körébe elsősorban a teletex, a fakszimile és a videotex tartozik.

Ennek a könyvnek elsődleges célja, hogy az időben szereplő távközlési szolgáltatások általános ismertetésével, a felhasználónak nyújtott szolgáltatásai és lehetőségeik bemutatásával segítsen tisztázni a lehetséges felhasználási területeket. A szerző a könyvet elsősorban a szakmabelieknek szánja, s a klasszikus távközléssel foglalkozók közül azoknak, akiket érdekelne a távközlés új formái, a számítástechnikusok közül pedig azoknak, akik a számítógépes rendszerek ilyen alkalmazását kívánják megismerni. Azok is használnak foghatják a könyvet, akik valamilyen telematikai rendszerrel kapcsolatos implementációban vesznek részt.

Allaga Gyula:
Bűbájt
 (Budapest, 1988. SZÁMALK, 81 oldal.
 Ára: 120,— Ft.)

Tomí útmában egy számítógép belsejébe téved. Bűbájt tündér segít neki hazatalálni. Útközben megismerkesztési számítógépes utasításokkal találkoznak, beszélgetnek, ismerkednek, s hogy tovább juthassanak, meg fufangos feladatokat is kapnak. Mire Tomí végre hazaér — és fölébred — úgy érzi, hogy LET apóval. FOR hűsival, NEXT névvel, Felét Elekkel, Error ármerstrel, az Integer Oriáxssal és Gatu bohával, no meg a többiekkel együtt új barátot szerzett magának: a számítógépet. A színes, sok képpel illusztrált könyv elsősorban kisiskolásoknak készült.

Forgalomvizsgálat

Több mint tíz esztendő telt el azóta, hogy a közlekedés szakemberei felmérték az ország útjainak forgalmát. Egy évtized alatt azonban sokat változott a helyzet. Megnövekedett a járműpark, sok tekintetben megváltoztak a szállítási szokások. Időszzerűvé vált egy újabb felmérés arról, hogy a különböző települések között a jelenlegi körülmények között milyen a forgalom.

A Közlekedéstudományi Intézet leányvállalata, a Transinov vállalta, hogy elvégezi ezt a feladatot. A szervezet szakemberei tíz napon keresztül hatszáz benzinkútnál, a határállomásokon, a szállítással foglalkozó telephelyein végezték a felmérést. Több mint kétszáz ezer kérdőívet adtak át személy, és teherautó-vezetőknek, valamint a menetrend szerint közlekedő távolsági autóbuszok utasainak. A kérdésekről tudakozódtak, honnan, hova utaznak, az utakon milyen gyakorisággal közlekednek. A teherautósoknál a szállítmányok különbözősége iránt kérdezősködtek. A határállomásokon a külföldi rendszámú kocsik tulajdonosait arról kérdezték, milyen gyakran lépik át a határt.

Az adatokat más-más szempontok szerint számítógépen dolgozzák fel. Az eredmények támpontot adhatnak az országos úthálózat fejlesztési programját tervező szakembereknek arra, hogy hol, milyen utat építsenek.

Szimulációs geológia

A számítógépes szimuláció várhatóan forradalmasítja a hagyományos geológiát, különösen néhány olyan folyamat tanulmányozásában, mint a folyadékáramlások vagy azok a kémiai átalakulások, amelyek az üledékes medencék időbeli fejlődésére jellemzők. A szimulációk érthetőbbé tehetik számunkra az ércartalmú rétegek és mindenkiféle a szénhidrogén-tartalmú földtani képződmények kialakulását. Ha ehhez még hozzávesszük a rétegtani adatokat, akkor minden együtt van ahhoz, hogy teljes áttekinthetőséggel tanulmányozhassuk a geológiai korok üledékes medencéinek fejlődését. Már napjainkban is ezt a módszert alkalmazzák több földtani alakulat tanulmányozásakor, különösen a Mexikói-öbölnél, lévén ez az egész észak-amerikai kontinens szénhidrogénekben leggazdagabb tája.

Gépagysejtek

A japán Fujitsu kutató, olyan bioszámítógépet terveztek, amelyvel szimulálni lehet az emberi agy funkciót. A modell előkészíti az utat a gondolkodó számítógépekhez. A Fujitsu-modell kapacitása körülbelül 1000 agysejt teljesítményességének fele meg. Ezzel még távol áll ugyan az emberi agy képességétől, mivel az embernek mintegy 14 milliárd agysejtje van. Kapacitása azonban egyértelműen nagyobb, mint a hagyományos sejtprocesszor elvű számítógépeké, melyek teljesítménye csak körülbelül hat agysejtnek felel meg.

Beszéd szintetizálás

Személyi számítógépbe épített szintetizátort fejlesztettek ki a kijevi Gluskov Kibernetikai Intézet munkatársai. Ennek elsődleges újdonsága az az általuk kidolgozott matematikai rendszer, amely lehetővé teszi, hogy egy hang reprodukciójára a lehető legnagyobb műveltséget kelljen felhasználni. A hangképző szervek működésének utasítások ugyanis a műveletek száma meghaladja az ezret, az új mód szerben azonban ez csak hatvan. Ennek ellenére a beszéd minősége olyan, hogy könnyű összetéveszteni az emberi beszéddel. Az újdonság sok helyütt felhasználható. A repülőgépeken például szintetizátor jelentheti a műszerek mérésadatait, különösen a normálistoló eltérő értékeit, vagy esetleg a szerelőnek segít a futószalag működésének ellenőrzésében.

Közvélemény-kutatás

Még 1969-ben jött létre — az országban elsőként és sokáig egyetlen ilyen szintű intézményként — a Komárom megyei pártbizottság közvélemény-kutató csoportja. Feladata, szerepe kettős volt: egyrészt a politikai döntések előkészítéséhez friss információ szolgáltatása, másrészt lehetőségét adott a különböző részederék feltárására. A csoport fennállása óta 59 felmérést készített. Ezek többsége úgynevezett véleményvizsgálat volt, melynek célja az aktuális események kapcsán az érintettek nézetének, véleményének megismerése. Leggyakrabban a kérdőíves módszert alkalmazzák. Jelenleg mintegy száz aktivista segíti a csoport munkáját, zömmel pártmegbízottként végzik a kérdező-felvetelű munkát. A feldolgozás aztán a KSH megyei igazgatóságának számítógépen végzik. A csoport munkájának továbbfejlesztését a nyilvánosság kiteljesedése is szükségessé teszi, s napjainkban várható, hasonló feladatokkal, további közvélemény-kutató csoportok megalakulása is.

Alkatrészes és Amiga

Ha a szerveztek munkájáról, annak minőségéről esik szó, kulcskérdés a megítélés szempontjából az alkatrészes-utánpótlás. Sokan beszánkondnak azért, ha ilyen-olyan ok miatt lehetetlenülni vélik gépek javítását.

Megkérdőztük az Agroindustria Innovációs Vállalat Professional Szervizének vezetőit, milyen eszközökkel igyekeznek fenntartani a hozzájuk érkező gépek folyamatos szervizét. Mint megtudtuk, részben magánimportból, részben pedig hivatalos csatornánál, különböző kereskedő cégek közvetítésével jutnak hozzá a nélkülözhetetlen alkatrészekhez. Amikor viszont erre nincs mód, üzletfeleik megtartása érdekében azt is vállalják, hogy saját kockázatukra kész gépeket vásárolnak, s azokból „termelik ki” az alkatrészeket. Ilyen feltételek között köztöttek például kizárólagos garanciális javításra vonatkozó megállapodást az Enterprise gépekre, a Centrumtól követően a Novotrade-élt is. A Professional — bár munkatársainak létszámát és kirendeltségeinek számát jelentős mértékben csökkenteni kívánja, sőt az a pietyka is elterjedt, hogy „lehúzzák a rolót” — igéret szerint szolgáltatásainak körét tovább szélesíti: tervezzi az Amiga gépek szervizének kiépítését is.

A hetven főfoglalkozású munkatársat foglalkoztató szerviz, amely a legkülönfélébb gépek szervizével foglalkozik, a Commodore-tól a PC-ig, 1989-ben 70 millió forintos fix árbevételre tervez, szerződéses kapcsolata 1200 céggel van. Hívrása 48 órán belül az ország bármely pontján megjelennek, a közigazgatási határon belül ez az idő 24 órára csökken.

Adóbevallás készítése



AZ

ADÓGARAS

IRODA

GARANCIÁVAL ELKÉSZÍTI AZ ÖN ADÓBEVALLÁSÁT

Ha Ön a valószínűleg megleszlen lábai velünk:

- az elmúlt évben szerzett jövedelmait,
- a jövedelmek megszerzése érdekében felmerült költségeket (költésgeltározás esetén)

valójuk, hogy korrekt bevallást készítsünk az Ön részére, melyet kérésére dokumentációval egészítsünk ki.

Senkinek adatait nem adjuk ki, jövedelmi adatait díjtalanul elárthatjuk, ezeket semmilyen formában nem öntjük meg.

PRÓBÁLJA MEG! KÉSZÍTTESEN PRÓBASZÁMITÁST!

A szolgáltatásból személyi számítógépet használunk, amelynek programját adószámról ellenőrizték.

A szolgáltatás feltétele, hogy Ön hozza magával a szükséges adatokat (igazolásokat, számlákat stb.).

A szolgáltatást különösen ajánljuk annak, akinek adóbevallása bonyolult:

- egy jövedelmefajából több alattalammal szerzett jövedelmel,
- többféle jövedelme van,
- költségtérvezése van,
- külföldi jövedelme van,
- más országok adójából van jövedelme,
- megszerthető jövedelme van.

FIGYELEMI

Díjtalanul megvizsgáljuk, hogy teljes-e Önnek adóbevallást készítenie.

NE FELEDJEI

Az adóbevallás postázásának és az adóhatárokkal befizetésnek határideje

1999. március 20.

©SZÜV

Társvállalkozók: Kárgazdaság Egyetemi Kiszárványkezelő
Születés Dátaszámkezelési Csoport

Az adóbevallás elkészítéséhez szükséges nyomtatványok az iradókban díjtalanul kaphatók.

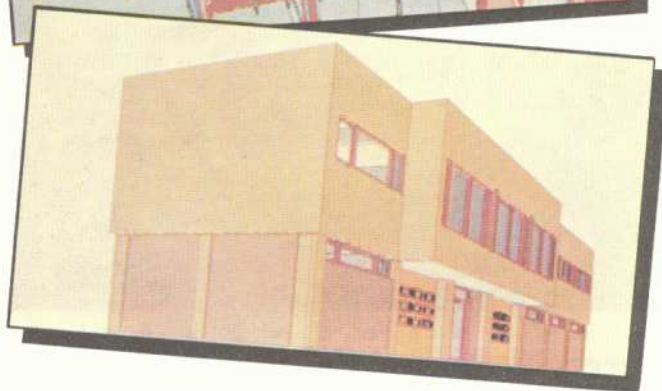
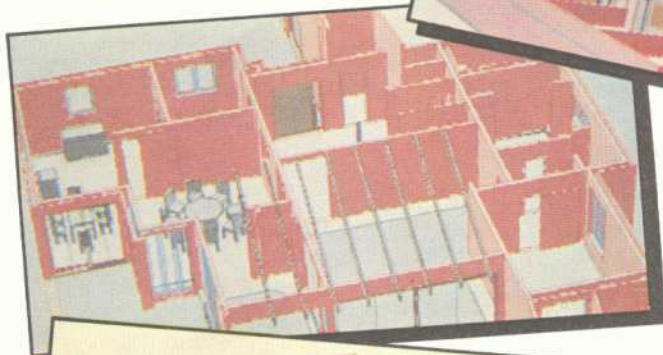


Graphisoft

ArchiCAD

1143 Budapest, Szobránc köz 10.
Telefon: 637-396
834-662

3 Dimenziós Modell-Orientált Építészeti Tervező Program



Az ArchiCAD programrendszerrel kis- és közepes méretű épületek gyors, interaktív grafikus tervezése, a tervek háromdimenziós megjelenítése, valamint az előzetes szükségletszámítás és a költségkalkuláció készíthető el.

A programcsomag felépítésében és logikájában alapvetően különbözik a szokásos két- és háromdimenziós, általános célú tervező rendszerektől. Az általános célú tervező programok stratégiája, hogy a grafikus tervezési processzus közös vonásait ragadják meg, hogy így lehetőleg minél szélesebb körben és minél általánosabban használható rendszereket alakítsanak ki.

Az ArchiCAD — bár természetesen tartalmazza a legfontosabb általános rajzi funkciók számítógépes megvalósítását — nem próbálja a felhasználók minél szélesebb körét kiszolgálni, hanem közvetlenül és kizárólag az építészeti tervezés sémáját követi. Ezzel a rendszerrel egy építészeti alaprajzon nem vonalakat rajzolunk, hanem „falakat”, amelyeket a falvastagságnak és léptéknek megfelelő távolságban levő két párhuzamos vonal reprezentál, ezek bármelyike falak kapcsolódásánál megszakad, és a közöttük levő területet a fal anyagának megfelelő szírozás tölti ki. Ha a program tudja, hogy nem vonalakat, hanem falakat rajzolunk, akkor mindezeket a funkciókat automatikusan elvégzi.