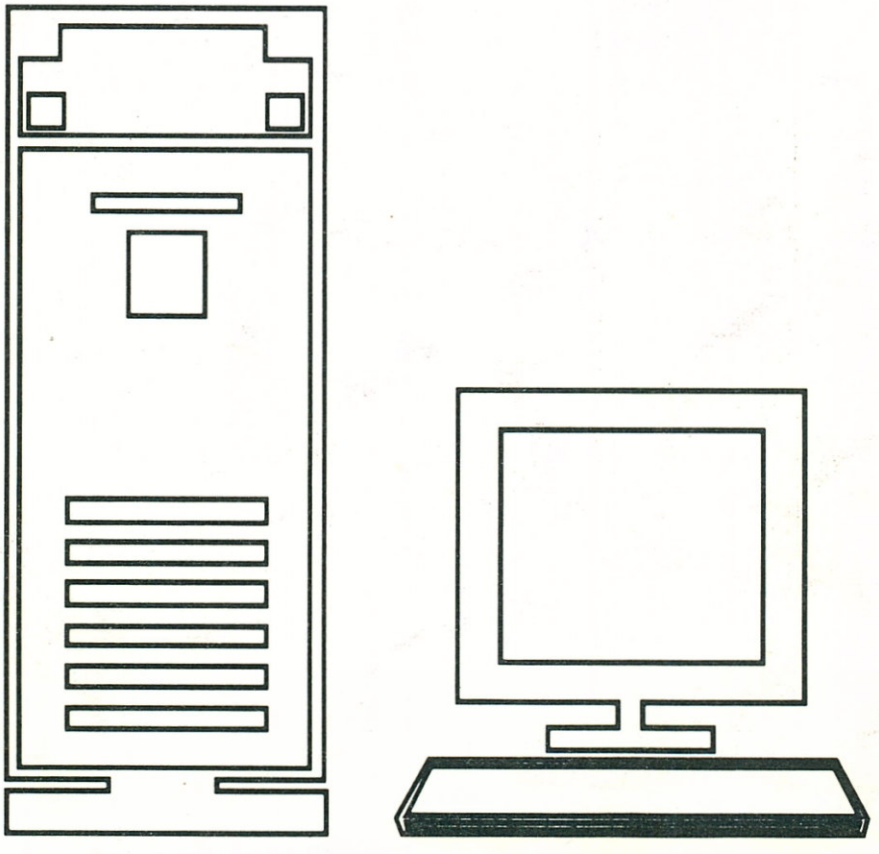
(Sz)Ámítástechnika

1.1



Kvassay Jenő MSZKI

Készítették:

Hubert Tibor tanár irányításával

Pfeiffer Ferenc tanár Mód Géza egyetemi hallgató

és a Kvassay Jenő MSZKI diákjai:

Hamza Réka	IV/a
Váradi Adrienn	III/d
Klinger Richárd	III/c
Puskás Zoltán	III/a
Csulik Máté	II/a
Varga Gergő	II/a
Breuer Péter	I/a
Dénes Balázs	I/a
Hamza Balázs	I/a

A könyv grafikáit Vályi Dorottya (III/a) és Breuer Péter (I/a) készítették.

A könyv készítői köszönetet mondanak:

Antal Ágotának
Lancsák Zoltánnak
Haas Péternek (III/c)
Irányi Zsoltnak
Kőszegi Lászlónak (III/c)

&

a **Kvassay J. MSZKI** vezetőségének
a **MIKROPO** cégnek
a **TIKETT KFT**-nek

akik a könyv létrejöttében és megjelenésében segítséget nyújtottak.

Lektorálta : Dr. Zsakó László

1992

Tartalom

Tisztelt Olvasó, Kedves Fiatal Barátaim!

Személyes élménnyel kell kezdenem. Elnézést kérek, de a személyes élmények nem tűrik az etikettet, előtolakodnak, jó, elmondom gyorsan. Én, szívem szerint, s az ember szíve szerint tud a legjobban ilyeneket, útáltam a számítógépet. Így, hosszú ú-val. Mégpedig azért, mert féltettem tőle a világot. A világon persze a magam kis világát értettem, illetve azt a humánnak nevezett parcellát, ahol a költészet mind szerényebben meghúzódni igyekszik. Az alkotás elgépiesítésétől féltem, féltettem magunkat. Hogy az okos gép a kreatív embert pusztítja ki belőlünk. A billentyűket nyomogató embert én akkor nem tartottam semmire.

Persze, elérkezett a pillanat, amikor nem volt mit tennem, megerőszakoltattam magamat egy számítógéppel. A dolog nem ilyen drasztikus, de érzékelteti rettentő ellenkezésemet. S most ott tartok, 37 évesen, hogy valamennyit már tudok arról a világról, amelytől olyannyira irtóztam. Először csak egy primitív szövegszerkesztő fekete-fehér mezője izgatott, azután a NC kék távlatai, majd mindenféle kis-nagy és rezidens programok következtek, s egy napon be akartam már tekinteni a gép belsejébe is. Mint a gyerek, öreg fejjel, de valahogy nagyon fiatalon, és ami a számomra is meglepő volt, nagyon ártatlanul. S belepillantottam.

Ez eddig, amit bevallhatok. S most olvasom ifjú barátaim könyvét, nem kevés meghatódottsággal és büszkeséggel, s amit nagyon fontos és nehéz egy tanárnak kimondani, pedig alapvető lenne ez a *viszonyban*, tisztelettel. S arról kéne értekeznem, mit tud, s mit nem tud ez a könyv. Nem az én tisztem, annyit el kell azonban mondanom, hogy többet tud, mint gondolnánk vézna külsejét szemügyre véve, s kevesebbet azok elvárásánál, akik ebből akarnak megtanulni programokat írni, szövegeket szerkeszteni. Nem tánciskola és nem zsonglőrködés. Az, aminek lennie kell: egy szorgalmas és sok szakértelemmel bíró kis közösség áttekintése, a néhány ember együtt mit lát fontosnak abból, amit számítástechnikának hívunk, némileg elrettentő terminológiával hívjuk így. Fontosnak gondolom, hogy kellő humorral, sőt, iróniával viszonyulnak nagyon is szent tárgyukhoz, így eleganciát kölcsönöznek a feldolgozás módjának. Rajzokkal oldanak, és általam már megfejthetetlen (vagy mert öreg vagyok és felejtek, vagy mert az én hajdani iskolámban nem tanítottak számítástechnikát) képletekkel, illetve annak feltűnő betűhalmazzal riasztanak, másokat csalogatnak. Használható ez a munka kézikönyvnek, de használható szemelgető délutáni füzetnek is. Ki tanulni akar, ki mazsolázgatni, tanuljon aki, és mazsolázzék, aki. Ennél több okosat nem igen tud elmondani egy outsider számítástechnikáról írott könyvről. Akkor meg miért írtam ezt a pár sort? Jeleztem már, büszkeségből, hogy olyan iskolában taníthattam a közelmúltig, ahol a diákok már nem csak tanulnak a könyvekből, de a tanulás könyveit maguk írják, szerkesztik - maguknak.

Zalán Tibor

Számítástechnikai alapismeretek

Bevezetés

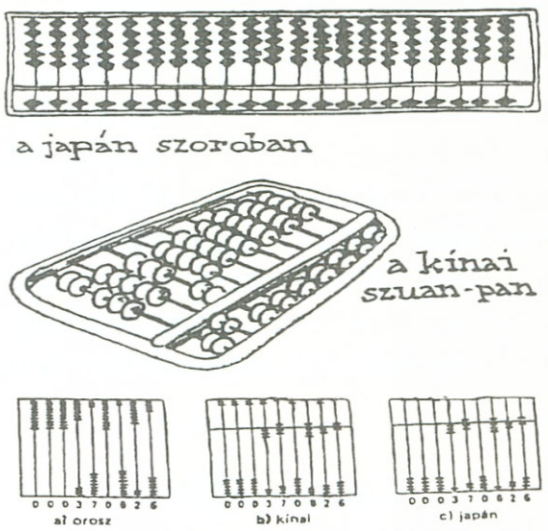
A számítástechnika tanulásához és az ezzel kapcsolatos irodalmak olvasásához (megértéséhez), s a számítógép használatához feltétlenül szükséges bizonyos szavak, fogalmak ismerete. Ezek hiányában a számítástechnika fogalma annyit jelentene, mint az étkezés az ételnevek ismerete nélkül. Ezek megértéséhez kellünk tehát mi, akik keserves munkával megtanulták, megértették, s most kigyűjtve, csokorba szedve egységessé tettük mindezt. Melyet bármikor, bárhonnan, bárkitől megtudhatnál, itt és most, ezekben a fejezetekben mindezt megtanulhatod. Ezek birtokában otthonosan mozoghatsz a számítástechnika birodalmában (igaz, hogy csak kispolgár leszel), de bátran kezedbe vehetsz bármilyen irományt a *"Birodalomból"*. Szót érthetsz az idősebb birodalmiakkal is.

A témaköröket különböző szempontok szerint csoportosítjuk. Ezekben szó lesz a számítógép történetéről és felépítéséről, az információ fogalmáról, a programnyelvekről, s ezek tulajdonságairól. Ezek után betekintést nyerhetsz a számítógéppel kapcsolatos szoftverek világába, s egy idő után már bátran használhatod őket. Megismerkedhetsz a gép működésével s matematikájával is.

A számítástechnika története

A számítástechnika őstörténete

Az ember számára az őskorban is megjelent a számolás fogalma. Számolta társait, elejtett állatait, használati tárgyait. Elkezdett műveleteket végezni, összeadott, kivont. Ez az idők folyamán tovább fejlődött, lásd: az ókori Egyiptom fejlett matematikája. S jelentős volt a középkorban is, mert e tudományoknak az egyház nem vetett gátat. Ekkor kezdtek foglalkozni a műveletek gépi végzésével. Ezek az első gépek mechanikus, fogaskerekekből összeállított eszközök voltak.



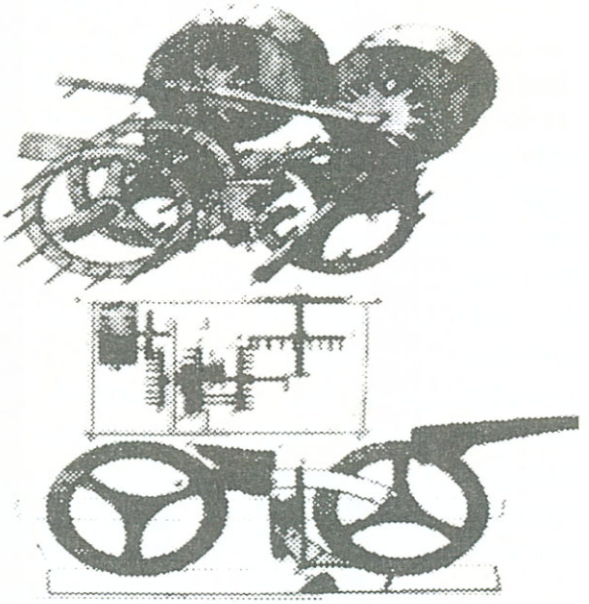
1.1 Ábra

A számítástechnika történetét az ókorban a golyós számítógéppel kezdjük. Ez a technikai eszköz segíti az embert a számítások elvégzéséhez.

Ezt az eszközt a mai napig használják az emberek. Bizonyos, hogy hallottál róla, hogy egyes általános iskolákban úgynevezett SZOROBAN-nal tanítják a tanulókat számolni, ami egy régen használt golyós számítógép.

Az 1.1 ábrán láthatod a különböző helyeken használt ilyen számolóeszközöket.

A számítástechnika fejlődésének következő nagyobb állomása az a mechanikus működésű számológép, amelyet PASCAL, a híres fizikus 20 éves korában (1652-ben) épített meg (itt láthatod az 1.2 ábrán), hogy apjának számítási munkáját megkönnyítse. Hiszen, ahogy a történet tartja, az apja magasrangú beosztása ellenére sem tudott jól számolni, de mivel új munkája sok számítás elvégzését kívánta meg, ezért a fiúi jámborság arra indította Pascalt, hogy apja terhén enyhítsen.



1.2 Ábra
Pascal gépének szerkezeti rajza

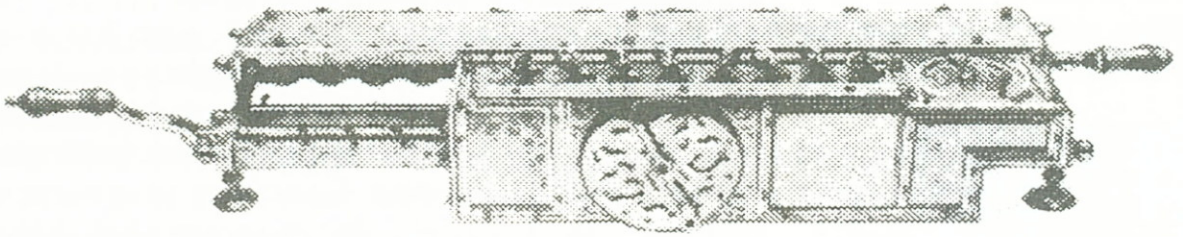
Ennek a gépnek a jelentőségét, amely csak összeadni és kivonni tudott, még Diderot is felismerte, amikor a gépet részletesen leírta a híres Enciklopédiában.

A következő nagy állomás a négy alapműveletes gép elkészítése, amely már összeadni, kivonni, szorozni és osztani is tudott. Ezt 30 évvel később készítette el a híres polihisztor LEIBNIZ, aki például a matematika és a fizika számos területén alkotott maradandót. Leibniz megfogalmazta munkájának eredményét:

Most már a csillagászoknak sem kell próbára tenni a türelmüket a rengeteg számolással.

"Kiváló emberekhez valóban nem méltó, hogy rabszolga módra órákat vesztegessenek el olyan számítások elvégzésével, amelyeket bárkire nyugodtan rá lehet bízni."

Leibniz az 1680-as években megépíti gépének harmadik példányát Nagy Péter számára, hogy a cár elküldje azt a kínai császárnak - bemutatva a keletieknek a Nyugat művészetét és iparát, fellendítve ezáltal a kelet-nyugati kereskedelmet.

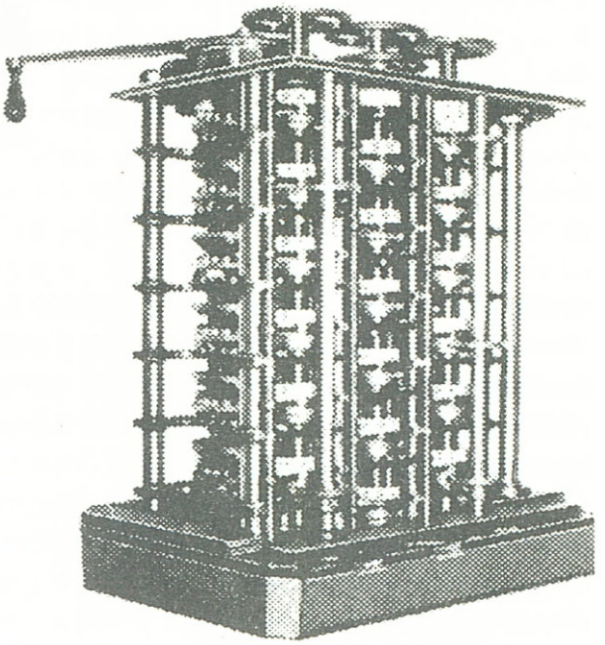


1.3 Ábra
Leibniz számológépe

Leibniz-nek a számítástechnika számos ágában vannak érdemei. A legnagyobb kettő mégis az , hogy egyrészt épített egy olyan gépet, amelynek utódja még a II. világháború idején, sőt az 1950-es években is a legfőbb és legáltalánosabban elterjedt számítási eszköznek bizonyult. Másrészt felismerte és megfogalmazta, hogy a számolást lehet automatizálni. Leibniz programját - az unalmas, de egyszerű feladatok automatizálását - tette magáévá Charles BABBAGE (1791-1871), aki talán a modern tudománytörténet egyik legkülönösebb alakja volt.

Tanulmányait a Cambridge-kollégiumban végzi, majd 26 éves korában londoni professzori állást kap. 1828-ban a Cambridge-i egyetem matematika professzorává

választják (Newton egykori katedrájára). Egyetlen előadást sem kell tartania, 1839-ben mégis lemond, hogy csak a számítástechnikának éljen. Az akkori számítástechnikai problémák legaktuálisabb kérdése a hajók számára készített, úgynevezett navigációs táblázatok rengeteg hibájának kiküszöbölése volt. A XVIII. század közepéig nem létezett a tengerészek számára olyan módszer, amelynek segítségével kiszámolhatták volna, hogy melyik hosszúsági fok alatt tartózkodnak. Így igen nagy gondot jelentett a navigáció, hogy nagyobb távolságokon úgy irányíthassák hajóikat, hogy azok az előre meghatározott kikötőbe érjenek. Később az égitestek mozgásainak alapján ilyen navigációs táblázatból tudta megállapítani egy hajóskapitány, hogy hogyan navigálja hajóját.



1.4 Ábra
Babbage differenciagépe

A matematikusok legfőbb segítségei azokban az időkben a táblázatok voltak.

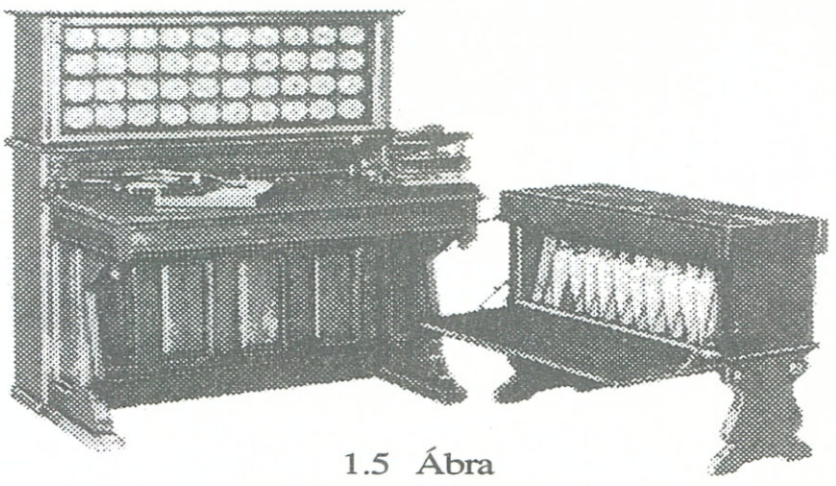
Egy statisztika a táblázatokra 1834-ből :

1	matematikus
140	függvénytáblázatában
40	táblázatból kiválasztva
3700	hibát találtak.

Ilyen táblázatokban rögzítették a tudósok a már kiszámított, vagy egyéb úton szerzett tapasztalataikat, hogy azokat mások is felhasználhassák.

Babbage a könnyen tévedő embert akarta helyettesíteni a "csalhatatlan" géppel, és ezért készítette el az úgynevezett differenciagépét, bizonyos sokat használt függvényértékek kiszámítására. 1833-ban félbehagyja ezen gép készítését (a gép egy működőképes modellje elkészül és ki is állítják 1882-ben), és hozzákezd az úgynevezett analitikus gép megtervezéséhez, amely a mai számítógép (tehát nem számológép) őse. Ez a gép mechanikus működésű, külső programozású digitális számítógép lett volna, de sajnos csak a tervek készültek el, a gép egészben soha nem épült meg.

A gép jelentőségének és Babbage munkásságának ismeretét nem kis mértékben Ada LOVELACE-nek köszönhetjük, aki a híres költő, BYRON leánya volt. A rendkívül tehetséges ifjú matematikus (Babbage) és Ada barátsága révén Ada Lovelace olyan jegyzeteket írt és programokat készített a géphez, amely e gép jelentőségét igazán bemutatja. A gép programozása és az adatok bevitele kívülről, az úgynevezett lyukkártyával történik. (A lyukkártya egy régi 1 dolláros nagyságú kártya, amelyen sorokban és oszlopokban számok vannak elhelyezve és ezen számok megfelelő kilyukasztása jelenti a kártyán az adatneveket vagy programrészeket.)



1.5 Ábra
Hollerith Lyukkártyás gépe

A gép két fő részből állt. Az úgynevezett tárolóból, ahova a változó adatokat visszük be; és a malomból, amelyben mindig csak azok az adatok vannak, amelyeken műveleteket kell végeznünk. A műveleteket is kártyán visszük be a gépbe, s ezen kártyák sorrendje jelenti a műveletek sorrendjét is.

Az analitikai gép ilyen módon teljesen általános jellegű.

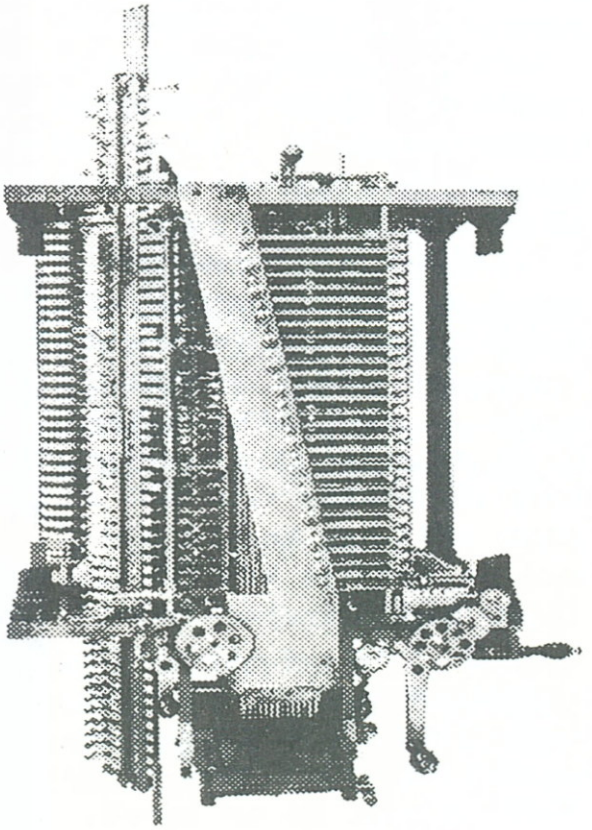
A számítás szabályait két kártyacsomag határozza meg, amit később újra és újra felhasználhatunk, akár egy másik kártyacsomagon.

A gép, mint Babbage valamennyi tervezett gépe, nem készült el. A tervek alapján Babbage nagyobbik fia a malom részt elkészítette.

Babbage, mint ember, nehezen összeférhető egyéniség volt, de érdeklődési köre igen széles körű, s a technika sok területén alkotott úttörő jellegű találmányokat. Csak felsorolásként álljon itt néhány példa: vasúti kocsikra szerelhető dinamométer, búvárharang, szemtükör, színpadi fényeffektusok.

Babbage gépének terve kb. 100 évig nem került az érdeklődés középpontjába, mert a tudomány érdeklődése az elméleti kérdések felé fordult. Ezek nem igényeltek gyors, nagytömegű számításokat. Az adatfeldolgozás terén volt nagyobb a fejlődés. Az Egyesült Államokban, az 1890-ben végzett népszámlálás eredményeinek kiértékelése olyan hosszú időt vett igénybe - több évet -, hogy mikorra az adatokat rögzíteni tudták, azok már rég elavultak, nem voltak hitelesek.

Ezen probléma megoldására, a nagytömegű adatok kezelésének gépesítésére való törekvés, a számítástechnika másik ágára hívta fel a figyelmet. Herman HOLLERITH lyukkártyás gépének bevezetése átütő sikert hozott, hiszen a feldolgozás az előbbiekhez képest hatod annyi időt vett csak igénybe.



1.6 Ábra
Babbage analitikus gépének malom része

Ez a Hollerith alapítja az IBM cég elődjét. S ekkor (1890) már 63 millió személy és 150 ezer polgári körzet adatait az így elkészített módon, a kártyák gépesített rendszerezésével lehetett feldolgozni. Ezen kártya az 1970-es évek elején is az egyik legelterjedtebb adattárolási módja volt a számítógépeknek.

Babbage gépének eredményét 90 évvel a terv megszületése után vették elő.

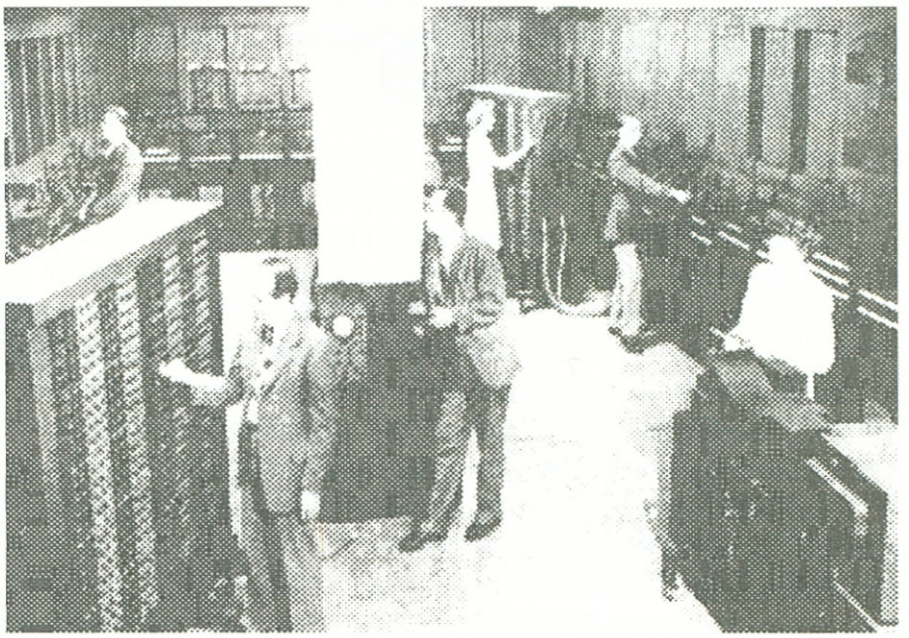
A ballisztika (ballista = hajítógép), a mechanika ezen speciális ága - amely tudományág a lövedék mozgását vizsgálja az ágyúcső elhagyásának pillanatától a célbaérkezés idejéig - számítási szükséglete, lett, a számítógépek fejlődésének elsődleges mozgatórugója a XX. század elején. Leegyszerűsítve azt számolja ki, hogy milyen vízszintes és függőleges szögben és milyen sebességgel kell az adott súlyú lövedéket kilőni ahhoz, hogy a megfelelő légköri viszonyok mellett a célba találjon. (Gondold meg, hogy mennyire nehezíti a számítást, ha ez a cél mozog!)

Akkoriban ezeket a számításokat táblázatokban rögzítették, és abból kikeresve adták meg a szükséges irányokat. Az 1940-es évek elejére az Egyesült Államok hadserege által követelt táblázatok készítése már olyan fontosságú és nagy feladattá

vált, hogy Ballisztikai Kutató Laboratóriumot (1938) hoztak létre, amely ezen táblázat adatainak minél gyorsabb kiszámítására végzett kutatásokat. Gondoljuk meg: ezek a számítások többnyire a négy alapműveletre vezethetők vissza, a szorzást és osztást természetesen minden gép lassabban hajtja végre, mint az összeadást és a kivonást. Ezen kívül kb. ugyanannyival lassabban az adatok bevitelét. Ezért mi szorzással számolunk most. Egy ember, gép segítsége nélkül kb. 5 perc alatt szoroz össze két 10 jegyű számot. Az 1930-40-es évek asztali mechanikus számológépének akkor a szorzáshoz 10-15 másodpercre volt szüksége, egy tipikus röppálya kiszámításához 750 szorzás kellett, ez 10-20 perc volt.

Egy tipikus gép tüzérségi táblázata 2000-4000 röppálya kiszámítását igényelte. Ha csak 3000 röppályát számítunk, akkor 1 hónapra van szükség egy ilyen táblázat kiszámításához. Ez a helyzet azért volt tarthatatlan, mert nemcsak nagyon sok munkát igényelt, de a harcban álló csapatok kezébe a számítások miatt késve lehetett egy már elkészített, új fegyvert adni.

Ennek a laboratóriumnak a kutatási terméke az ENIAC (Elektronic Numerical Integrator And Computer), amely az első elektronikus tárolt programozású univerzális számítógép volt. Ezt a gépet 1943-ban kezdték építeni és 1945-re készült el. 18000 elektroncsövet tartalmazott. Az elektroncső élettartama, mint az izzó élettartama, korlátozott. A legtöbb cső üzemeltetési ideje 3000 óra volt. Ez azt jelenti, ha feltételezzük, hogy egyenletesen égnek ki a csövek, akkor 1/6 óra alatt, vagyis 10 percenként ég ki egy cső. A 18000-ből meg kell keresni, hogy melyik égett ki, és ki kell cserélni. Ez a gép kb.1 Kbyte memóriájú volt. A felvett energiája 174 kWh, 180 m^2 alapterületű volt, s a fentiek ellenére is alkalmazták 1956-ig, lebontásáig. Hiszen például egy előbb említett táblázatot 30 nap helyett legfeljebb 9 óra alatt számított ki.



1.7 Ábra
ENIAC

A számítástechnika következő mérföldköve egy olyan személy nevéhez fűződik, akinek munkássága alig választható el az előbb említett ENIAC gép létrehozásától. Ez a személyiség a magyar származású és a budapesti Fasori Gimnáziumban végzett NEUMANN János (1903-1957). A gazdag bankárcsaládból származó Neumann 1921-től két egyetemet végez egyszerre, és 1927-ben lesz a berlini egyetem tanára. 1930-tól az Egyesült Államokban dolgozik, és 1944-től részt vesz az ENIAC megvalósításában. Az újabb gép az EDVAC tervezeténél felvázolja azokat az elveket, amelyek alapján a mai számítógépek működnek. Meghatározza a számítógép általános modelljét:

legyen egy aritmetikai egység, ami a számítási műveleteket végzi;

legyen egy vezérlőegység, amely a programot értelmezi;

legyen memória, ahol az utasítás és az adat együttesen tárolódik (ezzel ellentétben, mint már olvastad, az ENIAC-nak külső programtára volt);

legyen egy beviteli (INPUT) egység;
legyen egy kiviteli (OUTPUT) egység;
legyen külső tároló, amelyre csak ritkábban van szükség.

Az EDVAC az első binárisan működő belső tárolású, digitális számítógép. A számítógépek fejlődésének ebben az időszakában még sokféle elnevezésű gépet építettek a világ különböző részein, amelyekről most nem tudunk beszámolni, de ha érdekel javasoljuk, hogy olvasd el H.M. Goldsteinnek, Neumann János barátjának és munkatársának nagyon érdekes könyvét: A számítógép Pascaltól Neumannig című.

Az 1950-es évek közepétől a 80-as évek elejéig általánossággá vált a nagy számítógépek sorozatgyártása, amelyeknek egy-egy órai működése hazánkban például 6000-8000 Ft volt, hiszen nagy áramfelvétellel, különleges klímaberendezéssel működtek. A gép központi egységét hívták processzornak. 1972-ben jelentek meg először a mikrogépek központi egységei, a mikroprocesszorok. (INTEL)

A számítógép-generációk

A számítógépek felépítése és működése alapján a számítógépeket különböző generációkra osztjuk fel. Négy csoportra oszthatjuk őket. (Vannak még nulladik generációjú gépek - ezek az ún. elektromechanikus /relés/ gépek. Szakmai folyóiratokban, sőt időnként napilapokban is, olvashatunk az ötödik generációs gépekről. Ezek pontos jellemzőit a tervezők még nem teszik közzé, hiszen ezek a holnap gépei.)

Az első generációs gépek

Az első generációt alkotják azok a gépek, amelyekben az aritmetikai és a logikai egységben a műveletvégzéshez ***elektroncsöveket*** használtak fel. Hatalmas egységek voltak, drága volt az előállításuk. A programozása kizárólag gépi nyelven történt, teljesítőképessége néhány ezer művelet/másodperc volt (kis kapacitású, lassú).

A második generációs gépek

A második generációs számítógépekben megjelentek a diszkrét - egyedi - félvezetők (diódák, ***tranzisztorok***, ferrit gyűrűs tár /gyors, 1 gyűrű 1 bitet jelentett/), melyek következményeként a kapcsolási idő, a gépi méretek csökkentek. A belső tárolók kapacitása, illetve a számítási sebesség növekedett (**1** millió művelet másodpercenként), a gép megbízhatósága javult. Megjelentek az első szoftvertermékek, programnyelvek.

A harmadik generációs gépek

A harmadik generációs gépekben nagy integráltságú félvezetőket (***integrált áramköröket***) használtak, műveletvégzési sebességük **10-15** millió művelet másodpercenként. Ezek a gépek általában félvezető memóriát használnak. Lecsökken a hardver mérete. Megjelentek a korszerű operációs rendszerek.

A negyedik generációs gépek

A negyedik generációt a mikroprocesszor megjelenésétől számítjuk. A számítógép alapelemeit (integrált áramkör, tár) néhány processzor tartalmazza. Méretcsökkenés, nagy megbízhatóság jellemzi ezeket a gépeket. Ezek a ma számítógépei.

Az információ

Az információk az ember számára az érzékszerveivel érzékelhető ingerek. Ilyen inger a tapintás, a látás, a hallás stb. Az egyes érzékszervekre érkező jelek elkülöníthetők. És hogy a jelek közül melyik érkezik meg, ez adja az információt. A gépnek egyféle érzékszerve van és ezzel csak két jelet tud megkülönböztetni, (igazat-hamisat, igent-nemet, 1-0), s ez fizikailag 0 vagy 5V-ot jelent. Ennek a jelnek attól van információtartalma, hogy a gépben mikor és hol érzékelhető. Az ember agyában eltárolódó információnak a tárolási módját nem ismerjük, míg a gépben ez 1-1 kétállapotú tárolóhelyen történik. Ezek szervezése olyan, hogy 8-asával vannak egymáshoz rendelve, és ezen 8-as blokkokban történik a kezelésük. A kétállapotú információt ***BIT***-nek nevezzük, a 8 bitből álló egységet pedig ***BYTE***-nak.

Kódolás, kódrendszerek

Az emberi érintkezésben már a kezdetek kezdetén is nagy szerep jutott a kódnak és a kódrendszereknek. Először a hangképzés és a mondatok tartozott a kódnak nevezhető fogalmak közé.

Ezek önmagukban nem sokat jelentenek, de kialakultak olyan szabályok (kódrendszerek, megállapodások), melyekkel a mozdulatok és a hangok értelmet kaptak. Később más kódok és velük együtt más kódrendszerek is létrejöttek (ilyenek pl. füstjelek, írás). Azonos kódokhoz más és más kódrendszerek kapcsolódhatnak pl. beszélni többféle nyelven is lehet, ezek más-más kódrendszerek. A kód- és kódrendszereknek a szükségessége azért volt jelentős és azért jelentős ma is, mert az embereknek kapcsolatot kellett teremteniük egymással, s ma még nincs agy-agy kapcsolat és ezért valamilyen közvetítésre van szükség (hang-levegő, fény, elektronika). Ezen közvetítők közül az elektronika ma már elég nagy szerephez jutott. Mivel hagyományosan nem volt megoldható nagy távolságra az információközlés és ma a nagy tömegű, gyors adattárolás is az elektronika vívmánya.

Az elektronikában használt kód, lehet analóg, illetve digitális. Az analóg jel nem megszámlálható feszültségi szinttel küldi az információt, míg a digitálisnál ezek a szintek megszámlálhatók (véges sokan vannak). A számítástechnikai elektronikában két feszültségi szintet különböztetünk meg. E szerint 0-nak illetve 1-nek hívjuk a jeleket. így mi kódként, 0-ák és 1-esek sorozatáról beszélhetünk. Erre sok kódrendszer jött létre a számítástechnikában. Általában ezek funkció szerint különböznek egymástól pl.

- utasítási kódrendszer
- szövegmegjelenítési kódrendszer (ASCII)
- számábrázoló kódrendszer
  - helyiértékes (lebegőpontos,egész)
  - BCD

Ezek mellett, illetve ezeken belül megkülönböztetjük még a kódokat változó- és fixhosszúságú kódokként.

Ebben a fejezetben három kódrendszerről lesz szó bővebben.

Az első az ASCII-kódrendszer, amely a PC-ken elterjedt. Jellemzője, hogy 256 jelet különböztet meg. Ezekből épül fel minden információ a gépben. Ez a kódrendszer tartalmazza az ABC betűit, számokat és különböző jeleket.

Egy másik, a BCD (Binárisan Kódolt Decimális Számok) kódrendszer szintén igen elterjedt a számítógépeken. Mint a neve is mutatja, a decimális (tizes számrendszerbeli) számokat binárisan tárolja (kódolja). Tehát nem konvertálja át az adott számot kettes számrendszerbe, a műveleteket is tizes számrendszerben végzi, s az eredmény is végül ebben "érkezik" meg hozzánk. Itt egy szám egy byte-on tárolódik.

Tudod a választ ?

1., Sorold fel a számítástechnika úttörőit!

2., Milyen részekből állt Babbage analitikus gépe?

3., Mi a lyukkártya?

4., Mi a Neumann-elv lényege?

5., Mi jellemezte az ENIAC-ot? (felépítése, előnyei, hibái)

6., Mit tudsz a számítógép-generációkról?

7., Mi az információ?

8., Mi a BIT és a BYTE?

9., Mit értesz a kód és a kódrendszer fogalmán?

10., Sorolj fel kódrendszereket, és jellemezd őket!

Hardver

Bevezetés

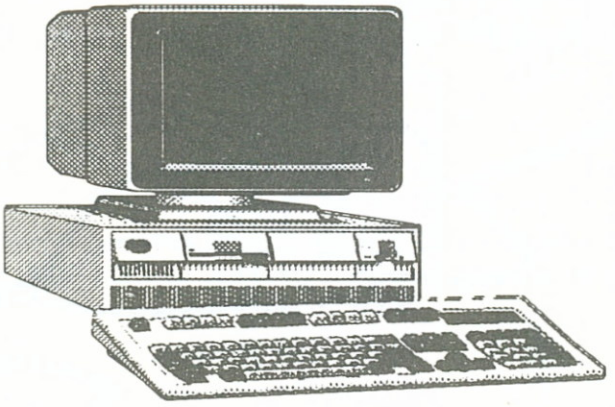
Ha számítógépről beszélünk, két alapfogalmat kell megkülönböztetnünk: a hardvert (keményárut) és a szoftvert (lágyárut).

Talán hallottál arról, hogy a szülő és az anya megkülönböztethető. A táncvizsgán a szülő az, akit büszkeség tölt el, ha a fiacskája jól táncol és mindenki őt nézi, és az anya az, aki a legszívesebben felpofozná a szőke fruskát, aki merengeti csodaszép kék szemét az ő egyetlen fiára. Ha kapsz egy pofont, akkor azt a szülőtől kapod. Aki belül veled sír, az az anya. Ugyanígy a hardver és a szoftver is megkülönböztethető, hiszen ha fogsz egy számítógépet, azt darabjaira szeded és így vizsgálod, az a hardver. Ha bekapcsolod és kapcsolatot teremt veled, az már a szoftver. Tehát ami megfogható, az a hardver. Ebben a fejezetben az általad használt hardverről lesz szó.

A számítógép hardvert fontos megismerni, hiszen elképzelhető, hogy tanácsot kérnek tőled, ha venni akarnak egy gépet, vagy használni akarsz egy szoftvert, és el kell döntened, melyik gépen fut, vagy mert benne van a tananyagban.

A mikroszámítógép két alapvető egységre bomlik szét; az alaplapra és a beviteli/kiviteli egységekre, idegen szóval input/output egységekre, rövidítve I/O-kra. Mi a továbbiakban csak egy átlagos IBM hardverről fogunk részletesen szólni, más gépek felépítéséről nem beszélünk most még. Úgy közelítjük meg a témát, ahogy te látod a számítógép teremben a gépet kívülről.

Mi van az asztalon? (IBM konfiguráció)



Persze ezek lehetnek a földön is, de akkor elbotlasz bennük!

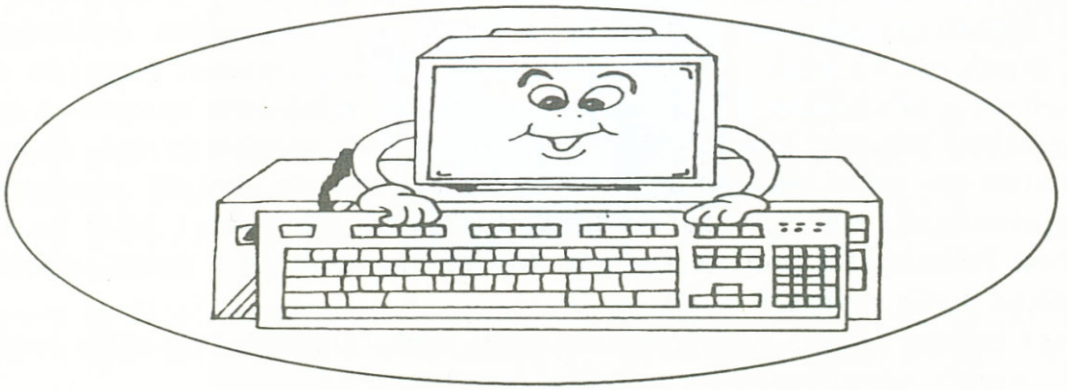
Alap konfigurációnak hívják azt a felépítést, amikor egy doboz van, egy képernyő (display, monitor), amit nézel (output), és egy billentyűzet (klaviatúra), amivel beküldöd a "gondolataidat" (input).

Amit nézel (monitorok)

A monitorokat osztályozhatjuk színük szerint és felbontásuk szerint. Színük szerint lehetnek két színűek (monochrome) vagy több színűek, ezeket színes (color) monitoroknak nevezzük. Felbontásuk szerint lehetnek kis felbontásúak szöveges használatra, vagy nagy felbontásúak szövegek és grafikus ábrák megjelenítésére is. Érdekes lehet még a monitor mérete, ami mostanában egyre változatosabb alkalmazást tesz lehetővé. (Pénztárgépeken lévő arasznyi monitortól a műszaki rajzlap nagyságú monitorig bezárólag.) Ma már vannak olyan monitorok, melyek fénykép minőséggel, vagy afölött képesek megjeleníteni.

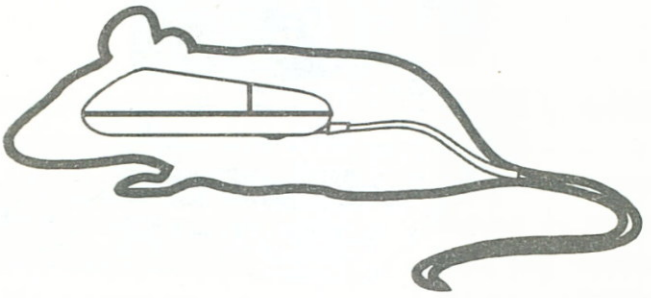
Hogyan viheted be az információt? (INPUT)

Az információ alapvető beviteli eszköze a billentyűzet (klaviatúra, konzol). Vannak rajta írógép-billentyűk, funkció, vezérlésmódosító és speciális billentyűk. Írógép-billentyűn az ABC betűit és a számokat vagy a különféle jeleket (karakterek) értjük. A funkcióbillentyűket (F1-F12) egy-egy program használja. Vezérlésmódosító billentyűk például az ENTER, a SHIFT, a CAPS LOCK, a SCROLL LOCK, az ALT, a CTRL. Speciális billentyűk a nyilak, a BACKSPACE, a HOME, az END, az INSERT, a DELETE, a PAGE UP, a PAGE DOWN, a TAB, az ESC.

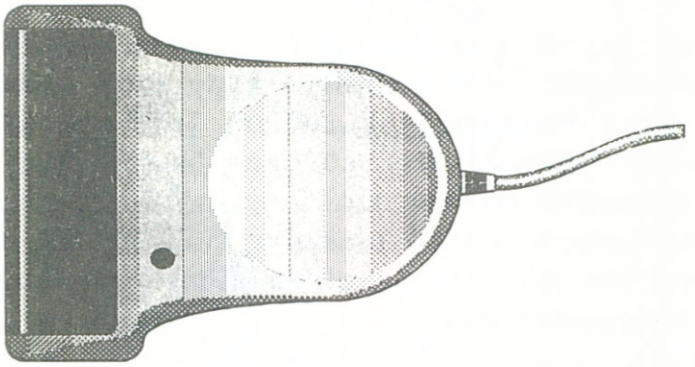


Sok billentyű funkciója programfüggő, azaz attól függ, melyik programot használod. A billentyűk funkcióit a gyakorlatban, folyamatosan érdemes elsajátítanod.

Egyre elterjedtebb beviteli eszköz az egér (mouse), amely relatív (viszonyított) mozgást tud visszaadni. Különböző felépítésű egerek vannak. Az egér használata könnyen elsajátítható.



Vannak olyan eszközök, amelyekkel a számítógépekbe lehet képi, illetve hangi információt bevinni. Ezeket a gépben feldolgozhatjuk. Ilyen a **scanner**nél, amivel képi információt tudunk bevinni. Ez az információ akár lehet egy írott szöveg is. A scannereket is csoportosíthatjuk kézi- és lapscannerekre. A kézi scannernél (ábra), amelyet kezünkkel mozgatunk, sokkal drágább, de pontosabb a lapscanner, melynek mozgatását motoros szerkezet végzi. Videokamerát is használhatunk scanner-ként.



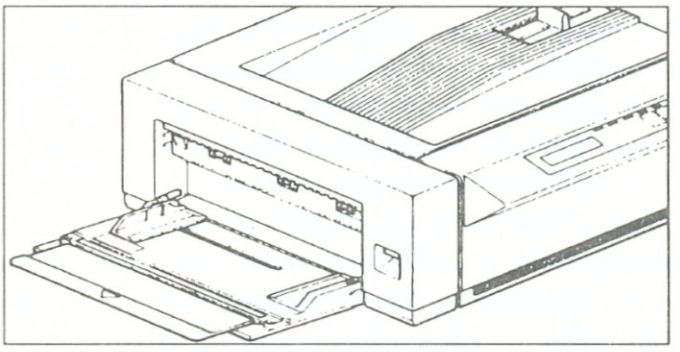
Áruházakban biztosan láttál vonalkód olvasó berendezést, amelyiknek működési elve nagyon hasonlít a scanner-hez.

A gépből jövő információ megjelenítése (printerek)

A legelterjedtebb kimeneti eszköz a nyomtató (printer). Leggyakoribb fajtái a **mátrix**, a **lézer** és a **tintasugaras** printer.

A mátrix printer írófeje tüskékből áll. Működés közben a fej adott sebességgel halad el a festékszalag előtt. Amikor a karakter kirajzolása szükséges, a tűkre kalapácsok ütnek rá, és a tűk e festékszalagon keresztül nyomot hagynak a papíron. A szalag a papír és a tűk között fut. A mátrix nyomtató működése hasonlít a képpontokból felépített grafikus képernyő működésére. Itt is azt kell meghatározni, hogy a papír bal szélétől kezdve az adott sorban milyen magasságban és melyik oszlopban kell egy pontot megjeleníteni. Az egymás alatt elhelyezkedő tűk száma jelzi az egy karakter magasságához felhasznált pontok számát. Minél nagyobb ez a szám, a felbontás annál jobb. Általában a tűk száma 9 v. 24.

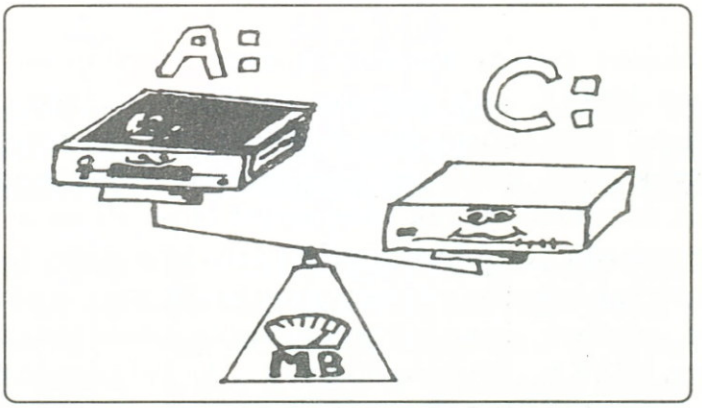
A lézer printer (ábra) a mátrix printernél jobb felbontást ad (erre szolgál a lézer), és gyorsabb, szebb, csendesebb nyomtatást tesz lehetővé.



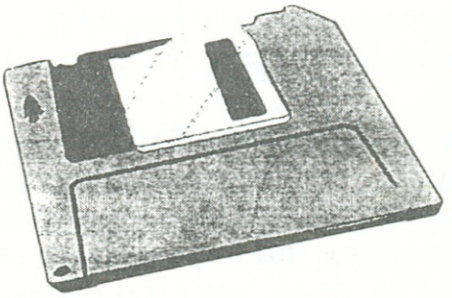
A tintasugaras printer olcsóbb a lézernél, közel olyan minőségű, de lassúbb és rövidebb élettartamú.

Mi van a dobozban?

Ha felnyitod a doboz tetejét, sok úgynevezett **chip**-et (integrált áramkört, a továbbiakban **IC**-t) találsz. Ezek vízszintes és függőleges nyomtatott áramköri (**NYÁK**) lapokon helyezkednek el. Találsz benne egy tápegységet, ami a szükséges szuflát adja a gépnek. Találsz még különböző zárt dobozokat itt-ott lyukakkal. Ezek a háttértárolók, amelyeknek négy fajtáját különböztetjük meg. A hajlékonylemezes meghajtó (**floppy drive**), a merevlemezes meghajtó (**winchester**), a szalagos tár (**streamer**) és a **CD ROM** (optikai lemez).



Többnyire azonban csak az első két háttértároló található meg a dobozban. A hajlékonylemezes meghajtó adathordozója a hajlékony mágneslemez (**floppy disc**). A meghajtó működése hasonló a hagyományos lemezjátszóéhoz, de a tű helyett mágneses olvasófej van, amely ellentétben a lemezjátszó tűjével, nem érintkezik a mágneslemez felületével, és mozgása sem kötött, egy program szabályozza.



A lemezek tárolókapacitása különböző, általában 360 kbyte-ra (kilobyte) vagy 1,2 Mbyte-ra (megabyte) készített (formázott) lemezeket használnak. Az IBM első

típusai (pl. az XT) 360 kbyte-os lemezeket használtak, és ehhez való meghajtót. A későbbi típusok már 1,2 Mbyte-os lemezeket is képesek olvasni. A fenti lemezek mérete $5^1/_4$ inch (kb.13,2 cm), újabban nagyon gyakori a $3^1/_2$ inch (kb.9 cm), amelynek megbízhatósága és védettsége is jobb; ezeket 720 kbyte-ra, illetve 1,44 Mbyte-ra, esetleg 2,8 Mbyte-ra szokták formázni (ábra).

A merevlemezes meghajtó a hajlékonylemezes meghajtó gyors testvére. A lemezek itt nem cserélhetők, sőt légmentesen vannak elzárva a külvilágtól. Ezekben egyszerre több lemez is van, és az író-olvasófej is közelebb van a lemezekhez. A lemezek tárolási kapacitása is lényegesen nagyobb, hiszen 10-20 Mbyte-tól akár több Gbyte-ig (gigabyte) is terjedhet. A merevlemezes tárolón lévő adatok elérési sebessége 4-20-szorosa a floppy sebességének.

A streamer a közönséges audio kazettához hasonlóan működik. Azzal a céllal hozták létre, hogy a merevlemezes tárolók már nem állandóan használt, de időnként még szükséges tartalmát elmentsék (archiválás).

A CD ROM ugyancsak olvasható, de alacsony ára és nagy tárolókapacitása miatt egyre elterjedtebb tárolóeszköznek bizonyul. Oktatási anyagok vagy program-gyűjtemények, enciklopédiák terjesztésében nagyon jól felhasználható.

A gép alján lévő nyomtatott áramköri lapot **alaplap**nak hívjuk, itt található a legfontosabb alkatrész, a **mikroprocesszor**, ami külsőre alig különböztethető meg a többi IC-től. Itt helyezkedik el a működéshez elengedhetetlenül szükséges kétféle tárolótípus, a **ROM** és a **RAM**. Itt találhatók a bővítőkártyák számára párhuzamosan elhelyezett foglalatok, amelyekbe a bővítőkártyák tetszőleges sorrendben helyezhetők el.

A mikroprocesszor típusa határozza meg az alaplap jellemzőit. Az IBM gép mikroprocesszorait az INTEL cég gyártja. Az IBM XT-kben a 8088-as sorozatszámú, az AT-kben a 80286-os, a 80386-os és a 80486-os típusú processzorok találhatók. Ezek a típust jelölő számok nem csak számegyenesen helyezkednek el sorban, hanem a tudásuk, gyorsaságuk és elkészülésük időpontját is jelöli ez a sorrend.

A mikroprocesszorok fejlődése mellett folyamatosan fejlődtek a co-processzorok is, amelyek a matematikai műveletvégzést gyorsítják meg a számítógépeken. Így minden mikroprocesszorhoz lehetett egy co-processzort illeszteni annak fejlettsége szerint (8086-8087, 80286-80287, 80386-80387). A felsorolásból kimaradt 80486-os mikroprocesszorba már gyárilag be van építve a co-processzor. Főbb alkalmazási területük a bonyolult matematikai számításokat igénylő programok, ahol a mikroprocesszortól átveszik a számítások jelentős részét. Ezáltal a futási idő sokkal rövidebb lesz. Ilyen felhasználói programok a CAD, az animációs és a térábrázoló programok.

A ROM - angol nyelvű rövidítés (**R**ead **O**nly **M**emory) - csak olvasható memóriát jelent. Ide kerülhetnek azok a programok, amelyek a gép ki- és bekapcsolása után is változatlanul megmaradnak. Ez a memória adja a gép bekapcsolás utáni induló tudását.

A RAM - angol nyelvű rövidítés (**R**andom **A**ccess **M**emory) - olvasható-írható memóriát jelent. Többek között ide írjuk vagy lemezről olvassuk be programjainkat és adatainkat. A ROM kapacitását a gép gyártói döntik el, a RAM kapacitásától függ, hogy mekkora programot vagy mennyi adatot olvashatsz be egyszerre. Tehát gépvásárláskor te határozhatod meg, hogy feladataid és pénztárcád alapján mekkorát vegyél belőle.

A függőleges elhelyezésű kártyáknak nagyon sokféle szerepük lehet. Minden gépben van olyan kártya, adapter, amely a monitorhoz szükséges. Ennek típusától is

függ, hogy a monitor képes-e színes, grafikus megjelenítésre. Ezen kártyák legelterjedtebb típusai a következők: MGA (Mono Graphics Adapter), amelyet a kétszínű monitorhoz alkalmazunk; CGA kártya (Color Graphics Adapter), ez már színes, de elavult kártyatípus, rossz a felbontása és csak néhány színt tud; EGA kártya (Enhanced Graphics Adapter), több színt tud és nagyobb felbontást tesz lehetővé, de már ez is eléggé elavult; VGA kártya (Video Graphich Array), amely jó felbontása és sok színe miatt széles körben elterjedt. Most kezd elterjedni a szabványos TIGA (Texas Instruments Graphics Adapter). A különféle újabb kártyákról és a VGA típusú kártyák fajtáiról itt most nem ejtünk szót. Az eddig felsorolt kártyák többnyire IBM szabványúak voltak, ellentétben a mellettük elterjedt Hercules grafikus kártyával. Ez a kártya nagyfelbontású, ám csak kétszínű monitorhoz alkalmazható. Érdemes megjegyezni, hogy a kártyákon elég nagy memória is szükséges, mivel a képet bizonyos jellemzőkkel tárolni kell. Minél nagyobb felbontású a kép, annál több a memória a videokártyán.

Hogy használni tudjuk a háttértárolókat, szintén szükség van egy vagy több kártyára. Kártya kell külön az egérhez, a scanner-hez, ha hálózatba akarod kötni a gépet.

A kártyák egyetlen csavar kicsavarása után kivehetők és betehetők. Ha új kiegészítést veszel magadnak, nem biztos, hogy betétel után működni fog, hiszen elképzelhető, hogy a kártyán lévő kis kapcsolók nem a megfelelő helyzetben állnak. Érdemes ezek elhelyezéséhez a mellékelt utasítást elolvasni vagy szakember segítségét kérni.

Hasznos tanácsok

Most lássuk azt a szituációt, amikor már mindent tudunk a gépről, hogy mi van benne, illetve minek kellene benne lennie, és szeretnénk leülni a géphez dolgozni.

Szerencsések vagyunk akkor, ha előttünk áll fel valaki a bekapcsolt géptől, s így egyből dolgozhatunk.

De mi van akkor, ha a gép nincs üzemképes állapotban? Ebben az esetben *be kell kapcsolni*. Ha jól előkészített gépünk van, akkor rövid időn belül sípol, villog és máris munkára kész. Ha ez nem így történik, annak több oka is lehet: pl. nem találjuk a kapcsolót, nincs 220 V, illetve nincs a gép a hálózatba csatlakoztatva. Ha pedig fizikailag rossz a gép, akkor nem sokat tehetünk a munka eredményessége érdekében. A *kapcsoló* általában a *gépdobozok jobb oldalán* található, de lehet hátul (pl. Macintosh) vagy elöl, esetleg rejtetten egy kis ajtó mögött. A másik két problémát értelemszerűen kell korrigálni.

Ezen hibák megszűnte után a gépnek működnie kell, de az is lehet, hogy nem minden tartozék üzemel még tökéletesen.

Ha például egy perces várakozás után a monitoron nem látszik semmi változás, akkor ez ismét több okra vezethető vissza. Az egyik: nincs rendesen a számítógépre és a *monitorra* csatlakoztatva a *jelkábel*, s *hálózati csatlakozó* (ez utóbbi lehet a gépdobozba, illetve hálózatba illesztve, a kábel kialakításától függően).

Ha ez rendben is van, még mindig fennállhat az a veszély, hogy nincs kép. A másik lehetőség: a *monitorokat* általában *külön kell bekapcsolni*, s lehet, hogy ez esetben itt a hiba. A bekapcsolt állapotot egy világító LED (light-emitting diode, kis szines lámpa) is jelzi.

Ha a kép még mindig nem jelentkezett, akkor előfordulhat, hogy a *fényerő* és a *kontraszt* van elállítva. Ezek szabályzói általában a monitor elején találhatók.

Az eddigi ellenőrzések és javítások után a monitorod működik. Ha mégsem történne meg, ez annak lehet a következménye, hogy a dobozban lévő grafikus kártyának van *illesztési problémája* (ez a többi kártyával is előfordulhat). Ekkor kérd egy szakavatottabb segítségét.

Ám ha a monitor már működik, de mégsem tűnik tökéletesnek a kép - gondolunk itt arra, hogy VGA monitornál például nincs piros szín. Ekkor előfordulhat, hogy kábelszakadás van (ennek javítása szinte lehetetlen), vagy a csatlakozó végén a *"tüskék" el* vannak *görbülve*, ami egy brutális, illetve még brutálisabb ember munkájának a következménye.

Ha a monitor még mindig nem működik, akkor passz.

Tegyük fel, hogy a gép működik, és ezt látjuk a monitoron, de sípol egyet, és hibaüzeneteket ír ki. A továbbiakban megnézzük a leggyakrabban előfordulókat.

A billentyűzethibának négy-öt oka lehet, ötödik ha rossz a *billentyűzet*. Egyik lehetőség, ha *nincs az alaplapra csatlakoztatva* (ha pótoltuk a hiányt, újra kell indítani a gépet). Másik, ha a billentyűzet alján lévő kapcsoló *nem az adott géphez van állítva* (XT - AT állítás lehetséges, utána újra kell indítani a gépet). A harmadik az, ha még *gépindulás közben maceráltad* a billentyűzetet. Ilyenkor a gép által felkínált billentyű "billentésével" folytatható az élesztés). Előfordulhat még az is, hogy egy menüs képernyő jelenik meg, ami hardver setup beállítására alkalmas. Ha nem direkt léptél be ebbe a menübe, az ESC billentyűvel lépj ki belőle, mert használhatatlanná teheted vele a gépet (esetleg Y-t és egy ENTER-t is kell nyomni az ESC után). Billentyűzet hiba előfordulhat akkor is, ha a billentyűzet zár *le van zárva*, ezt ki kell nyitni. (Itt a doboz elején található KEY feliratú fém ketyerére céloztunk.)

Ezenkívül *hibák* adódhatnak még a *háttértárak nem megfelelő bejelentkezéséből*. Javításuk *újraindítás*sal talán lehetséges, de egyéb lehetőségre a hibák valószínűtlensége miatt nem térünk ki.

Ha pedig arra utasít a gép, hogy helyezz *operációs rendszert az (A:) meghajtóba*, akkor vagy rendszert nem tartalmazó lemezt használsz, vagy egyáltalán nincs lemez a meghajtóban (ez winchester nélküli gépen értendő). Javítás: megfelelő lemez elhelyezése. Ha van winchester és ugyanezt az üzenetet kapjuk és nincs lemez a meghajtóban, akkor a winchesteren romlott el a rendszer. Javítás: új rendszer elhelyezésével. Ha viszont van lemez a meghajtóban, vagy csak a kar el van fordítva, akkor a kar felengedésével és az Enter leütésével lehet javítani. Ezen tünetek okát talán jobban megérted, ha tisztában vagy a gép logikájával.

Az IBM PC típusú gépek valamilyen háttértárolóról várják az operációs rendszer betöltődését, ezért az élesztgetés végeztével a háttértárakhoz fordul a rendszerért. Ha egy tárolónk van, akkor azon keres, ha több, akkor bizonyos sorrendben nézi végig azokat. Általában egy hajlékonylemezes meghajtó (floppy) és egy merevlemezes meghajtó (winchester) található egy gépben. Ebben az esetben először a floppy-n, s utána a winchesteren keresi az operációs rendszert. Ettől eltér a sorrend, ha valamelyik egység nincs beinstallálva a setup programba (itt vannak felsorolva a géphez csatlakozó eszközök, s ezen listába való felvételt értünk beinstalláláson), illetve ha van olyan intelligens a setup program, hogy a sorrendcserét végrehajtja. Gyakori azonban az is, hogy egy gépben több floppy vagy winchester található. Ilyenkor hardveresen (a kábelek átcsatlakoztatásával és a kapcsolók átkapcsolásával) van kijelölve, hogy az azonos egységek közül melyik az az egy, amely képes betölteni az operációs rendszert.

A munka végeztével a gépet *ki kell kapcsolni*. Ez a már megtalált kapcsológomb segítségével tehető meg. Mielőtt kikapcsolnánk a gépet és tudjuk, hogy *a winchester nem automata parkos* (olvasófej biztonságba helyezés), akkor le kell futtatni egy parkoló programot, ami ilyen gépeken általában megtalálható (pl. OFF, PARK programok). Ez azért fontos, mert a winchesterben szállítás közben a fejet tartó kar a lemezekre merőlegesen is elmozdulhat, s ettől tönkre mehet a fej és a lemez felülete is. Ismételt bekapcsolással a gép újra indul. Ha munka közben szeretnénk a gépet újra indítani (pl. lefagy a gép v. a memóriát akarjuk kiüríteni), akkor ez vagy a doboz elején lévő RESET feliratú gombbal, vagy pedig a billentyűzeten a CTRL-ALT-DEL billentyűkombinációval lehetséges. (Itt említenénk meg, hogy az újraindítás azonos fogalom a bootolással.)

Operációs rendszerek

Bevezetés

A mai számítógép alkalmazók közül még csak elenyésző kisebbség tudja, valójában miből is áll a gép, a nagy többség még csak nem is látta a csupasz hardvert.

Ezeknek a felhasználóknak "nincs is szükségük" a hardverre, számukra a tulajdonképpeni gépet azok a szoftverek jelképezik, melyek szinte teljesen elfedik azt. Ezek közül a legjelentősebb a rendszerszoftver (operációs rendszer), mely az ISO nemzetközi szabványosítási szervezet szerint a következő: "Az operációs rendszer olyan programrendszer, mely a számítógépes rendszerben a programok végrehajtását vezérli: így például ütemezi a programok végrehajtását, elosztja az erőforrásokat, biztosítja a felhasználó és a számítógépes rendszer közötti kommunikációt".

Az operációs rendszerek először a nagygépek jobb kihasználhatósága érdekében jelentek meg. Azonban az árak erőteljes csökkenésével a körülmények megváltoztak, és megjelentek a mikrók, melyeknél a hatékonysággal szemben a felhasználó minél barátságosabb és jobb "kiszolgálása" erősebb szempontnak bizonyult. Ezután már egyre-másra jöttek létre az operációs rendszerek, melyek szinte minden esetben felülmúlták elődeiket.

Az operációs rendszerek több csoportra oszthatók. A rendszerprogramozók igyekeztek úgy megírni a rendszerszoftvert, hogy az minél jobban megfeleljen a felhasználó igényeinek.

Ezek közül most három csoportot különböztetünk meg:

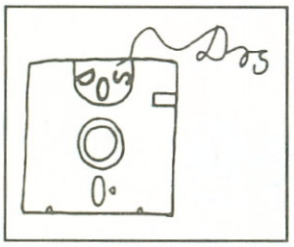
1. Egyfelhasználós operációs rendszerek
2. Grafikus Multitask operációs rendszerek
3. Multitask és Multiuser operációs rendszerek

Egyfelhasználós operációs rendszerek

Az egyfelhasználós rendszerek eredetileg a nagygépek operációs rendszerei voltak. Mivel ezek mind kezdeti stádiumban lévők, költségesek és lassúak voltak, ezáltal egyszerre csak egy felhasználót tudtak kiszolgálni. A hardver egyre olcsóbbá, gyorsabbá és egyre nagyobb tudásúvá vált, ezáltal a figyelem középpontjába nem a munkával való ellátottság, hanem a felhasználónak biztosítandó kényelmes környezet került. Ezt a párbeszédes, barátságosan működő és segítőkész operációs rendszerrel lehet elérni. Bár nem a legjobbak, napjainkban mégis az egyfelhasználós rendszerek a legelterjedtebb operációs rendszerek. (Pl.:DOS)

A DOS

Az első DOS (Disk Operating System) verziót 1980-ban fejlesztette ki a Microsoft, célja egy könnyen kezelhető és gyorsan elsajátítható rendszerszoftver megalkotása volt. A DOS mint neve is mutatja (Lemezes Operációs Rendszer) gyökeresen eltért elődeitől. A DOS fejlődésében bekövetkezett jelentősebb változásokat a '.' előtti egész szám, így például a 3.20 verziószámban a hármas szám jelöli. A pont utáni számok a



fejlődés kisebb állomásainak felelnek meg. A régebbi operációs rendszerek általában az adott gép ROM-jába voltak beégetve és a bekapcsolás után azonnal aktivizálódtak, addig a DOS lemezről töltődik be a gép memóriájába.

Ezután a 80-as évek folyamán a Microsoft többször is továbbfejlesztette, és egyre újabb verziókkal jelent meg a szoftverpiacon.

DOS alapok

A DOS-ban a hardver és a felhasználók illetve a felhasználói programok közötti kapcsolatot három, egymástól különálló szintre oszthatjuk.

A hardverhez legközelebb álló szinten a BIOS (Basic Input/Output System) található. A BIOS az IO.SYS nevű állományban található amely ún. rejtett és rendszer típusú (attributum) file.Ez az állomány a rendszer inicializálásakor kerül a memóriába. Azokat a programokat tartalmazza, amelyek a:

-billentyűzet, képernyő (con);
-nyomtató (prn);
-soros interface (aux);
-dátum, idő (clock);
-lemezmeghajtók (A,B,C);

hardverelemeket vezérli. Abban az esetben, ha a fenti egységek közül valamelyik használatba kerül, akkor a szóban forgó egységben lévő vezérlő program meghívja a ROM-ban található BIOS rutinokat. Ez tehát a DOS leginkább hardverfüggő, ez kapcsolódik a gépbe beégetett ROM-BIOS-hoz

A középső szinten elhelyezkedő rutinokat az MSDOS.SYS tartalmazza, amely szintén rejtett és renszer típusú állomány.Az MSDOS.SYS többek között állományokon történő műveletek elvégzésére, a RAM kezelésére és a karakterek be és kivitelére alkalmas rutinokat tartalmaz.

Az előző két szintet magába foglaló modultól eltérően a parancsértelmező processzort egy látható állomány tartalmazza, melynek neve COMMAND.COM. Ez a program egy speciális értelmező program, mely a DOS alatt fut (ellentétben azzal a téves hiedelemmel, hogy ez maga a DOS). A belső, leggyakrabban használt parancsokat maga a COMMAND.COM tartalmazza.

A DOS fenntart egy speciális fájlt, az AUTOEXEC.BAT-ot, ami azért különleges, mert a rendszer indításakor automatikusan betöltődik. Ebbe a fájlba saját parancsainkat rakhatjuk bele, ami DOS parancsokból és batch parancsokból áll. A BATCH parancsok egyszerű dolgokat látnak el, főleg fájlműveletre alkalmasak, ezeket bármilyen BATCH programban felhasználhatjuk, nem csak az AUTOEXEC.BAT-ban.

A meghajtók

A DOS több eszközzel is rendelkezik, amely a felhasználóval teremt kapcsolatot. Az eszközöket a továbbiakban meghajtóknak nevezzük. A különböző meghajtókat a DOS címkékkel látja el. A lemezegységek esetében ez a címke egy betű (Pl.: a hajlékony lemezre az A: v. B: betűvel, a merevlemezre a C: v. D: betűvel hivatkozhatunk). A billentyűzetre és a nyomtatóra három betűvel hivatkozhatunk (con, prn).

Az adatállományok

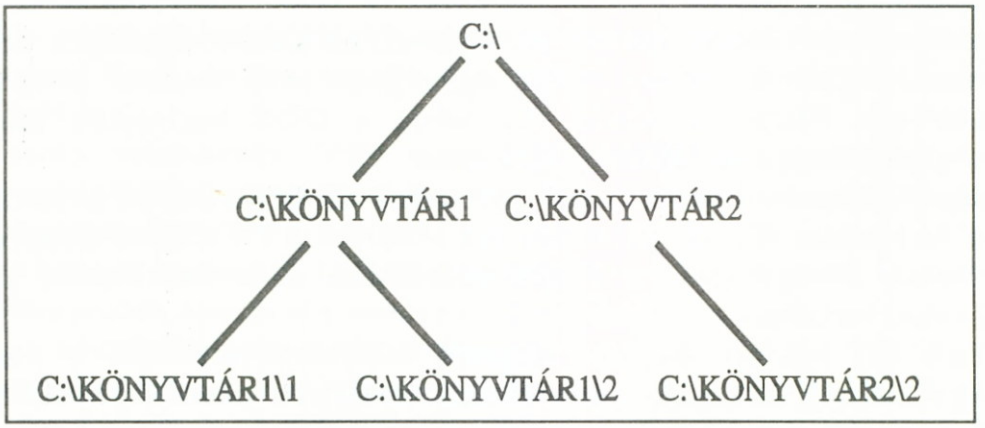
Az adatállományokban (fájlokban) adatainkat, szövegeinket, programjainkat tárolhatjuk a lemezen. A fájlokra névvel és három betűs kiterjesztéssel hivatkozhatunk. A fájlokat több csoportba oszthatjuk. A futtatható adatállományokban (COM, EXE, BAT kiterjesztésűek) programjainkat tárolhatjuk.

A szekvenciális adatállományokban szövegeket, adatokat, táblázatokat stb. tárolhatunk.

A fájlstruktúra

A DOS alatt lehetőség van arra, hogy a lemezegységeken (merev, hajlékony) tárolt file-okat alkönyvtárakba tegyük. Ez azért előnyös, mert a lemezekre nem ömlesztve kerül az adatok tömege, hanem rendezetten, logikai szempontból elkülönítve alkönyvtárakba rendezve. Ezzel a rendszerrel a DOS megkönnyíti a felhasználó eligazodását az adott lemezen. Egy alkönyvtár elérési útvonalát a backslash (\) jel segítségével adhatjuk meg,

Pl.: **C:**könyvtár1\1



ahol a 'C' betű a meghajtót jelöli, a könyvtár1 egy alkönyvtár neve, és a könyvtár1\1 egy könyvtár1-en belüli alkönyvtár neve. (Egy-egy alkönyvtáron belül elméletileg szinte végtelen új alkönyvtár lehet.)

Fájlkezelő utasítások

Ahhoz, hogy az adatállományokat, programokat hatékonyan tudjuk használni, szükségünk van bizonyos adatállomány kezelő utasításokra. A DOS-ban kétféle utasítás típus van: a belső és a külső. A belső utasítások a command.com-ból töltődnek a memóriába. A külső utasítás ellenben külön programként mindig akkor töltődik a memóriába, amikor kiadjuk. Az utasítások közül a legalapvetőbb a **DIR** parancs, mely segítségével a képernyőre listázhatjuk az aktuális könyvtár tartalmát. A munkánk során szükség lehet a fájlok másolására, ezt a **COPY** paranccsal tehetjük meg. Ennek a formája **COPY** innen-ide (pl.: **COPY** ezt c:\ebbe\a\konyvtar\ba) Ha egy file-t törölni akarunk, akkor erre a **DEL** parancs szolgál pl.: **DEL** törlendő.com (ennek hatására a törlendő.com file eltűnik a lemezről). Alkönyvtár létrehozása az **MD** paranccsal lehetséges (pl.: **MD** ujkönyv ennek hatására egy új könyvtár keletkezik), alkönyvtár letörlését az **RD**

paranccsal végezhetjük (ha az adott könyvtár üres). Egy alkönyvtárba a **CD** paranccsal léphetünk (pl.: **CD** ujkonyv), az alkönyvtárból kilépni pedig a **CD**.. parancs kiadásával lehet.

Helyettesítő karakterek

Gyakran szükség lehet arra, például a file-ok törlésénél, hogy az azonos típusú állományokat egységesen kezeljük. Erre a DOS két lehetőséget ad. Ha a file teljes nevét vagy a kiterjesztését akarjuk helyettesíteni, akkor a * jelet kell használnunk. (DIR *.exe: ennek a parancsnak a hatására a könyvtárban lévő összes állomány közül csak az EXE kiterjesztésűek jelennek meg, míg a DIR adat.* parancsnál azok az állományok, amelyek 'adat' névre hallgatnak. Ezek kiterjesztése bármi lehet) Ha csak azt szeretnénk, hogy egy betűt helyettesítsünk akkor a ? jel lehet a hasznunkra (pl DIR ezeket.pc? ekkor az összes 'ezeket' nevű fájlt kiírja, aminek a kiterjesztésének az első két betűje pc, pl.: PCC , PCX).

Lemezkezelő utasítások

Ahhoz hogy a lemezeket, merevlemezeket használni tudjuk, inicializálni kell azokat. Ez azt jelenti, hogy felkészítjük arra, hogy az IBM fogadni, használni tudja őket. Ezt a folyamatot szaknyelven formázásnak nevezzük. A DOS-ban a lemezeket a FORMAT utasítással formázhatjuk (FORMAT A:). Ennek az utasításnak sok paramétere van. (Paraméternek nevezzük azokat a szavakat vagy betűket, amikkel a parancs neve után gépelve a parancsot utasíthatjuk, hogy valamilyen speciális funkcióját nyújtsa). Mivel egy lemezt sokféleképpen formázhatunk, ezek közül csak néhányat említünk meg.

Paraméterek:

/V = Megadhatjuk a formázandó lemez nevét;
/S = felmásolja a rendszer állományokat a lemezre;
/4 = 1.2Mb-os lemezmeghajtóban 360 kb-osra formáz.

A lemez nevét a **VOL** paranccsal is megadhatjuk.

A lemezeket a **CHKDSK** programmal tesztelhetjük, ez a program minden számszerű adatot kiír a képernyőre.

Néhány BATCH utasítás

ECHO OFF v. **ON** kikapcsolja vagy bekapcsolja a visszhangot (ha be van kapcsolva, akkor a BATCH program futtatásakor kiírja a képernyőre a program sorait).

PAUSE - egy billentyű lenyomására vár.

PATH - ez egy rendkívül hasznos utasítás, akkor használjuk, amikor egy alkönyvtár tartalmát bárhonnan el akarjuk érni (más alkönyvtárakból). Formája: PATH C:\DOS. Ha ezt a parancsot kiadjuk, akkor a DOS nevű könyvtár tartalma bárhonnan elérhető lesz.

Grafikus multitask operációs rendszerek

Az operációs rendszerek - mint ahogy imént már említettük - fejlődésük során egyre inkább a felhasználó közeliséget célozták meg. A grafikus operációs rendszerek e próbálkozások gyümölcseként keletkeztek. Ezek az operációs rendszerek egyszerű

kezelhetőségük, könnyű elsajátíthatóságuk és tetszetős környezetet nyújtó "felületük" miatt méltán váltak kedvelt és elterjedt operációs rendszerré. Ilyen rendszer Pl.: a Macintosh operációs rendszere vagy a Windows.

A MACINTOSH

A Macintosh operációs rendszert az Apple cég fejlesztette ki saját, az addigi gépektől különböző személyi számítógépe részére. Ez a rendszer volt az első memóriába "égetett" grafikus, multitask operációs rendszer, mely igen nagy előrelépésnek számított a használhatóság szempontjából. A Macintosh nagy előnye az egyéb operációs rendszerekkel szemben, hogy az alatta futtatott programok mind egy szabványnak felelnek meg, és ezáltal könnyen betanulható, elsajátítható.

A WINDOWS

A Windows-t az operációs rendszerek közé soroljuk, bár a DOS alatt indítható. Segítségével minden DOS funkció és utasítás végrehajtható. Mégsem ez az, ami annyira közkedveltté és elterjedtté tette, hanem a kellemes környezetet nyújtó grafikus felhasználói felület, mely ikonos rendszerével gyors, pontos, és a felhasználó által már megszokott állandó lehetőségeket biztosít. Bár a billentyűs beviteli lehetőségeket is képes használni, a Windows mégis alapvetően grafikus rendszer, tehát grafikus beviteli egységgel (egér) lehet a legkényelmesebben használni. A szoftverüzemeltető szempontjából mindazokkal az előnyökkel szolgál, amelyeket a különböző programok használatakor az azonos környezet nyújthat. E rendszer segítségével az azonos célú, gyakran használt programok egy logikai csoportba irányíthatók, ezáltal kezelésük egyszerűbbé és gyorsabbá válhat. A rendszer lehetőséget ad több program azonos időbeli futtatására, ezért a Multitaskos rendszerek közé soroljuk. Ennek egyik legszembetűnőbb példája a Print Manager. Mint ahogy neve is mutatja, a Windows-nak ez a része végzi a printeléssel kapcsolatos feladatokat, méghozzá úgy, hogy a memóriában tárolja a printeléshez szükséges adatokat, és ezt akkor printeli ki, amikor a gép kihasználtsága azt lehetővé teszi. Így a különböző programok közötti adatcsere jelentősen meggyorsul, amit még az is segít, hogy az adatok (szöveg, kép, stb.) tárolására egy olyan központi lehetőséget ad (Clipboard), mellyel leveszi a terhet a felhasználó válláról.

Multitask és multiuser operációs rendszerek

Már a hatvanas évek végén nyilvánvalóvá vált, hogy a felhasználók soros úton való kielégítése nem képes a hardver teljes kihasználására. Mi vajon a kiút? Az ötletet az adta, hogy az egyes munkafolyamatok között szinte mindig található olyan ki nem használt idő, mely alatt a gép nem végez érdemi munkát. A multitask alapelve tehát az, hogy a kényszerű várakozási időket kihasználjuk úgy, hogy valamilyen ütemezési stratégiával a központi egységet átkapcsoljuk a futóképes munkák közé. Később az elvet módosították és az operációs rendszer maga osztotta fel a futási időket a tárban lévő programok között, így minden programból egyszerre csak egy kis részletet futtatott le, ezután átadta a vezérlést egy másik programnak, majd ismét tovább adta egy harmadik programnak, míg a vezérlés vissza nem került oda, ahol az első programot abbahagyta. Az ilyen típusú operációs rendszereket időosztásos rendszereknek nevezzük, ezek közé tartozik a UNIX is.

A UNIX története

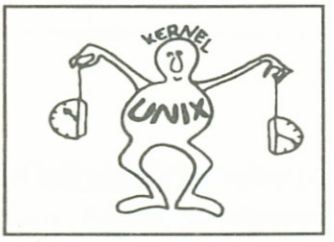
A multitask operációs rendszerek a hatvanas évek végén váltak szükségessé, így a UNIX kifejlesztése is erre az időszakra tehető. Többféle verzió született, amiket általában az egyetemek fejlesztettek ki saját használatra. A rendszer állandó változáson ment át, a legjelentősebb változás még az op.rendszerek történelmében is mérföldkőnek számít (1972): a rendszer magját, a kernelt átírták és továbbfejlesztették, de mindezt nem gépi nyelven tették, hanem a UNIX-hoz kifejlesztett C-nyelven (magasszintű nyelven ekkor írtak először op.rendszert).

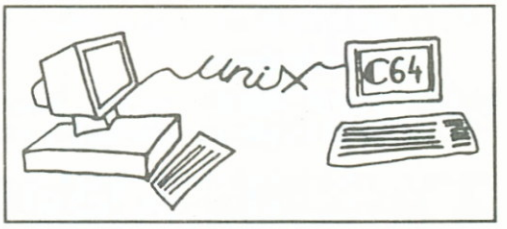
A C nyelv a UNIX-nak köszönhetően a 80-as évek legelterjedtebb nyelve lett.

1972. után a UNIX-on már több jelentős változtatást nem végeztek, bár új verziók rendszeresen megjelennek.

A UNIX lényege, előnyei

A UNIX többfelhasználós, időosztásos rendszer. Egy jó UNIX rendszer feltétele egy gyors központi gép, valamint szükséges mennyiségű terminálok (ez szükség esetén PC is lehet). A rendszert úgy kell kialakítani, hogy a központi, kiszolgáló gép teljesítménye arányban legyen a terminálok számával, ugyanis ha a kiszolgáló gép kis teljesítményű, a rendszer lelassulhat. A UNIX rendszer előnye, hogy az alá írt programok bármely típusú UNIX-os gépen futtathatóak, nincs szükség a programok újraírására. (Egy UNIX-os C64-en megírt program ugyanúgy használható UNIX-os IBM PC-n.) A UNIX lehetővé teszi, hogy az alatta futó programok egymással közvetlenül adatokat cseréljenek, ezen kívül az utasításokat egymással összefűzve is használhatjuk, így az egyik parancs kimenő adatát a másik parancs bemenő adatának használja. Ebből is látszik, hogy a UNIX-ot nem egy program vagy ötlet teszi jól működővé, hanem a programozás, az adatkezelés újszerű megközelítésének filozófiája. Ez a filozófia nem írható le egy mondatban, de a lényege, hogy a UNIX rendszer ereje, képessége nem magukból a programokból, hanem a programok közti kapcsolatokból ered.





A UNIX felépítése

A UNIX két fő részből áll: a kernelből, és a burokból. A kernel feladata a nagyobb, központi műveletek elvégzése; ez osztja el az erőforrásokat. A burok feladatköre pedig a felhasználói szinten beadott utasítások feldolgozása és továbbítása a kernel felé.

A UNIX rendszer használata alapfokon

Ahhoz, hogy kapcsolatba kerüljünk a UNIX-szal be kell jelentkeznünk a kiszolgáló gépbe, ezzel jelezzük, hogy dolgozni kívánunk a rendszer felügyelete alatt. A belépésre a **LOGIN** parancs szolgál. A login után be kell gépelnünk a felhasználói nevünket. A felhasználói név azért szükséges, mert a központi gép ezen a néven tart nyilván minket, és ezzel a névvel hivatkoznak ránk más felhasználók is. Ezután a parancs után a UNIX rendszer megkérdezi a jelszavunkat, de a jelszó beírása nem fog látszani a képernyőn. Általában egy bejelentkezés így néz ki:

LOGIN: név
PASSWORD:

Ha nem történt hiba akkor terminálunk bejelentkezik, és a rendszer kiírja, hogy van e levelünk. Ha van akkor olvassuk el a **MAIL** parancs segítségével. Ha azt szeretnénk megtudni, hogy ki van bejelentkezve a rendszerbe, akkor ezt megtehetjük a **WHO** parancssal, ennek hatására a rendszer felsorolja a bejelentkezettek nevét.

Egyszerű fájlkezelő műveletek

Az alkönyvtár-struktúra a UNIX-ban is hasonló, mint a DOS-ban, így a UNIX-ban is megtalálhatóak a DOS fájlkezelő parancsainak a megfelelői. Az **LS** paranccsal az aktuális könyvtár tartalmát írathatjuk ki, ha egy fájlt másolni akarunk, akkor ezt a **CP** paranccsal tehetjük meg, ahol meg kell adni azt a fájlt, amit másolni akarunk, és ezután azt, hogy hova másolja (**CP** innen ide). Az **RM** parancs segítségével a fájlt törölni tudjuk. Ezen kívül lehetőség van arra, hogy fájlokat összehasonlítsunk és megtekintsük a tartalmukat.

A felsorolt utasítások egyéb utasításokkal összefűzve válhatnak igazán értékessé, ha az LS-t összekapcsoljuk egy számláló utasítással, akkor megkapjuk a monitoron a könyvtárban lévő file-ok számát: **WC LS** (*a WC utasítás jelentése Word Count : számláló*)

Kérdések:

1., Sorolj fel 5 db DOS parancsot, és azt, hogy mire valók.

2., Készíts néhány alkönyvtárat, majd töröld le őket.

3., Másolj át egy fájlt egy általad készített alkönyvtárba, majd töröld le.

4., Hogy hívják azt a speciális programot, amit a DOS automatikusan elindít minden újraindításkor?

5., Mi a szerepe az operációs rendszereknek?

6., Milyen rendszerben helyezkednek el a file-ok a lemezen?

7., Milyen operációs rendszer a Windows és mi az elsődleges beviteli eszköze?

Kérdések:

8., Mit jelent az időosztásos rendszer?

9., Miért számít nagy előrelépésnek a UNIX 1972-es újraírása?

10., Milyen a UNIX felépítése?

4., Miben rejlik a UNIX újszerűsége?

Számítógépes hálózatok

A számítógépes hálózatok üzemeltetése létfontosságú a modern komputerizált szervezetek életében. A hálózatok fejlődése során egymástól eltérő rendszerek jöttek létre, amelyekben összekapcsolódnak egymással nagy- és személyi számítógépek, terminálok.

A helyi hálózatok (LAN)

A helyi (lokális) hálózatok - mint nevükből is kiderül - kis földrajzi területen helyezkednek el. Ez lehet akár egy épületen belül is - pl.: egy vállalat számítógépeit köti össze.

A lokális hálózatok összetevői

Az adapterkártya: Az adapterkártyát a hálózat állomásaként használni kívánt valamennyi személyi számítógépbe beépítik. Ez a kártya biztosítja a kapcsolatot a hálózat többi gépe és a mi személyi számítógépünk között.

A kábelrendszer: Az a kábel vagy vezetékrendszer, amely a hálózatban lévő eszközöket kapcsolja össze.

Fajtái:

a) Sodrott érpár

b) Koaxiális kábel: szigetelt, árnyékolt kábel

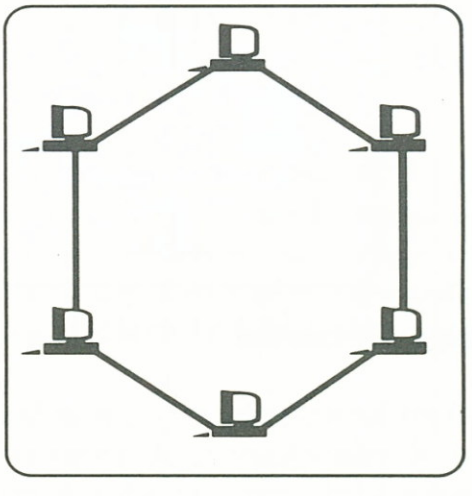
c) Optikai kábel: vékony üvegszálat tartalmazó kábel

A koncentrátor: Bizonyos lokális hálózati kialakítások koncentrátorokat használnak, hogy a hálózatban lévő eszközök egy központi helyen kapcsolódjanak (koncentrálódjanak). A koncentrátor egyszerűsíti a hálózat karbantartását, üzemben tartását.

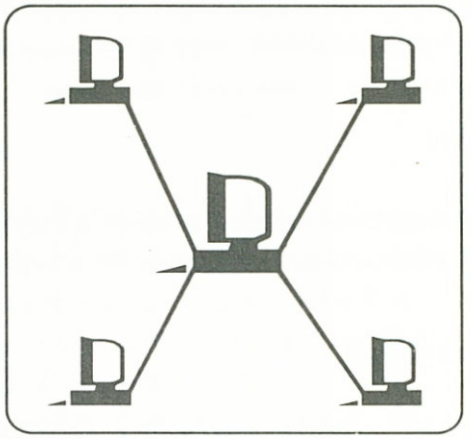
Hálózati elrendezések

A hálózatok elrendezés szerint különbözők lehetnek. Ezek közül most négyet ismertetünk.

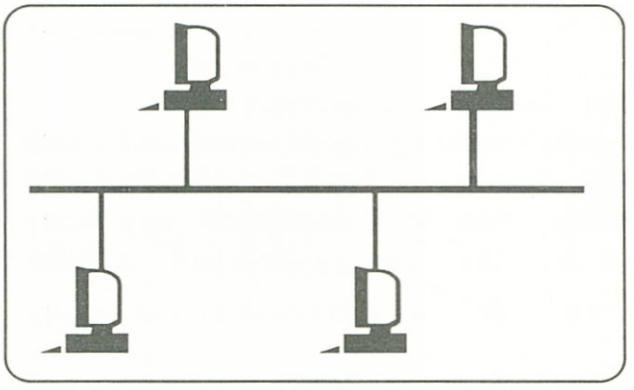
1. Token Ring (gyűrű) hálózat: Itt a munkaállomások gyűrűszerűen vannak egymás után fűzve, és az utolsó gép a gyűrűt bezárandó elsőhöz kapcsolódik. Az adat vagy üzenet folyamatosan halad a hálózaton, amíg el nem éri el a célállomást.



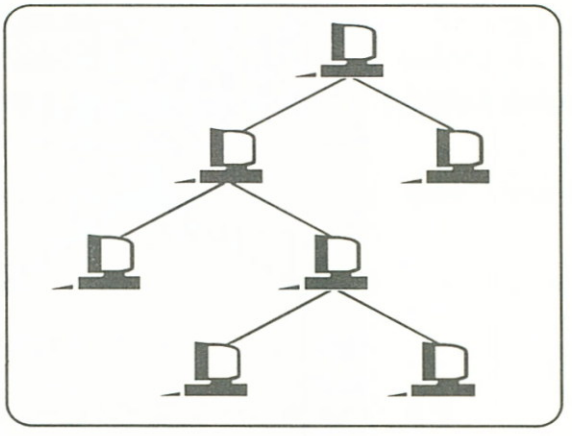
2. Csillag-elrendezésű hálózat (ARCNET): A munkaállomások csillag alakban helyezkednek el, közvetlenül a központi géphez kapcsolódva. Hátránya, hogy a hálózat kiépítése nagyon költséges.



3. Busz-hálózat (ETHERNET): Ebben a típusú hálózatban az állomások egy közös vezetékre vannak rákapcsolva. Mivel egyszerre csak egy adat tud a kábelen továbbítódni, ezért a munkaállomások között valamilyen rendszer szerint meg kell osztani az adási lehetőségeket.



4. Fa-hálózat: Minden munkaállomás hierarchikus rendben kapcsolódik egy v. több másik munkaállomáshoz. E hálózat hátránya, hogy egy munkaállomás kiesésével a nála alacsonyabb szinten lévő munkaállomások is kiesnek a rendszerből.



A hálózat központi egysége : A szerver

A szerver(ek) a hálózat koordinátora(i), minden hálózattal kapcsolatos munkának az ellenőrzői vagy éppen a végrehajtói. A szerver elosztja a perifériákat az egyes munkaállomások között, ezen kívül még az adatok minden munkaállomás által elérhető

tárolásának is eleget tesz. A hálózat operációs rendszerétől függően a szerver lehet kizárólagosan szerver feladatokat ellátó számítógép, vagy emellett önálló munkaállomásként is működhet. Az egyes hálózatok teljesítményét nagyban befolyásolja a szervergép teljesítménye. A számítógépes hálózatok nem létezhetnek összekapcsoló elemek és hardver tartozékok nélkül. Mégis, ha leülsz egy hálózatba kötött munkaállomás elé, ezek szinte nem is 'léteznek' számodra, csak a hálózati szoftverrel érintkezel. A szoftver a hálózat lelke. Alpvetően befolyásolja a hardver által nyújtott lehetőségek használatának módját és kihasználhatóságát is.

Hálózati operációs rendszerek

A hálózati operációs rendszer olyan szoftver, amely az általános kommunikációs szolgáltatáson túl további funkciókat (pl. erőforrások megosztása, elektronikus posta, hálózati névszolgáltatás) is nyújt. Nézzük meg részletesen a hálózati operációs rendszerek által nyújtott legfontosabb szolgáltatásokat.

Nyomtatószerver

A lokális hálózat egyik előnye, hogy képes megosztani a perifériákat. Ez különösen az olyan drága egységeknél fontos, mint pl. a lézernyomtató. A nyomtatószerver a hálózat valamennyi állomása számára lehetővé teszi, hogy egy nyomtatót használjon. Így a munkaállomásokon úgy használhatjuk a nyomtatót, mintha az közvetlenül a mi gépünkhöz lenne kötve. A szolgáltatáshoz tartozik még egy általános sorban állási nyilvántartás ami lehetővé teszi, hogy a nyomtatóra akkor is kiküldhessünk adatokat amikor a nyomtató éppen foglalt.

File-szerver

A nagy kapacitású lemez a másik erőforrás, amelyet a hálózat megoszthat. A file-szerver általában személyi számítógép, amely a nagy kapacitású lemezt kezeli. A megosztás megvalósitható könyvtár, file és lemezek alapján. Könyvtár alapján való megosztásnál a felhasználó egy adott könyvtárhoz, file alapján történő megosztásnál csak meghatározott file-okhoz férhet hozzá.

Elektronikus posta

Egyes hálózati operációs rendszerek a felhasználók egyszerű kommunikálását az elektronikus postával teszik lehetővé. Ennek a szolgáltatásnak a segítségével a felhasználó üzeneteket adhat fel, ill. fogadhat.

Hálózati névszolgáltatás

A hálózatba bekapcsolódó felhasználókat és a megosztott erőforrásokat a hálózati névszolgáltatás egy-egy névvel látja el, melyek alapján a felhasználók és az alkalmazói programok hivatkozahatnak rájuk. Azzal, hogy a neveket a megfelelő címekre lefordítja, a hálózati névszolgáltatás leveszi a szolgálatkérés megvalósításához szükséges helyes címzés gondját a felhasználó vagy a program válláról.

Összekapcsolhatóság

Az összekapcsolhatóság fogalma a lokális hálózatban a hálózat valamely állomásának egy a hálózaton kívüli személyi számítógéppel pl.:telefonvonalon keresztül létesített kapcsolatát foglalja magába, valamint egy vagy több más hálózattal történő kapcsolatot. Az egymással kapcsolódó hálózatok lehetnek azonos vagy különböző típusúak. Az összekapcsolódás megvalósítható közvetlen vagy távolsági úton is.

A NOVELL

Az alap Advenced NetWare operációs rendszer számos funkciót tartalmaz amit a lokális hálózatoknak tartalmazniuk kell. Pl.: file-megosztás, nyomtatómegosztás, elektronikus posta, távoli hozzáférés. Ez a hálózati operációs rendszer használható számos LAN-architektúrához.

File megosztás

A hálózatban dolgozó felhasználók lehetőséget kapnak a könyvtárak használatára ill. hozzáférésére. A felhasználók különféle jogokkal ruházhatók fel, melyek magukban foglalják a létrehozás, törlés, megnyitás, olvasás, keresés és módosítás jogokat. A rendszerszoftver lehetőséget nyújt arra, hogy akár egyetlen byte sorozatot is elzárhatóvá tegyünk. Ez a fajta 'titkolódzás' a hálózatok esetében létfontosságú, mert csak így lehet biztosítani azt, hogy a felhasználók csak a saját szemétdombjukon kapirgáljanak. A felhasználó egy adott file hozzáférési jellemzőit a file-attribútumok megváltoztatásával módosíthatja. A rendszer egészére is vonatkozik ez az elv, csupán az a különbség, hogy a rendszerhez tartozó fontosabb attribútumokat nem tudja mindenki állítani.

Nyomtatatómegosztás

A nyomtatómegosztás a nyomtatási feladatokat sorba állítja a nyomtatószerver kezelése alatt. A sorkezelés lehetőségei a nyomtatási munka törlését, a sor átrendezését, a munka felfüggesztését és folytatását, a sor különböző nyomtatókhoz való átirányítását jelentik.

Elektronikus postaszolgálat

A rendszer lehetőséget nyújt az elektronikus posta használatára, a felhasználók közötti üzenetek, jegyzetek, file-ok küldésére. Ezeket a különböző típusú dokumentumokat a rendszer segítségével akár egyszerre több felhasználónak is elküldhetjük. Lehetőség van arra, hogy ha levél érkezett, akkor a címzettet a képernyőn lévő üzenettel figyelmeztessük.

Előnyök és hátrányok

Bár a Novell nem a legjobb hálózati szoftver, mégis nagyon hatékony, mivel szinte a legelterjedtebb a maga nemében. Van néhány olyan funkciója amivel azonban az átlag fölé emelkedik.

A felhasználók osztályozása

A felhasználókat három csoportba osztjuk:

a) Rendszergazda az úgynevezett supervisor. Az ő hatáskörébe tartozik minden ami a rendszert érinti. Ilyen például a rendszer installálása, konfigurálása, programok karbantartása és működtetése, a felhasználók jogainak megállapítása. Ennek megfelelően a supervisornak mindenhez joga van a rendszerben.

b) Operátorok. Feladatuk a rendszer egy speciális elemének működtetése.

c) Felhasználók. Azon személyek, akik a hálózat szolgáltatásait veszik igénybe munkájuk végzéséhez. Jogaik úgy vannak megállapítva, hogy munkájukat maradéktalanul el tudják végezni, de mások munkáját még véletlenül se tudják zavarni.

Finomítások a jogokban

Sok szó esett már az úgynevezett jogokról. A rendszer e funkciója védelmet nyújt a felhasználóknak más felhasználókkal szemben, megakadályozva azt, hogy jogosulatlanul más területére tévedve akár véletlenül is kárt okozzanak. (Kivétel a supervisor, akinek mindenhez van joga.) A felhasználói jogok szabályozhatók a :

- bejelentkezési szabályok, jelszavak segítségével;
- különböző könyvtárakhoz tartozó elérési és kezelői jogokkal;
- file-okhoz való hozzáféréseket tartalmazó file attribútumok segítségével;

Mint a többfelhasználós operációs rendszerek álltalában, a Novell is rendelkezik a jelszavak használatának lehetőségével. Ez a gyakorlatban azt jelenti, hogy a felhasználónak a bejelentkezéskor meg kell adnia a nevét (username) és a jelszavát (password) ami alapján a rendszer megállapítja, hogy melyik felhasználói csoportba sorolható, milyen jogok illetik meg. A jelszavak természetesen titkosak, így az illetéktelenek számára a belépés csak álom marad. Azokban a hálózatokban, ahol adott számú terminálra viszonylag sok felhasználó jut sok gondot okoz a terminálok igazságos elosztásának problémája. Mivel a Novell általában ilyen területeken kerül felhasználásra (pl.:egyetemen) ezért felmerült a súlyozásos, pontozásos időosztás módszerének lehetősége. A módszer lényege, hogy a felhasználó státuszától függően egy adott pontszámot kap (adott időegységre), amiből a rendszer a munkával töltött idő alapján levonja a megfelelő számú pontot. A levonanadó pontszám függhet attól is, hogy milyen időpontban használta a rendszert (pl.:este olcsóbb lehet). A rendszer a pontszámmal nem rendelkező, ily módon illetéktelen felhasználót automatikusan kizárja a rendszerből.

Lehetőség van a könyvtáraknál szinte azonos jogok használatára, mint a felhasználóknál, azonban ezek az elérési jogok csak az adott könyvtárra vonatkoznak, a felhasználóktól függetlenül. Azt, hogy egy felhasználó egy könyvtárral mit tehet, azt a könyvtárra vonatkozó effektív jogok döntik el, amik a könyvtár elérési jogok és a kezelői jogok és kapcsolatából adódik.

Mint már a DOS-ban megszokhattuk, a Novell rendszerben is megtalálhatók a file-ok attribútumai. A Novell-ben ezek a következők:

-**R[ead]O[nly]/R[ead]W[rite]**: a file csak olvasható, vagy írható olvasható;

-**H[idden]**: rejtett file;

-**S[ystem]**: rendszer file;

-**SH[areable]/N[ot]Shareable**: a file megosztottan kezelhető egyszerre több felhasználó által, vagy sem.;

-**M[odified]**: a legutolsó mentés után módosult file;

-**I[ndex]**: a file egy adat-file index állománya;

-**E[xecute]O[nly]**: a file csak futtatható programot tartalmaz;

-**T[ransactional]**: a file a *Transactional Tracking System* kezelése alatt áll, melynek lényege, hogy az írások olvasások száma feljegyezhető.

Szoftver

Bevezetés

A Szoftver című fejezetben a számítógéphez használt szoftverekről olvashatsz. Szó lesz segédprogramokról, integrált programokról, grafikus programokról, szövegszerkesztőkről és a Windows-ról.

A segédprogramok közül a Norton Commander-ről, valamint kiegészítésként a Norton Utilities-ről, az integrált programokon belül a FrameWork-ről írtunk. Olvashatsz a CorelDraw!-ról, amely egy a Windows alatt működő grafikus program.

A CorelDraw! 2.00-ás verzióját ismertettük részletesen, míg az újabb verziójáról (a 3.00-ásról) kiegészítésként olvashatsz.

A Windows-t érdemes megtanulni, mert a DOS alá készült programok szinte kivétel nélkül elkészültek a Windows alá is, sőt; a Windows memóriakezelése miatt olyan programok is napvilágot láttak, amelyeket DOS alatt el sem lehetett volna képzelni.

Szoftver alapok

A szoftver a felhasználó és a hardver közötti kapcsolatot biztosítja. A programozó arra törekszik, hogy egy adott feladatot a szoftver segítségével minél gyorsabban, minél pontosabban oldjon meg.

A játékprogramok használói között találhatók tanárok és diákok, politikusok és kútfúrók, kismamák és nagymamák. Ez a tág felhasználói kör nem véletlen, hiszen a játékprogramok széles skálája vonzza a különböző korosztályokat.

A szövegszerkesztők lehetővé teszik a szövegek létrehozását, módosítását, blokkok kijelölését, mozgatását, másolását, a formázást, nyomtatást. A legtöbb szövegszerkesztőbe beépítettek néhány különböző betűtípust is (pl.: aláhúzott, *dőlt*, **vastag**), és néhány stíluslapot is. Kevés szövegszerkesztő megengedi, hogy grafikákat is beletehessünk a szövegbe.

Az adatbázis-kezelő programmal adatbázisban tudunk műveleteket végrehajtani. Adatbázis például az osztálytársaid adatai. A legkisebb egység itt külön az osztálytársad neve, külön a születési helye és külön az ideje lesz. Ezek lesznek a mezők. Egy osztálytársad itt használt adatai lesznek a rekordok. Két fő műveletet lehet az adatbázissal végezni: létrehozást és visszakeresést. A létrehozáskor megadjuk az adatbázis szerkezetét. A visszakereséskor a meglévő adatbázisból tudjuk visszakeresni az adatokat különböző szempontok szerint.

Az adatbázisok (A) tehát rekordokból (R) állnak. A rekordok viszont mezőkből (M). Egy mező eléréséhez tehát meg kell adni a rekordot és a keresett mezőt. Például: Kiss Pista születési hely.

Adatbázis									Ezen az ábrán látható, hogy a rekordok szerkezete mindig azonos.
Rekord			Rekord			Rekord			
Mező1	Mező2	Mező3	Mező1	Mező2	Mező3	Mező1	Mező2	Mező3	

Hasonló probléma például egy vállalat kiadásának és bevételének nyilvántartása. Itt egy táblázatszerű dolog jön létre. Ennek egyik oldalán a hónapok vannak felsorolva. Felül a bevétel, a kiadás és a múlt hónapról áthozott pénzösszeg van feltüntetve. Kis logikával rá lehet jönni, hogy a bevételt és kiadást nekünk kell beadnunk, de az áthozott mennyiséget ki lehet számítani. Erre való a táblázatkezelő program. Segítségével meg tudjuk adni, hogy a táblázat melyik pontja hogy kapja az adatokat. Például a február bevétele kézzel történik, de a februári átvitel a januári bevétel és kiadás különbségéből adódik.

A táblázatkezelő programok tehát elektronikus adatlapok létrehozására alkalmasak. Az adatlapok oszlopokra (pl.: bevétel) és sorokra (pl.: augusztus) vannak osztva.

Az egy oszlop és egy sor találkozásánál lévő helyet (pl.: novemberi kiadások) cellának nevezzük. A cellákba számokat, karaktereket és/vagy függvényeket helyezhetünk el.

A grafikus programok képek és tervek készítésére alkalmasak. A műszaki tervező programokat CAD-nek (**C**omputer **A**ided **D**esign) nevezzük.

A segédprogramok az operációs rendszer használatát könnyítik meg. A legelterjedtebb segédprogramok a Norton Commander + Norton Utilities.

Fontosabb dokumentum fájlok

Egy szoftver több fájlból állhat. Ezek egy része indítható, egy része a program további futásához szükséges. Ha több indítható fájl van, akkor nehezen dönthetjük el, hogy melyik mire való. Ezért vált szokássá, hogy a programok mellé egy előzetes tudnivalókat tartalmazó text fájlt tesznek. A gyors megtalálás érdekében READ.ME (OLVASS ENGEM!!) vagy OLVASD.EL nevet adnak ezeknek. Elolvasásukhoz szövegszerkesztőt vagy a DOS TYPE utasítását használd.

Általános utasítások

Miután elindítottunk egy programot, legfontosabb, hogy megtudjuk, mire való. Ez azért szükséges, mert el kell tudnunk dönteni, hogy mit várjunk a programtól. Ezt egyébként már a program mellett található dokumentum fájlokból megtudhatjuk.

A program és a felhasználó közötti kapcsolatot a parancsok, menük, ikonok teszik lehetővé. A parancsokat a billentyűzeten keresztül lehet bevinni a gépbe. Ma már kevés programot írnak így, mert kezeléséhez ismerni kell a parancsokat. Ilyen program a DOS is, de e mellé is írnak segédprogramokat (Norton).

Ezután alakultak ki a menüvezérelt programok. Ezeknél már a felhasználónak csak a menüpontokból kell választania. A menüpontok almenükbe vannak csoportosítva. Manapság már főleg menüvezérelt programokat használnak. A menü előhívására általában az F10 vagy az Alt gombokat használják.

A jobb grafikus lehetőségekkel együtt alakultak ki az ikonvezérelt programok. Ezeknél a programoknál már képeken ábrázolják az utasítást.

A leggyakrabban használt menüpont a HELP. Ebben a program használatához kapunk segítséget. Ezalatt a menüpont alatt találhatók a HELP INDEX és a HELP ON HELP parancsok. A HELP INDEX-ben kulcsszavak szerint vannak gyűjtve a help témái.

Ha olyan programot használsz, amiben a munkát fájlokban tárolja el a program, általában fájl kezeléshez is csinálnak menüt. Ennek a menünek a neve FILE (ki gondolná). A fájl megnyitásához az OPEN-t használd. új fájl létrehozása a NEW menüponttal lehetséges. Munkád elmentéséhez a SAVE vagy a SAVE AS parancsokat használd. A kettő között az a különbség, hogy a másodiknál a már meglévő fájlnév helyett új nevet lehet adni. Ha a munkádat abba akarod hagyni, de nem akarod elmenteni, akkor a CLOSE parancsot használd. Ebben az esetben a program megkérdezi, hogy elmentse-e a munkádat.

A FILE menüben található az EXIT vagy a QUIT parancs, ami a kilépéshez szükséges. Van egy speciális kilépési parancs, amit akkor célszerű használni, amikor csak egy kis időre akarsz kilépni a programból. Ez az utasítás a DOS SHELL vagy DOS PROMPT vagy DOS COMMAND. A programba visszatérni úgy lehet, hogy az EXIT utasítást adjuk ki.

Kérdések:

1., Mire való a HELP parancs és a READ.ME fájl?

2., Hogyan lépsz ki egy programból, ha csak rövid időre akarsz kilépni?

3., Mire való a SAVE AS?

4., Mi az EXIT és a QUIT parancs?

5., Mi a SAVE AS?

6., Milyen HELP funkciókat ismersz?

Norton programok

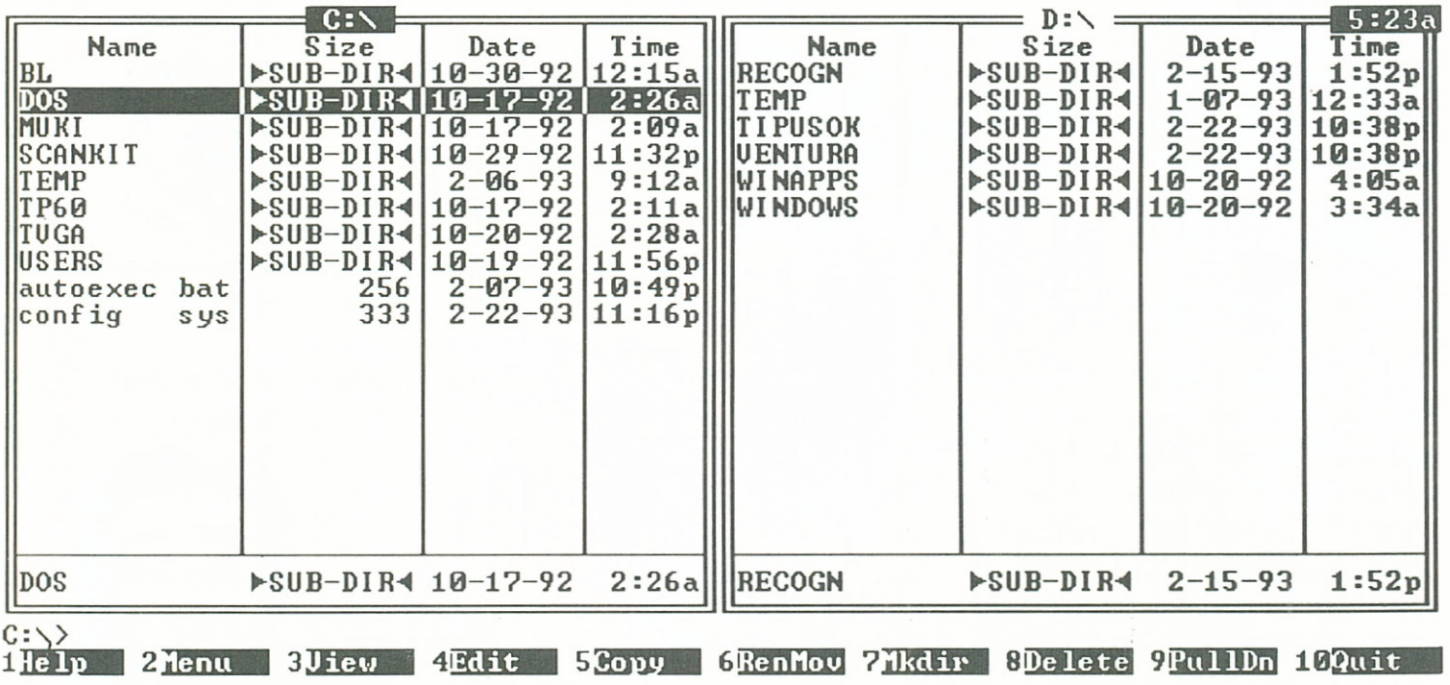
A forgalomban lévő Norton programok (Norton Commander, Norton Guide, Norton Antivirus, Norton Utilities, Norton Backup, Norton Editor és Norton Desktop for Windows) közül a Norton Commander-rel fogunk foglalkozni, a Norton Utiulities-t kiegészítésként találhatod meg.

Norton Commander

Bevezetés

A Norton Commander (a továbbiakban NC) egy csodálatos keretprogram, amely megkönnyíti a DOS kezelését. A DOS-t bármikor használhatod, te döntöd el, mikor és

milyen mértékben veszed igénybe a NC szolgáltatásait. A NC úgynevezett memória rezidens program, ezért miután elindítottál egy programot, a NC eltűnik a képernyőről, helyet adva az elindított programnak. Amikor kiléptél a programból, a NC újból látható lesz; dolgozhatsz tovább. Ennek az egyébként kényelmes megoldásnak van egy hátránya: mivel a NC a memóriában maradt és 'rá' töltötted a programodat, így alaposan csökkent a szabad memória mérete (kb. 15 kB-tal). Ezért mielőtt elindítasz egy programot, érdemes kilépni a NC-ből.



Programok elindítása, műveletek a fájlokkal

Mint már említettem, a NC elhagyása nélkül is el tudsz indítani programokat. Ezt megteheted az egérrel, de billentyűzettel is. Egérrel a bal gombot kell használnod: ha egyszer kattintasz rá a fájl nevére, akkor ráviszed a kurzort; ha viszont gyorsan kétszer kattintasz rá, akkor elindítod.

Használhatod a DOS-t is programok futtatásához: a képernyő alján az utolsó előtti sor egy majdnem üres sor: ide a DOS parancsokat lehet beírni (ez nem olyan kényelmes, de néha muszáj). Igaz, hogy DOS parancsot írtál be, de ettől még ugyanúgy a NC alól indítottad el, mintha egérrel vagy a kurzorral tetted volna. Ezt a sort szerencsére csak akkor kell használni, amikor egy program elindításához több paraméterre van szükség. A gyakran használt programokat be lehet írni a NC menüjébe (User Menu - F2), és ezek után nem kell mindig megkeresni a program indítófájlját, elég ha kétszer rákattintuk a menüben a kívánt fájlra, és Hopp!, elindul.

Szerencsére a NC készítői nem elégedtek meg ennyivel, ezért a NC két ablakában lévő fájlokkal sokféle műveletet végezhetsz: megnézheted a tartalmát (View - F3), írhatsz bele (Edit - F4), átmásolhatod (Copy - F5), átnevezheted (RenMov - F6), letörölheted 'őket' (Delete - F8), megváltoztathatod az attribútumát (Files - legördülő menük). Nemcsak a fájlokkal tudsz műveleteket végezni, hanem a könyvtárakkal is. Létrehozhatsz (MkDir - F7), átnevezhetsz (RenMov - F6), törölhetsz (Delete - F8) könyvtárakat (ez utóbbit csak akkor, ha nincs benne egy fájl sem).

Menü

A műveleteket háromféleképpen végezheted a fájlokkal, a könyvtárakkal és a meghajtókkal: használhatod a képernyő legalsó sorában lévő menüt, használhatod a legördülő menüt (F9) és használhatod a DOS parancssort is. Ebben a fejezetben az alsó menüsort tárgyaljuk. Ezeket a műveleteket a billentyűzet funkciógombjaival érheted el, vagy használhatod az egeret: kattints a kiválasztott menüre.

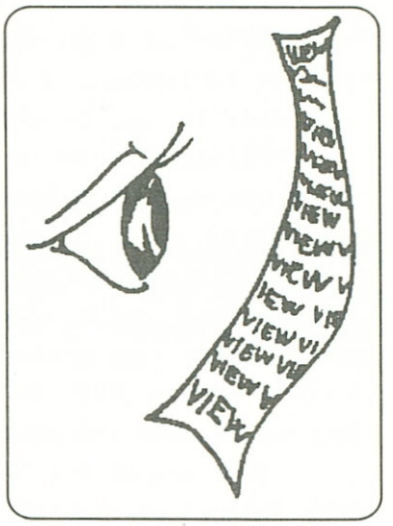
Help

Help (F1 billentyű): ezzel a funkcióval segítséget kérhetsz. A segítségben minden benne van, amit ennek a programnak a kezeléséhez tudni kell. Egyetlen hibája: a program készítői véletlenül angolul írták meg (az angol csak az angolul nem tudóknak okoz gondot, szóval ez nem is igazi hiba).

User Menu

Ezzel a funkcióval a saját legtöbbször használt programjaidat tudod elindítani. A NC legördülő menüjéből érheted el, vagy az F2 funkciógomb lenyomásával (vagy az egérrel, lásd Menü): ha a legördülő menüt választod, akkor a Files menüre kell kattintanod (Files). Utána az almenüpontok közül az User Menu-t kell választanod. Ekkor előjön egy ablak, ahol fel van sorolva néhány program. Ezek közül kell kiválasztanod a neked kellőt. Ezt is háromféleképpen teheted meg: billentyűzettel, a fel- és lefelé mutató kurzorgombokkal vagy a program elé írt betű egyszeri lenyomásával a billentyűzeten. Egérrel a program nevének sorára kell kattintanod gyorsan kétszer. Miután kiléptél az elindított programodból, az NC jelentkezik be újra, ezzel az ablakmenüvel.

View



Ezzel a funkcióval egy fájl tartalmát tudod megnézni anélkül, hogy elindítanád azt. Miután elindítottad ezt a menüpontot - az előzőekhez hasonlóan -, letörlődik a képernyő, és megjelenik a fájl tartalma. Ha ez a fájl egy elindítható program, akkor (kivétel .BAT) különböző ASCII kódokat látsz. Néha még az el nem indítható fájlokat sem tudod elolvasni: ezek a futtatható programokhoz kapcsolódnak, és csak a gép tudja ezeket a fájlokat 'elolvasni'. Az alsó menüsor ilyenkor megváltozik: az F1 gombra marad a Help, az F2 gombra Unwrap/Wrap felirat lesz kiírva, az F3 gomb nem működik, az F4 gombra Hex/ASCII szöveg íródik ki, az F5 és az F6 nem működik, az F7 Search, az F8 Viewer feliratú lesz, az F9 nem működik és az F10 továbbra is Quit marad.

Az F2 gombbal ilyenkor azt állíthatjuk be, hogy a szöveget 80 karakter hosszan lássuk (Wrap), vagy teljes hosszában (Unwrap). Ha a teljes hosszt választottuk, akkor a jobbra és a balra mutató kurzorgombokkal nézhetjük meg a fájlt, vagy az egérrel a képernyő közepén a jobb vagy a bal oldalra kattintunk a bal gombbal. Ha a fájl nem fér ki egy képernyőre, akkor a fel- és lefelé mutató

kurzorgombokkal tudod görgetni a képernyőt. Egérrel a képernyő második és utolsó előtti sora közé kell kattintanod. A bal felső sarokban ez van kiírva: Text View: [a fájl elérési útja] Col 0 [a fájl hossza] 100% (hol tartasz a fájlban). Ezek közül a Text View (lásd később az F8 funkciónál), a Col 0 (Column = oszlop szó rövidítése) és a 100% (hol tartasz a fájlban) szokott megváltozni.

Az F4 billentyűvel azt állíthatod be, hogy milyen formátumban lásd a fájlt. A két lehetőség: Hex és ASCII. Ha a Hex van beállítva, akkor a jobb oldalon a fájl karaktereit látod, míg a bal oldalon a karakterek hexadecimális értékeit. Ha az ASCII van beállítva, akkor csak a fájl karaktereit látod, az értéküket nem.

Az F7 billentyű hatására egy ablak jelenik meg, ahol üresen van hagyva egy sor. Ide a keresendő szót, mondatot, karaktert írhatod be. Az Enter gomb megnyomása után elkezdi az előbb beírt szöveget keresni a fájlban. Minél hosszabb a fájl, annál több ideig keres. Ha talált, akkor a keresett szöveget kiemelten írja át. Ha nem találta meg a szövegedet, akkor angolul értesít.

Az F8 billentyűvel a különböző szövegszerkesztők által létrehozott fájlokat is megnézheted, anélkül, hogy azt a szövegszerkesztőt elindítottad volna.

Az F10 billentyűvel kiléphetsz, miután megnézted, mi van a fájlban.

Edit

Ez a funkció egy kezdetleges szövegszerkesztő, amellyel kisebb (kb. 26 kB-os) fájlokat szerkeszthetsz át. Ezt a szövegszerkesztőt kicserélheted a saját kedvenc szerkesztődre (Options - legördülő menük). Itt a képernyő tetején a bal oldalon a fájl elérési útját láthatod, mellette a sor és az oszlop számát, ahol a kurzor áll, ezek mellett a még szabadon felhasználható helyet, és végül annak a karakternek az ASCII kódját, amelyiken a kurzor áll. A Shift és az F4 billentyű lenyomására egy új fájlt tudsz kezdeni.

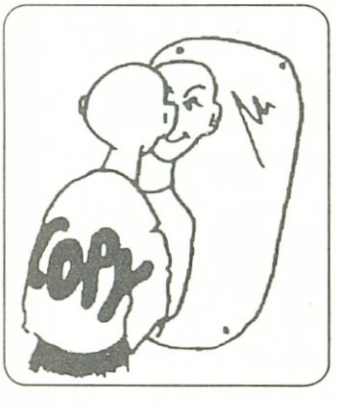
Itt a szövegszerkesztőben az F1 billentyűre szintén segítséget kapsz.

Az F2 funkciógombra el tudod menteni a fájlt a lemezre.

Az F7 gombra az előző fejezethez hasonlóan egy karaktert, szót vagy egy szöveget kereshetsz a fájlban.

Az F10 gombra kiléphetsz a szövegszerkesztőből (természetesen az előzőekben leírtak az egérrel is elérhetők).

Copy



A NC e funkciójával egy fájlt vagy kiválasztott fájlokat tudsz átmásolni egyik könyvtárból a másikba. Ha az egyik fájlra ráállítod a kurzort (billentyűzetet vagy az egér bal gombját használva) és lenyomod az F5 funkciógombot, akkor előjön egy ablak, amelyben ki van írva az átmásolandó fájl neve és - ha a NC jobb (vagy a bal oldalán; attól függ, hogy melyik oldalról akarsz másolni) egy másik könyvtár van megadva - egy elérési út. Ha ez az út neked nem felel meg, akkor ezt át kell írnod. Miután megvan, le kell még nyomnod az Enter-t és (a fájl hosszától függően több-kevesebb idő alatt) a fájlt átmásolja az egyik könyvtárból a

másikba. Több fájlt az egér jobb gombjával vagy a billentyűzet Ins (Insert) gombjával jelölhetsz ki. Ilyenkor a NC a fájlokat egymás után átmásolja, amíg kész nem lesz vagy meg nem állítod (Ctrl + Break, Ctrl + C, ESC).

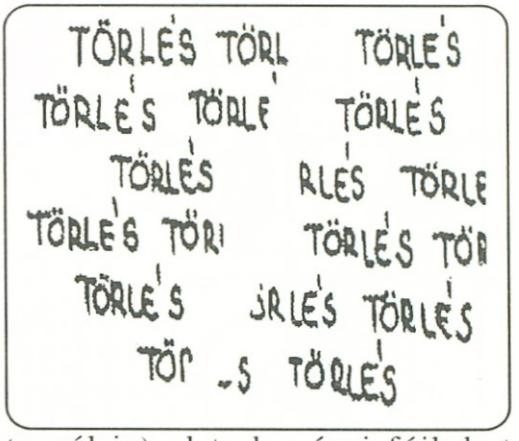
RenMov

Ennek a funkciónak a jelentése átnevezés vagy mozgatás (az angol RenMov Rename or move szavakból). Tehát ezzel a gombbal fájlt vagy könyvtárakat tudsz átnevezni vagy fájlokat tudsz átvinni egyik könyvtárból a másikba (ez nem azonos a másolással, mert csak az átvitt fájl fog létezni, az eredeti nem!). Ha egy fájl van kijelölve vagy csak rajta áll a kurzor, akkor eldöntheted, hogy átnevezni akarod vagy átmozgatni. Ha több fájl van kijelölve, akkor csak átmozgatni tudod (az összes fájlt a billentyűzet szürke + gombjával tudod kijelölni, az összes fájlt kijelöletlenné tenni a szürke - gombbal). A könyvtárakat csak átnevezni tudod, és azt is csak egyesével. Miután megnyomtad az F6 gombot, egy újabb ablak jön elő. Ebbe az ablakba - úgy mint a másolásnál -, be kell írnod azt az elérési utat ahová a fájlt vagy fájlokat mozgatni akarod. Ha az átnevezést választottad, akkor ide nem elérési utat, kell írnod, hanem azt a könyvtárnevet vagy fájlnevet amivé át akarod nevezni a könyvtárat vagy a fájlt. Mint a másolásnál, itt is ha a kívánt könyvtárban már létezik(nek) a fájl(ok), akkor a program megkérdezi, hogy rá akarod-e venni az új fájlt a régire.

MkDir

Ezzel a funkcióval egy könyvtárat hozhatsz létre. Miután lenyomtad az F7 gombot, megjelenik egy ablak, amibe neked be kell írnod a könyvtár nevét (max. 8 karakter hosszú lehet, de ha a név után pontot teszel, akkor még három karaktert is beírhatsz, mintha kiterjesztés lenne).

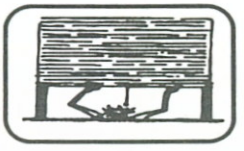
Delete



Az F8 Delete gombbal egy fájlt, egy könyvtárat vagy kijelölt fájlokat tudsz letörölni. Ha csak egy fájlt akarsz letörölni, akkor nem kérdez meg semmit, hogy akarod-e vagy nem; letörli (vissza lehet állítani [Norton Utilities]). Ha egy könyvtárat akarsz letörölni, akkor szintén úgy csinálja, mint az egy fájl törlésénél (könyvtárat is vissza lehet állítani, lásd Norton Utilities): kérdés nélkül letörli. Kivétel persze, ha a letörlendő könyvtár nem üres: ebben az esetben előbb a benne lévő fájlokat kell letörölni. Több fájl letörlése esetén már óvatosabb: visszakérdez, hogy biztos vagy-e benne. Ennél a funkciónál is (és a Copy, RenMov, Delete-nél is) el tudsz érni fájlokat anélkül, hogy belépnél abba a könyvtárba, ahol az a fájl található. Ezt a funkciót a Shift + F8 (másolásnál a Shift + F5, mozgatásnál a Shift + F6) gombokkal érheted el. Ekkor egy olyan ablak jön elő, amibe a törlendő fájl nevét kell beírni, elérési úttal

együtt. (Másolásnál először a másolandó fájlt kell beírni (elérési úttal együtt), utána pedig azt, hogy hova akarod másolni a fájlt; mozgatásnál ugyanígy kell eljárni, persze itt másolás helyett mozgatást kell érteni.)

PullDn



Az F9 gombbal érheted el ezt a funkciót, vagy úgy, hogy a legördülő menüre kattintasz egyet az egér gombjával. Ezt a funkciót a Legördülő menük fejezetben találhatod részletesen.

Quit

Az F10 billentyű hatására kiléphettek ebből a programból (nem érdemes).

Legördülő menük

Legördülő menü

Az elnevezés az angol Pull Down (lehúzható) szóból ered (van rá másik kifejezés is: redőny menü (Brrr..). Ezek a menük az F9 billentyű lenyomására aktivizálódnak, de ugyanúgy működnek akkor is, ha az egér bal gombjával egyszer rákattintunk a menü szövegére. Néhány almenüpontot úgy is elérhetünk, hogy lenyomunk egy billentyűkombinációt (pl.: Shift F9 - Save setup), ezeket a gombokat majd a funkciójának ismertetésénél találhatod meg.

Left

Ebben a részben a bal oldali ablakra vonatkozó beállításokat olvashatod.

. *Brief* - ha ezt a pontot választod, akkor csak a fájlnevek és a kiterjesztéseik jelennek meg, de három oszlopban. így több fájlt tudsz egyszerre megnézni, nem kell hozzá keresgélni sokáig. Ezt a pontot a 'b' betű lenyomásával is elérheted (amennyiben a Left menü le van engedve).

Full - ez a pont a fájlneveket, a kiterjesztésüket, a hosszukat (könyvtárak esetén a SUB-DIR vagy az UP-DIR feliratokat), a keletkezési (vagy az utolsó változtatási) dátumukat és idejüket írja ki a képernyőre. Ezt a pontot, ahogy az előbb, az 'f' betűvel is előhozhatod.

Info - ez a pont a jobb oldali ablakról ad információt (teljes kapacitását, a még szabad helyet). Ehhez a ponthoz az 'i' betűt használhatod.

Tree - ez a pont szintén a jobb oldali ablakhoz kapcsolódik: a könyvtárakat faágszerűen mutatja meg, amelyben mozogni is tudsz a könyvtárak között a kurzorgombokkal. Elég kétszer a kívánt könyvtárra kattintani az egérrel, és a jobb oldalon meg is jelent a könyvtár tartalma. Ehhez a ponthoz az 't' betűt használhatod.

Quick View - ez a funkció letörli az ablakot a bal oldalon, majd miután rákattintottál az egér gombjával a kívánt fájlra, a fájl tartalma megjelenik a bal ablakban. Ez különösen akkor hasznos, ha egy szövegfájlt keresel, de nem tudod a nevét, és nem akarod elindítani a szövegszerkesztőt. Ehhez a ponthoz a 'v' betűt használhatod.

Link - ezzel a funkcióval két számítógépet lehet összekapcsolni. Ehhez a ponthoz a 'k' betűt használhatod.

On/Off - ezzel a funkcióval ki-be kapcsolhatod a bal oldali ablakot. Ehhez a ponthoz az 'o' betűt vagy a Crtl-F1 billentyűket használhatod.

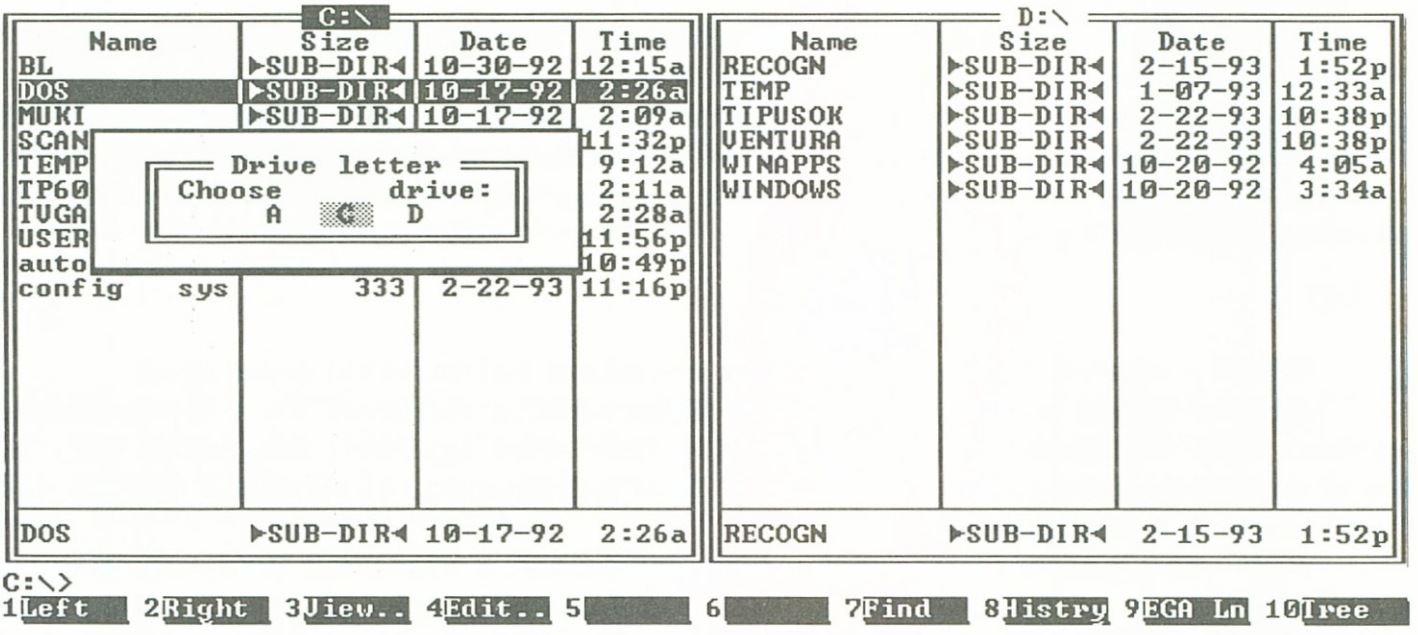
A következő öt ponttal a fájlok kiírási sorrendjét tudod megváltoztatni:

Name - névsorban ('n' betű);
Extension - kiterjesztések szerint ('x' betű);
Time - keletkezési idő szerint ('m' betű);
Size - méretük szerint ('s' betű);
Unsorted - sorrend nélkül ('u' betű).

Re-read - ha időközben változtattál valamit a könyvtárban, akkor nem biztos, hogy már ki van írva. Ha nincs, akkor használhatod ezt a funkciót ('r' betű vagy Crtl-R).

Filter - itt kiválaszthatod, hogy milyen fájlok kerüljenek kijelzésre ('l' betű).

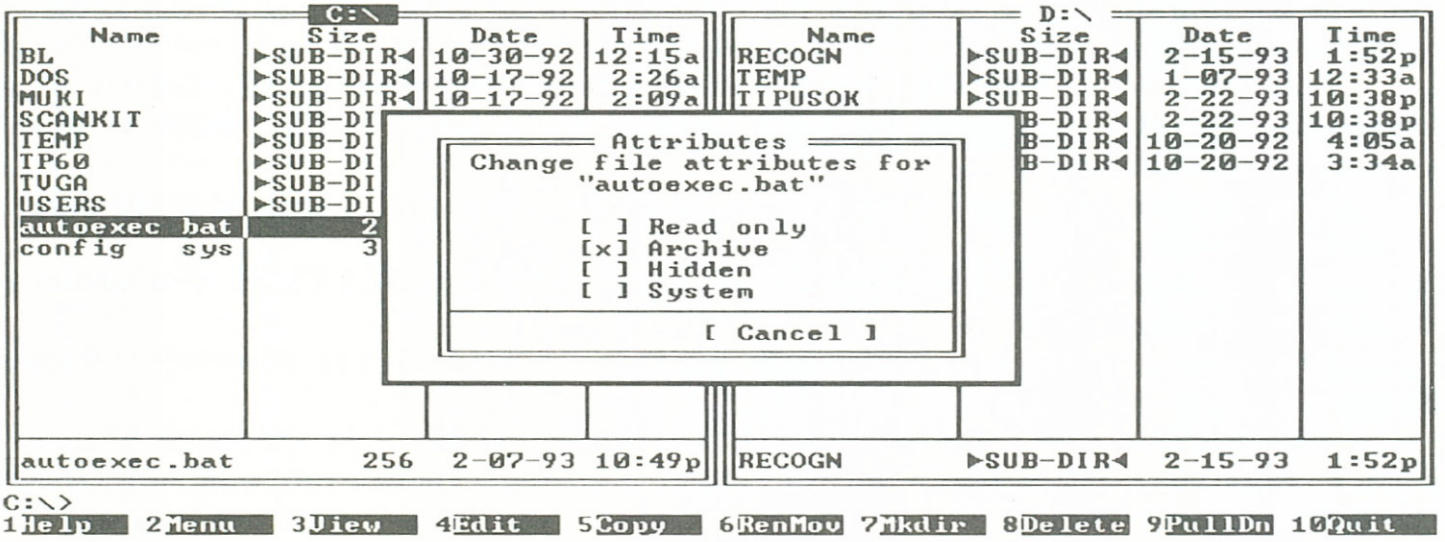
Drive - itt kiválaszthatod, hogy melyik meghajtót akarod használni ('d' betű vagy Alt-F1).



Files

Ebben a menüben az első nyolc és az utolsó funkció már ismerős lehet, ezeket nem tárgyaljuk újra. A négy új pontot alább olvashatod.

File attributes - ('a' betű) ha a kurzor egy fájlon áll (nem egy könyvtáron), akkor annak a fájlnak megváltoztathatod a típusát: csak olvasható (read only), rejtett (hidden), rendszer (system), arhív (archive). Ezek közül az első három különleges: a csak olvasható és a rendszer típusú fájl amikor le akarjuk törölni, még visszakérdez, hogy valóban komolyan gondolod-e a letörlését. Ez azért fontos, mert így elkerülhető a védett fájlok véletlen letörlése. A rejtett fájlt csak akkor látjuk, ha a Configuration menüpontnál beállítjuk a Show hidden files gombot 'be' állásba.



Send files - ezzel a funkcióval ('s' betű) fájlokat küldhetünk át a másik számítógépre (ha össze vannak kapcsolva).

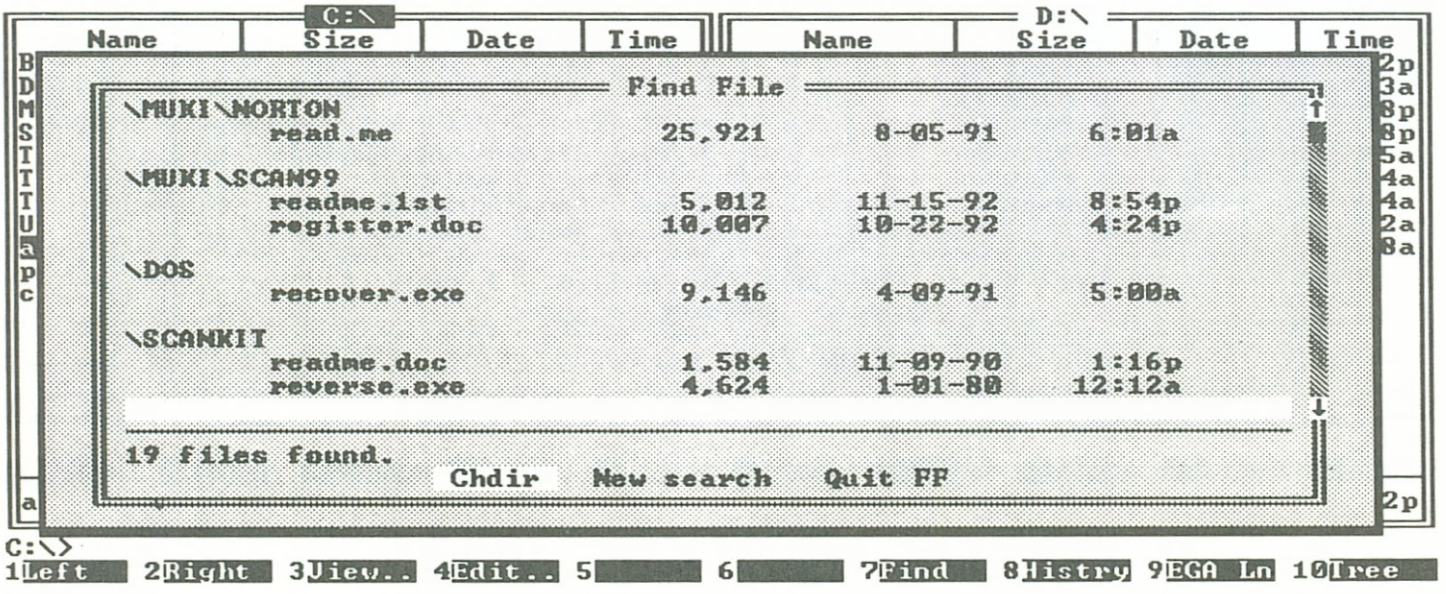
Select group - ('g' betű vagy az ún. szürke plusz (+) gomb a numerikus billentyűzeten) ezzel a funkcióval a fájlokat csoportosan kijelölhetjük (csak kiterjesztés szerint). Ha csak néhány különböző fájlt akarunk kijelölni, akkor az Insert gombot kell használnunk.

Unselect group - ('n' betű vagy a numerikus billentyűzeten lévő mínusz (-) gomb) ezzel a funkcióval a kiválasztott fájlokat kiválasztatlanná tehetjük.

Commands

NCD Tree - ('n' betű vagy Alt-F10) ezzel a funkcióval egy ablak jön elő, amiben egy fastruktúrájú könyvtárrendszert láthatsz. Itt könyvtárat válthatsz, úgy mint a Tree menüpontban.

Find file - ('f' betű vagy Alt-F7) ezzel a menüponttal egy fájlt kereshetsz az aktuális meghajtón.



History - ('h' betű vagy Alt-F8 vagy Ctrl-E) ezzel a menüponttal az eddig DOS-ból kiadott parancsaidat hívhatod elő. A megjelenő ablakban elég, ha kétszer rákattintasz a kívánt parancsra és azt a gép újból végrehajtja.

EGA lines - ('e' betű vagy Alt-F9) ezzel a funkcióval több adatot irathatsz ki a képernyőre, mivel a karakterek kisebbek lesznek (ezt a funkciót csak EGA és VGA képernyőn használhatod).

Swap panels - ('s' betű vagy Crtl-U) ezzel a menüponttal a jobb és a bal oldali ablakokat cserélheted fel.

Panels On/Off - ('p' betű vagy Crtl-P) ezzel a funkcióval az éppen nem aktuális ablakot tudod kikapcsolni.

Compare directories - ('c' betű) ezzel a funkcióval két könyvtárat tudsz összehasonlítani. Ez csak a különböző fájlokat választja ki.

Send/Receive mail - ('r' betű) ezzel a menüponttal üzenetet küldhetsz a másik gépnek, vagy visszakeresheted a hozzád küldött üzeneteket (ennek feltétele, hogy a két gép össze legyen kötve).

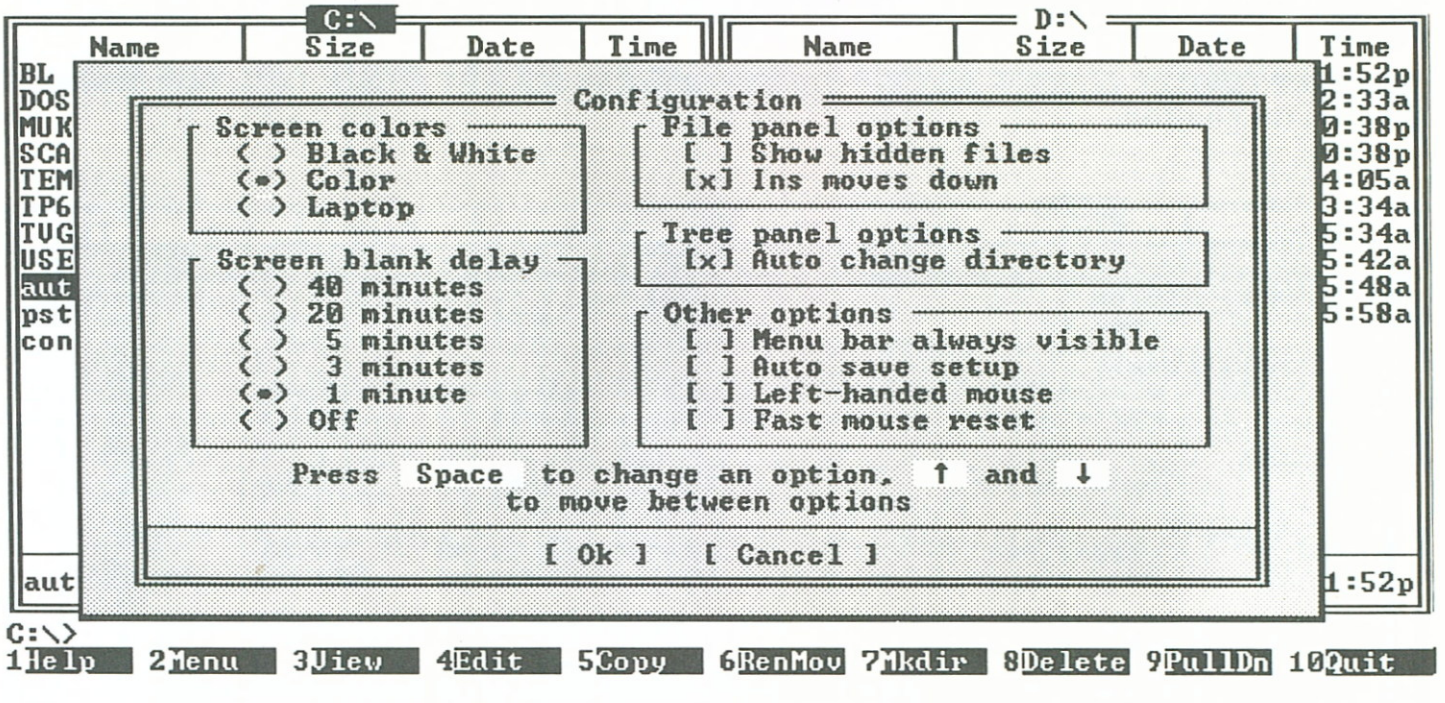
Commander mail - ('l' betű) ez a funkció a postázó funkció.

Menü file edit - ('m' betű) a Norton Commander elindításakor megjelenő menübe beírhatod a programokat, így két egérgomb kattintással elindíthatod a programot.

Extension file edit - ('x' betű) megadhatod a gépnek, hogy bizonyos kiterjesztésű fájlokat mivel együtt indítson el (pl.:ha van egy Word szövegfájlod, akkor az automatikusan elindul a Word programmal együtt, miután kétszer rákattintottál a szövegfájlra).

Options

Configuration - ('c' betű) ezzel a funkcióval beállíthatod, mennyi idő múlva álljon pihentető módba a képernyő, láthatóak legyenek-e a rejtett fájlok, stb.



Editor - ('e' betű) itt beállíthatod, hogy az F4 billentyű hatására a Norton szövegszerkesztője jelenjen-e meg, vagy egy általad kiválasztott.

Auto menus - ('a' betű) ezzel a menüponttal azt állíthatod be, hogy amikor elindítod az NC-t előjöjjön-e a menü ablak a programjaid nevével. Ez is mint az alább látható öt menüpontnál, akkor van engedélyezve, ha a menüpont előtt egy gyökjel van.

Path prompt - ('p' betű) ezzel a menüponttal a DOS parancssorban lévő könyvtár elérési utat tudod eltüntetni, vagy ha nincs ott, előhozni.

Key bar - ('k' betű vagy Crtl-B) itt a legalsó sorban lévő menüt tudod eltüntetni, vagy előhozni.

Full screen - ('f' betű) ezzel a funkcióval teljes képernyőssé teheted a NC-t, vagy ha teljes képernyőn van, akkor félképernyőssé kicsinyítheted le.

Mini status - ('m' betű) az ablakok alsó sorában lévő kijelzőt tüntetheted el, vagy hozhatod elő.

Clock - ('l' betű) a jobb felső sarokban lévő órát tudod eltüntetni, vagy előhozni.

Save setup - ('s' betű) a Configuration menüben beállított pontokat tudod elmenteni.

Right

Itt nem sok új funkciót lehet felsorolni, a legtöbbet már elolvashattad a *Left* menüpontnál. Itt csak a változásokról lesz szó.

On/Off - ezt a funkciót itt nem a Ctrl-F1 gombbal hívhatod elő, hanem Ctrl-F2-vel.

Drive - itt ugyanúgy választhatsz meghajtót, de az Alt-F2-vel.

Framework

Az integrált programcsomag előnye

A Framework egy integrált programcsomag. Amelynek előnyeit akkor tudod kihasználni, ha kapsz például egy olyan feladatot, hogy készíts kimutatást- táblázatos formában -osztálytársaid tanulmányi eredményéről és írj hozzá párszoros értékelést.

Ehhez a Framework használata nélkül először készítened kellene egy táblázatot egy táblázatkezelő programmal. Majd elő kellene venned egy szövegszerkesztő programot, amiben megírhatod azt a párszoros szöveget (amit a táblázatkezelőben nem tudnál).Bonyolult és fáradtságos a feladatot így megoldani, mert két program alapos ismeretén kívül több idődbe kerül, mintha ezeket egy program keretein belül végeznéd el.

Tehát szükséged van egy olyan programra, ami egyszerre tud például táblázatot is készíteni és bele is tudja ezt illeszteni egy szövegkörnyezetbe ugyanazon programon belül.Ezt valósítja meg az integrált programcsomag, így a Framework is.

Felmerülhet benned ama költői kérdés, hogy akkor minek vannak még egyáltalán külön táblázatkezelő, szövegszerkesztő, adatbázis-kezelő programok?! Ennek az a magyarázata, hogy ha az integrált programcsomag egyetlen szolgáltatását akarod kiragadni a többi közül, akkor azt az arra specializált programban jobban meg tudod oldani. Például ha csak a Framework szövegszerkesztőjét akarod használni, akkor hamar megkötöttnek éreznéd magad, mert meg sem közelíti a WORD színes lehetőségeit. Mihelyt viszont összetettebb feladatot kell megoldanod, akkor már az integrált programcsomagok használata kerül előtérbe.

A kezdők és haladók számára is nagyon jó ez a program, mert könnyen kiismerhető, logikus felépítésű és van magyar változata is. Már kis tudással és némi gyakorlással nagyon érdekes dolgokat tudsz majd csinálni. Az a legjobb, ha nem csak azokat a lehetőségeket valósítod meg, amit itt leírunk, hanem minél több saját ötletet is kipróbálsz. A program válaszol és informál téged, csak figyelned kell a munkaterület alján lévő sáv középső részét (mert ez három részre van osztva két függőleges vonallal) vagy a legalsó sort.

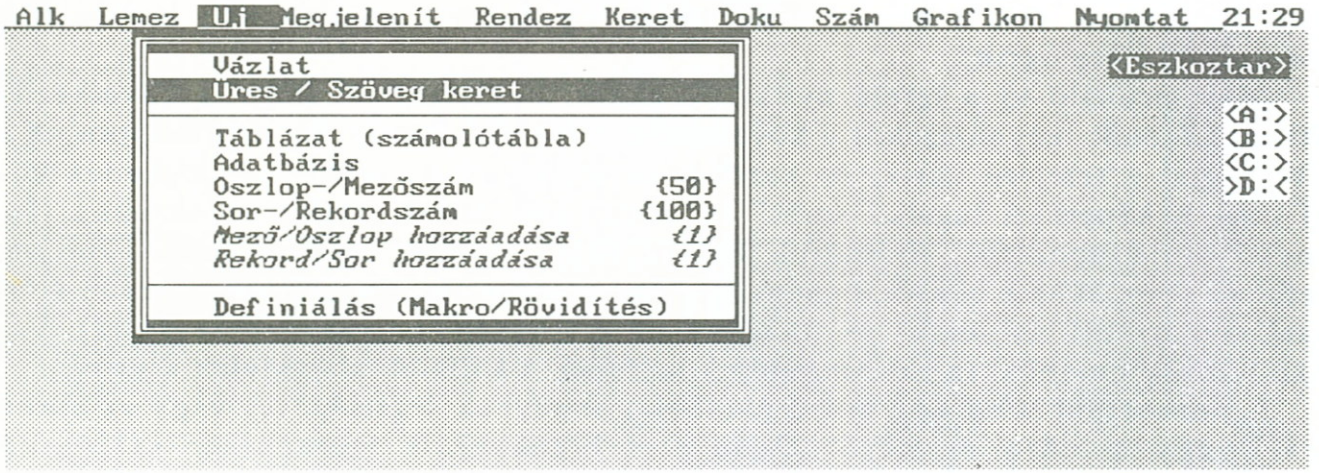
Ha ez a pár szavas információ nem elég, akkor csak az F1 billentyűt kell lenyomnod, és máris bővebben olvashatsz róla, ez a HELP. Kellemes dolog, hogy ez a szolgáltatás nem olyan, mint egy könyv, amiben folyton mindent keresni kell, hanem ezt a keresést a számítógép végzi. A képernyőn kis vonallal, úgynevezett kurzorral jelölt témáról ad bővebb információt. A kurzor a billentyűzetről nyilakkal (fel, le, jobbra, balra) mozgatható a képernyőn. (Ezzel mutathatjuk meg a programnak, hogy hol vagyunk és mit akarunk csinálni).

A program indítása

A Framework indítása rendkívül egyszerű. A gép bekapcsolása és bejelentkezése, (ami általában C> alakban történik ,a C betű helyett más betű is állhat) után a következő a feladatod, hogy utasítást adj a Framework betöltésére, amely a program nevének rövidítésével történik. Tehát írd be: FW, és nyomd le az ENTER-t. Az ezután megjelenő

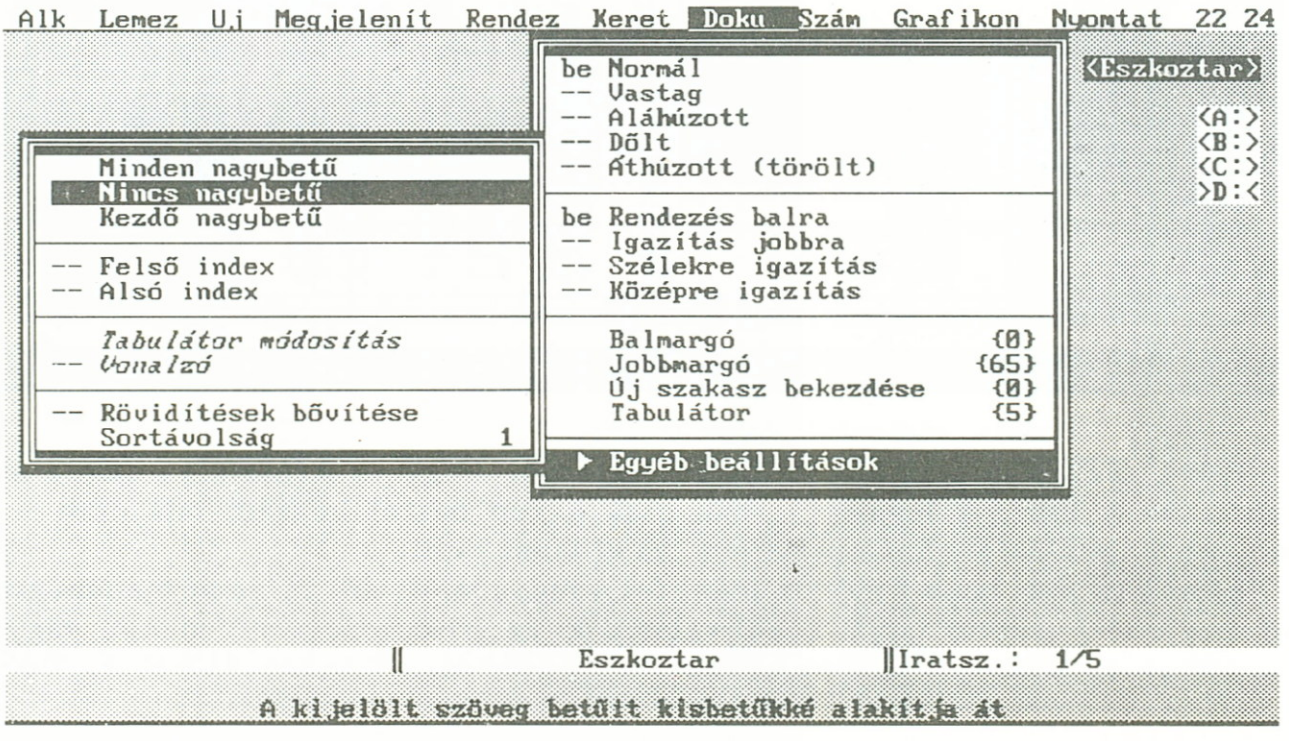
képernyőn láthatod a Framework feliratot.Ezután már csak a program utasításait kell végrehajtanod (vagyis egy szöveg megjelenése után ENTER-t ütni). És ha a program utasít rá, még az ESC-et is. És máris a rendelkezésedre áll a Framework.

A legelső sorban szavakat látsz. Ezeket fogjuk menüpontoknak, az egész sort pedig menünek nevezni. Ez a hely, ahol a programban használható legáltalánosabb beállítások és utasítások vannak. Az UJ menüpont előhívásakor ezt látod:



Ugyanígy a képernyő alján is látható egy csík, ami három részre van osztva. Ezt állapotsornak hívjuk, az alatta lévő sorban olvashatod azokat az üzeneteket, amelyeket a gép ad számodra (ez található középen, a másik két résszel később foglalkozunk).

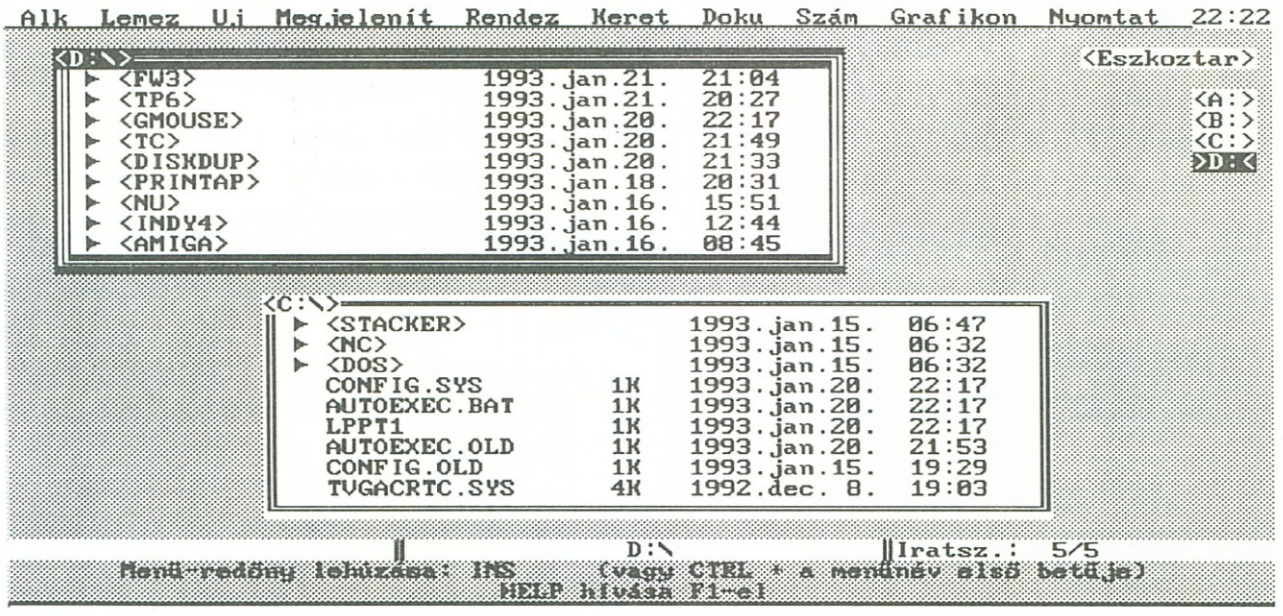
Az igazán érdekes rész a képernyő közepén található, az a mi munkaterületünk, ami egy íróasztalra hasonlít. Máris otthonosabban érezheted magadat, ha nemcsak látod, hanem ki is próbálod ezeket a területeket. Először is el kell jutnod a kiszemelt helyre, például a már említett menübe (legfelső sor). A közlekedéshez billentyűket kell lenyomnod, ebben az esetben például az INSERT gombot.



Itt már nemcsak egy menüsort látsz, hanem egy alatta lévő téglalapot is, amiben különböző szavak vannak. Ezt fogjuk almenünek nevezni, mert a menüsorban található szavak mind egy-egy csoportot jelölnek. Ezeket a szavakat főmenüpontoknak hívjuk. Ha kiválasztasz egy főmenüpontot, akkor megjelennek a csoport tagjai egy téglalapban felsorolva (vagyis az almenü). Próbáljuk ki, hogy ez a többi menüpontnál is így van-e. Ehhez újabb billentyűlenyomás szükséges, mégpedig a nyilakat ábrázoló gombok közül a jobbra mutatót, vagy ha visszafelé akarsz haladni, akkor a balra mutató nyílgombot kell megnyomnod. (Ha a végére értél, akkor ugyanezt a gombot kell lenyomnod, és akkor a sor elejére ugrik).

Most próbáld ki a lefelé nyilat, és akkor megláthatjuk, hogy a következő sort emeli ki, ismételt lenyomásakor a következőt, a legalján pedig, ha újra lenyomjuk ezt a gombot, újra a tetejére ugrik, és kezdhetjük elölről a műveletet. A lépkedés közben nézd meg a képernyő legalsó sorát! Itt némi ismertetést találsz az aktuális (vagyis kiemelt) almenüpontról. Próbáljuk ezt is végig minden menüpontnál. Ha nem a következő sorra akarsz ugrani, hanem az előzőre, az is megvalósítható. Csupán a felfelé mutató nyilat kell lenyomni. A menüből az ESC billentyűvel (bal legfelső), vagy a szürke mínusz gombbal (a jobb oldalon a legfelső, a LED-ek alatt) lehet kijutni.

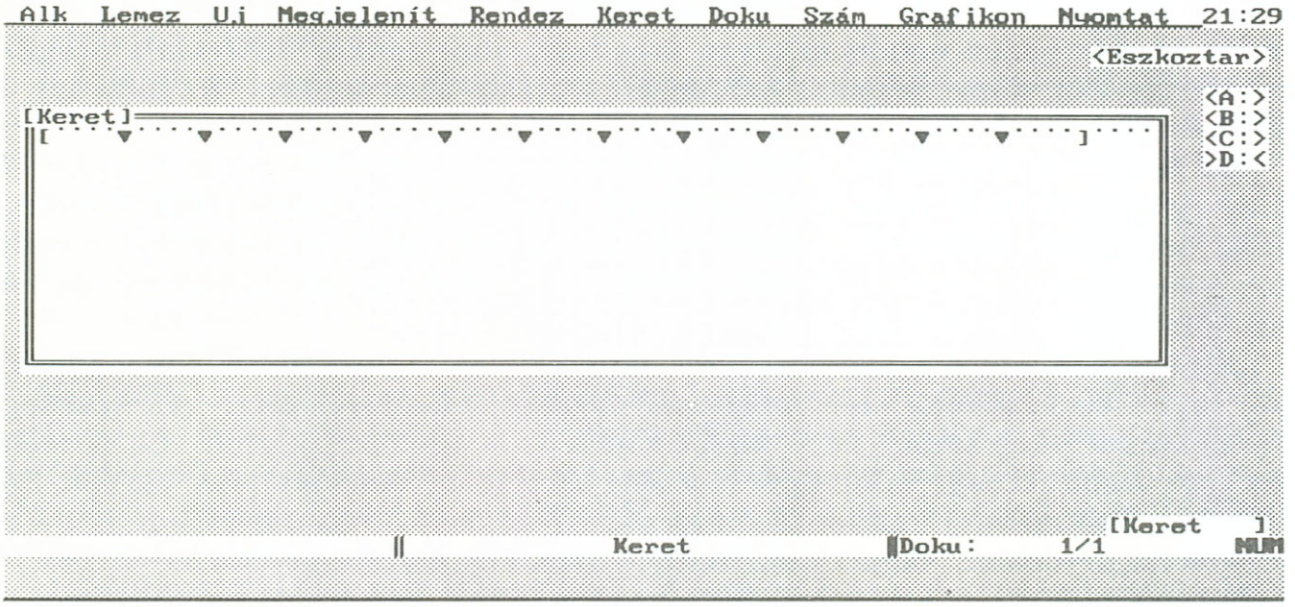
Ismerkedjünk meg egy újabb gombbal, amelynek segítségével vagy az eszköztár felirat, vagy a <A:> és <C:> (ilyen más betűkkel is szerepelhet <D:>) között közlekedhetünk. Tehát nyomjuk meg a Scrool Lock-ot, majd vagy a lefelé, vagy a fölfelé nyílgombot, és láthatjuk, hogy mindig vált a fényesen világító terület. Most próbáljuk ki ezt is. Láthatjuk, hogy a téglalapok közül egy mindig más színűvé válik. Ebből tudhatjuk, hogy most éppen hol tartunk. Például:



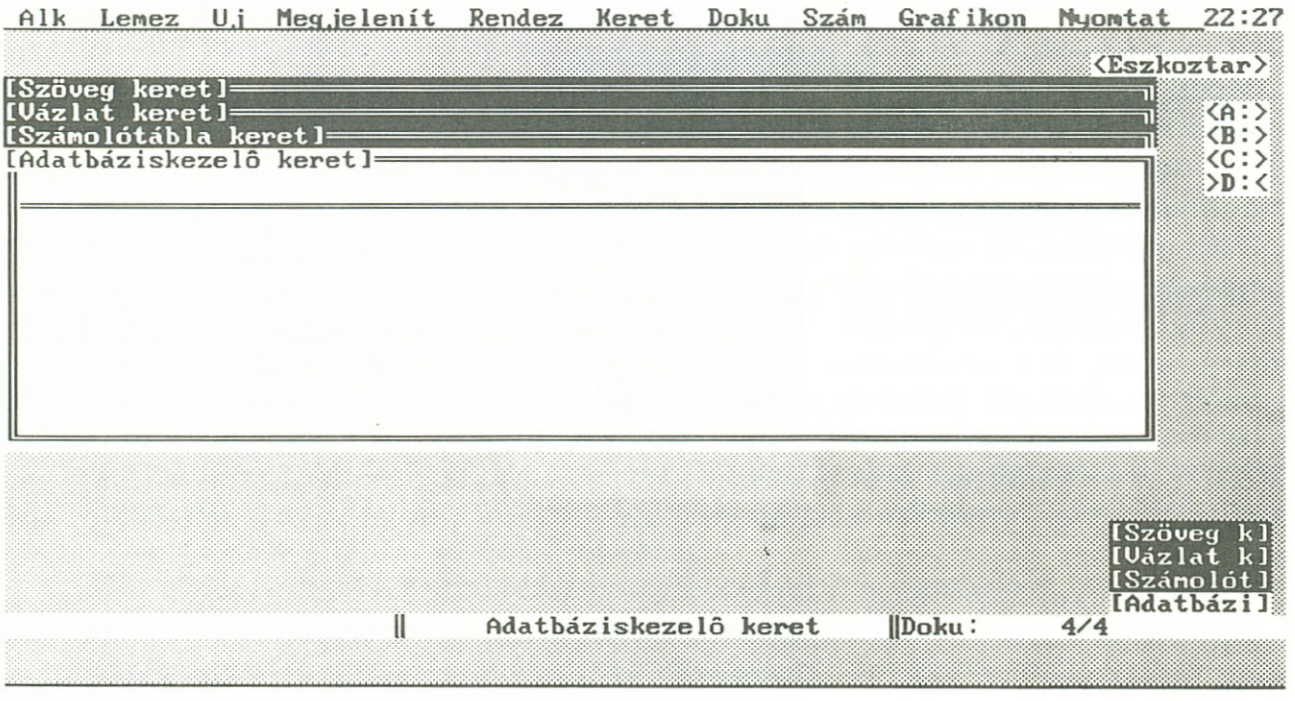
Némi gyakorlás után elmondhatod magadról, hogy az íróasztalon szinte minden helyet el tudsz érni. A legalsó sorban csak a gép beszél hozzád, oda csak ő írhat. Egyetlen hely van még, ahová praktikus, ha el tudsz jutni.

Először a már ismert módon menj fel a menübe, és állj rá az *új* menüpontra. Az ekkor megjelenő csoport tagjai közül válaszd ki az *üres/szövegkeret* menüt. vagyis állj

rá, és nyomd meg az ENTER gombot. Ezután megjelenik egy négyszög, mely egy ablakhoz hasonlít, ezt keretnek fogjuk nevezni. Ezzel már létre is hoztál egy keretet, és ott állsz a keret bal felső sarkában a kurzorral, vagyis itt:



Ez a Te saját szövegkereted. Tehát ha egyet már meg tudsz csinálni, akkor többet is, és ha valamiből több van, vagy később használni akarod, akkor valamivel meg kell különböztetni. Erre a legalkalmasabb az, ha nevet adsz a keretednek. Ehhez semmi mást nem kell tenned, mint beírni a nevét. Például a saját keresztnevedet. Viszont fontos, hogy figyelj arra, hogy ne használj egy nevet többször. Ezután nézz szét újra a képernyőn. Ha ügyes vagy, rögtön felfedezel egy újabb téglalapot a képernyő legalsó sarka felett jobbra, amiben a saját neved található, vagy az a szó, amit ráírtál a keretedre. Több keret esetén ezt látod:



Ez a téglalap arra szolgál, hogy mint a valódi íróasztalon, ha elpakolsz egy pár papírt egy kupacba, akkor innen előveheted a neve alapján, majd visszarakhatod.

Nézzük meg, hogy ezt hogyan valósíthatod meg itt a Framework íróasztalán. A kurzor tehát a keret szélén áll, és mikor a név beírása után ENTER-t ütöttél a keret szélének színe megváltozott. Ezért most újra nyomd meg az ENTER gombot. A képernyőn lévő íróasztal majdnem olyan üres lesz, mint amikor behívtad a programot. Egy különbség van, hogy az imént felfedezett téglalap nem tűnik el. Ez jelzi azt, hogy a keret a rendelkezésedre áll, csak nem látod.

Már félre tudjuk tehát rakni a papírjainkat. Ezt a helyet, ahol a neveddel jelölt keretcím egy téglalapon látható, a valóságos íróasztalnak megfelelően irattárcának nevezzük. Ide az ideiglenesen félrerakott, összecsukott kereteid kerülnek, de itt található azoknak a kereteidnek a neve is, amikkel dolgozol, tehát ha keresünk valamilyen keretet az íróasztalon, akkor itt találjuk meg.

Ha hosszabb időre akarjuk tárolni, akkor a valósághoz hasonlóan úgynevezett iratszekrénybe kell raknunk, ami mentéssel történik. Mentésre akkor van szükségünk, ha kereteinket a programból való kilépés után is használni akarjuk. Ez a menüből történik. Az iratszekrény, mint egy szekrény általában, részekre van osztva, és egy ilyen részben akármennyi keretet tárolhatunk. A berakáskor és kihíváskor is meg kell adni az elérési útvonalat, vagyis hogy melyik szekrény melyik polcán és milyen néven raktuk el a keretünket.

Most vegyük elő azt a keretet, amit az előbb elraktunk. Ezt úgy valósíthatod meg, hogy újabb ENTER-t ütsz a behívni kívánt keret nevén. Ekkor újra előtűnik az alkotott kereted.

Szövegírás

Kezdjük el a valódi feladat megírását.

Ehhez bele kell lépnünk a keretbe, mert oda írhatunk nagyobb szövegeket. Ezt szürke plusz gomb lenyomásával valósíthatod meg (amely a billentyűzet jobb felső sarkában található meg). Ekkor megjelenik a kurzor a középső területen, és várja, hogy begépeld az irományodat. A keret szélére való kilépésre a szürke mínusz gombot kell lenyomnod (ez a szürke plusz fölött található). A szöveg begépeléséhez semmi mást nem kell már csinálnod, csak a betűket beütni, majd a sorok végét ENTER-rel lezárni.

Most írjunk be egy rövid szöveget, például egy Murphy törvényt, a címe:A nyomtatási hibák alattomosságának axiómája.

"A döntő hibát csak akkor fedezed fel a szövegben, amikor a szöveget már kinyomtattad, az állományt pedig törölted".

Van néhány, a szövegszerkesztésnél nélkülözhetetlen dolog, amit fontos megismerned. Ilyen például, hogy a szövegbe való közlekedésre használhatod a már ismert nyílgombokat. De például, ha a sor elejére akarsz jutni, egyszerűbb, ha benyomod a Home gombot. Ha az aktuális sor végére akarsz jutni, akkor az End billentyűt kell lenyomnod. Szükséged lehet egy elrontott karakter törlésére, amelyet, ha a kurzortól balra lévő karaktert akarod eltüntetni, akkor a Backspace billentyűt kell lenyomnod annyiszor, ahány karaktert akarsz törölni. Ha viszont azt a betűt akarod törölni, amelyen állunk, akkor a Del billentyűt kell megnyomnod. Ez a folyamat is ismételhető, ameddig akarod.

Nagy betűket a Shift-tel írhatod.

Egy rövid szövegben alkalmazott szerkesztőfunkciók nagyobb dokumentumoknál már elég fárasztóak, de szerencsére semmi szükség nincs például 5 sor karakterenkénti

törlésére. Mert kiterjesztjük a parancs hatályát. Ezt úgynevezett kijelöléssel tehetjük meg, ami azt jelenti, hogy a kurzort a például törlendő szöveg elejére kell állítani, majd az F6 billentyű lenyomása után a nyílgombok irányításával "kifesthetjük" azt a szöveget, amivel műveletet akarunk végezni. A kijelölést Enter-rel zárjuk. Ezután például nyomjuk meg a Del gombot, mintha egyetlen karaktert akarnánk törölni. Ezután rögtön eltűnik a kijelölt rész.

A kijelöléssel más műveleteket is végezhetünk pl.: mozgathatunk. Ehhez csak ki kel jelölni egy szövegrészt, majd meg kell nyomni az F7 billentyűt (ami a mozgatás parancsot adja ki), majd oda kell menni a nyílgombok segítségével, ahová a szöveget mozgatni akarjuk. Majd az ENTER gomb lenyomása után itt jelenik meg a kijelölt szöveg (a másik helyen pedig már az F7 billentyű lenyomására eltűnik). Próbáld ki például, az irományod szórendjének megváltoztatásánál.

A mozgatáson kívül másolhatsz is, ugyanúgy mint a mozgatásnál, csak nem az F7-et, hanem az F8 billentyűt kell használni. Az eredeti szöveg a helyén marad.

Nagyobb szövegnél az áttekinthetőség javítása miatt szükség lehet egy-egy szó kiemelésére pl. más betűtípussal. A Framework keretein belül ez is megvalósítható. A menete: jelöld ki a megváltoztatandó szöveget, majd menj fel a menübe, és válaszd ki a DOKU menüt. Majd válaszd ki a megfelelő karaktertípust (amelyek kombinálhatók is) a kiválasztás nyílgombokkal történik, így mehetsz sorba, és ha megtalálod a legmegfelelőbbet, akkor a már ismert ENTER lenyomásával érvényesítheted a betűtípus megváltoztatását. Az eredményt rögtön meglátod a képernyőn.

Nagy szöveg esetén nem elhanyagolható a külalak. A szöveg megjelenítésén a különböző rendezésekkel javíthatsz. Ilyen például a jobbrarendezés, melyhez a DOKU menüből, az *igazítás jobbra* menüpontot kell kiválasztanod. A program az Enter lenyomása után -az eddig akármilyen formában beírt szöveget- berendezi az utasításnak megfelelően (persze csak a kijelölt részt). Próbáld ki ezt: "A szövegszerkesztés megmozdíthatatlansági törvényei" című újabb Murphy megállapításon. Az eredmény:

- jobbrarendezéskor:

1. ha egy szót akarsz törölni, akkor garantáltan az egész sor eltűnik.

- balrarendezéskor:

2. Ha egy sort akarsz törölni, akkor a teljes bekezdés eltűnik.

- középrerendezéskor:

3. Ha egy bekezdést akarsz törölni, akkor az egész szöveg eltűnik.

- szélekrerendezéskor:

4. Ha az egész szöveget akarod törölni, akkor semmisem történik.

A szövegszerkesztés csak egyetlen a Framework-kel megoldható feladatok közül. Ezért ismerjünk még néhány lehetőséget.

Táblázatkészítés

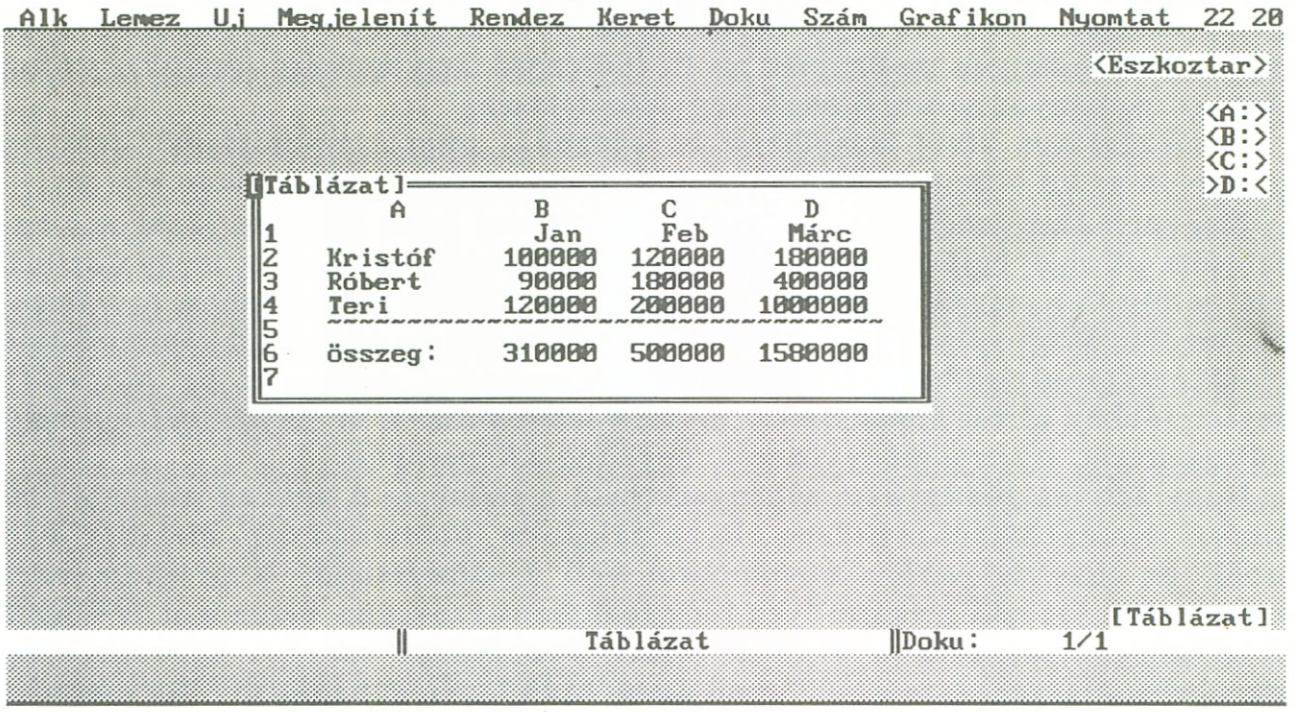
Újabb lehetőség például, hogy bonyolult számításokat a programmal végezhetünk el. Nagy cégek anyagnyilvántartására is alkalmas. Mivel a feladatot táblázatos formában oldja meg, ezért kapta a táblázatkezelő nevet.

A jobb elképzelhetőség kedvéért nyissunk meg egy táblázatot. Lépjünk be tehát az új menübe, és válasszuk ki a számolótábla menüpontját, és nyomjunk ENTER-t. Ekkor egy keret jelenik meg . (A már ismert módon adjunk nevet neki).

Táblázat lényege, hogy kis téglalapokból épül fel, ezeket celláknak nevezzük, és ezekbe írhatunk majd számokat, betűket, vagy képleteket.

Például ez a keret a legmegfelelőbb a zsebpénzek nyilvántartására is. A táblázat oszlopokból (A,B,C,...) és sorokból (1,2,3,...) áll. Ezek alapján hivatkozhatunk a cellákra. Például B-6 azt jelenti, hogy a B oszlop 6. soráról van szó. Most töltsünk fel adatokkal egy táblázatot.

Például így:



A B6 cellába képletet kell írnod, hogy a gép ki tudja számolni a barátaid havi zsebpénzét. A képletet nem fogjuk látni a képernyőn, csak a legalsó sorban. A cellában már a számítási eredmény látható.

Ha nagyobb és több képletet akarunk beírni, akkor zavaró lenne, hogy mindig csak egy sort látunk a képernyő alján. Ha megnyomjuk az F9 gombot, akkor a teljes képernyő nagyságában dolgozhatunk. Ez a billentyű kapcsolóként működik, ismételt megnyomásakor eredeti nagyságára (egy sorra) zsugorodik.

Összeget képezhetünk kétféle módon is. Az F2 gomb lenyomása után írjuk be a B-6 cellába: @SUM B2..B4, vagy beírhatjuk úgy is, hogy felsoroljuk az összeadandó tagokat. Írjuk be tehát a B6 cellába: +B2+B3+B4.

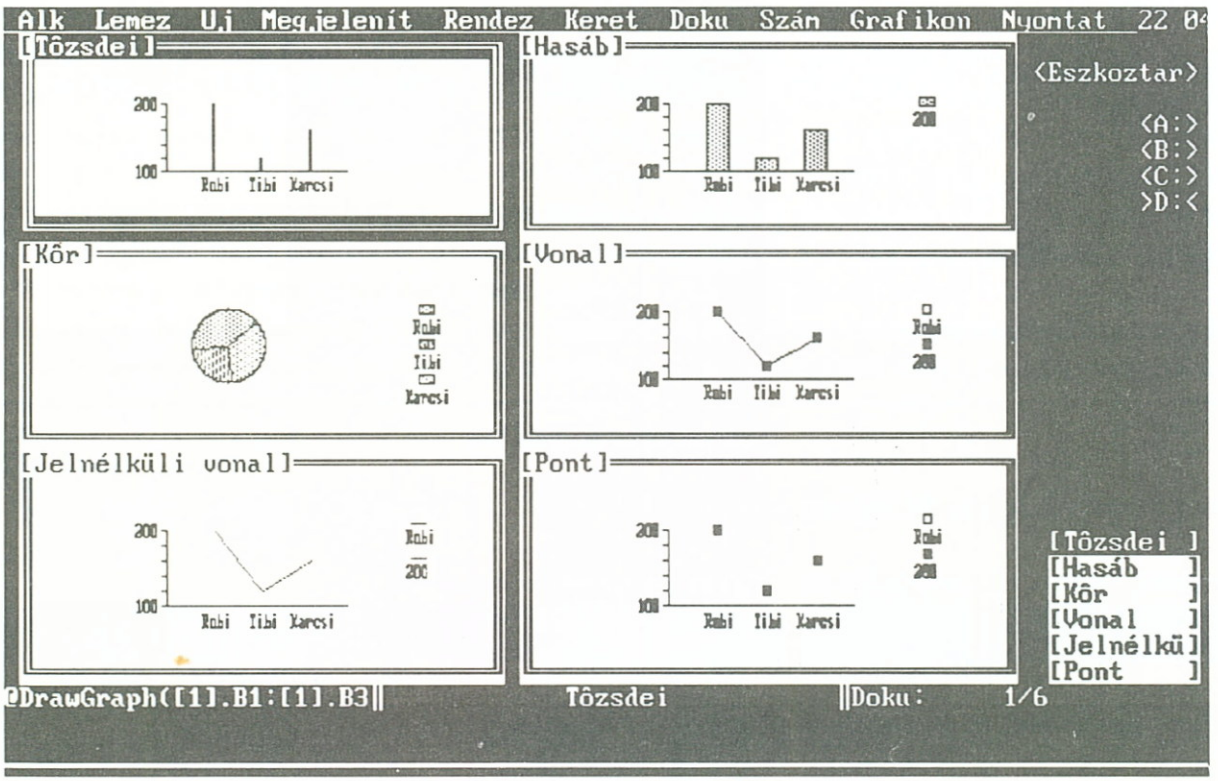
Felmerülhet az a gond is, hogy az egyik adat megváltozik, és akkor újra kellene számolni az egészet. A Framework megment ettől. Csak az adatot kell megváltoztatni, és a gép ennek megfelelően számolja ki a összeget (és a pénzedet is). Próbáld ki, hogy a C3 cella tartalmát 1300-ra javítod, és figyeld meg, hogy ennek megfelelően megváltozik a C6 cella tartalma.

Grafikonkészítés

Felvetődhet benned az az ötlet, hogy milyen jó lenne grafikus kimutatással szemléltetni a munkádat. Ez is megoldható a programmal, a táblázathoz grafikont készíthetünk. Állítsuk a kurzort a megfelelő táblázatkeretre, és jelöljük ki azokat a

cellákat, amelyekről grafikont akarunk rajzolni. Ezután menjünk a GRAFIKON menübe, és válasszunk a 8 grafikontípus közül. Majd írjuk elő a táblázat x tengelyén szerepeltetni kívánt sort. Végül válasszuk ki a menüből a grafikonrajzolási menüpontot. Állítsuk a kurzort a grafikon elhelyezésére szánt keretre, és nyomjuk le az ENTER billentyűt. Az F9 funkcióbillentyűvel a grafikon is teljes képernyőre nagyítható, majd ismételt megnyomással az eredeti nagyságára állítható vissza.

Így néznek ki a grafikon keretek:



A felsorolt lehetőségeken kívül még számtalan feladat megoldására alkalmazható ez a rendkívüli program, a FRAMEWORK!!!

Összefoglaló kérdések:

1, Milyen szóval indítható el a Framework?

2, Hol található a menü és a menüpont?

3, Hol van az állapotsor?

4, Hol látható a munkaterület?

5, Mi történik az Insert gomb lenyomása után?

6, Hol láthatod az almenüt, és hogy néz ki?

7, Hogyan közlekedhetünk a főmenüben és az almenüben?

8, Mi történik az ESC billentyű lenyomására, ha a menüben vagy?

9, Mit csinál a Scroll Lock billentyű?

10, Hogyan tudsz létrehozni egy szövegkeretet?

11, Hol látod a keretek nevét egymás alatt felsorolva?

12, El tudod-e rakni a kereteket úgy, hogy bármikor elővehesd, de mégse legyen előtted kiterítve?

13, Milyen billentyűvel hívhatod meg az irattárcán lévő egyik keretet, hogy láthasd a tartalmát?

14, Mi a mentés lényege?

15, Hogyan léphetünk be a keretek belsejébe?

16, Mire szolgál a szürke mínusz gomb?

Szövegírás kérdések:

17, Hova írhatunk nagyobb méretű szöveget?

18, Hogyan juthatsz az aktuális sor végére egy gomb lenyomásával?

19, Mit csinál a Backspace billentyű?

20, Milyen gombbal, illetve milyen gombokkal lehet még törölni?

21, Hogyan írhatsz nagybetűket?

22, Mi az F6 billentyű funkciója?

23, Hogyan válthatsz betűstílust?

24, Milyen szövegrendezési lehetőségeit ismered a Framework-nek?

Táblázat készítés kérdések:

25, Hogyan hívjuk a táblázatot felépítő téglalapokat?

26, Mit jelölnek a táblázatban az 1, 2, 3, 4, ... számok és az A, B, C, D ... betűk?

27, Hogyan lehet képletet beírni?

28, Hogyan alkalmazzuk és mire az F9 billentyűt?

29, Mi történik, ha az egyik adatot megváltoztatjuk a táblázatban?

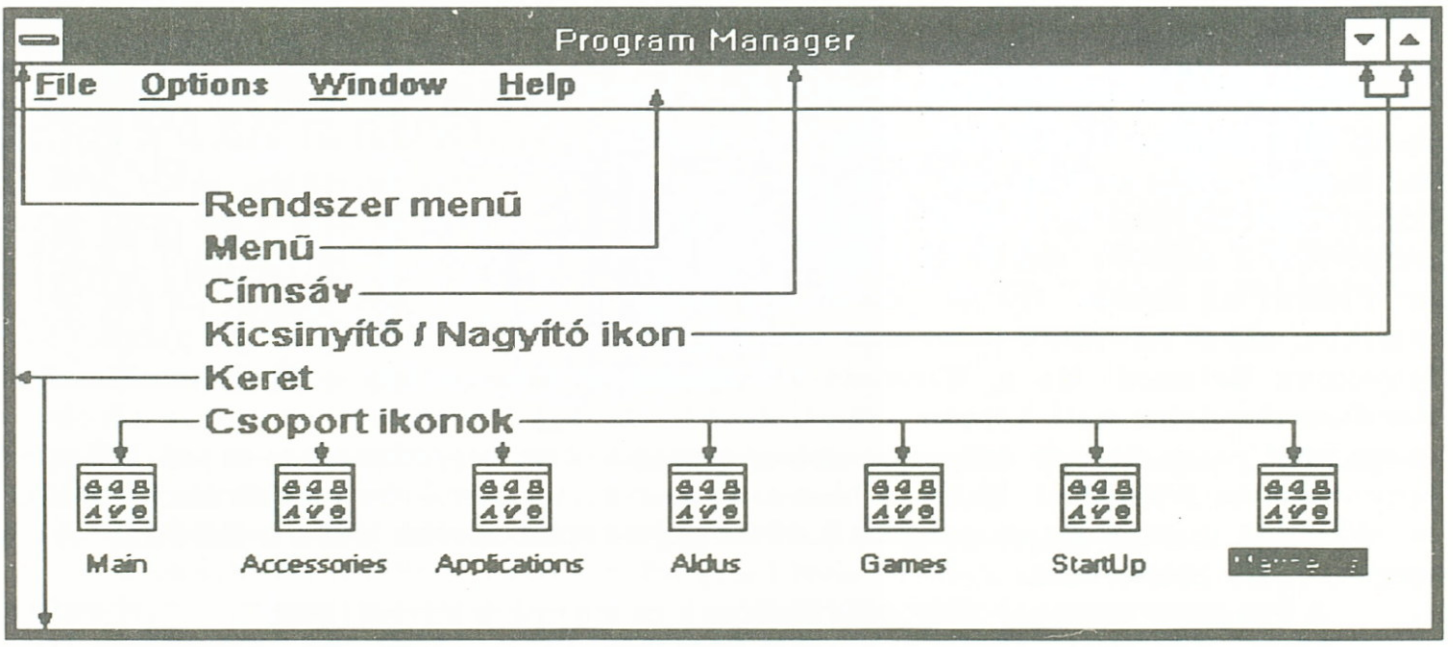
Windows 3.1

Bevezetés

A Windows 3.1, mint azt az angol neve is mutatja, egy ablakos rendszerű program, egyfajta operációs rendszer. A DOS-ból csak annyira van szüksége, hogy a DOS a prompt mögé beírt Win parancsot értelmezhesse. Ezután már nincs a DOS-ra szükség, a Windows 'uralja' a gépet. Ha elindítottad a Windows-t, már nem is igen használod kedvvel a DOS-t. Az Operációs rendszerek fejezetben már olvashattál valamit a Windows-ról, de az csak egy tömör ismertető volt, mivel mint operációs rendszerről annyi elég. De mint szoftver, a Windows-ról többet kell írni, mint egy ismertető, hiszen a DOS programok nagy többsége megtalálható Windows alatt is. Ha megnézel egy hirdetést amelyben programokat árulnak, nincs olyan amelyikben ne lenne legalább öttíz 'for Windows' felirattal ellátott cím.

Ebben a fejezetben a Windows kezeléséről, használatáról olvashatsz, majd a következő fejezetben egy rajzolóprogramról, a CorelDRAW!-ról, amely Windows alatt fut.

A Windows ablakai



A Windows kétféle ablaktípust használ: egyik az ún. felhasználói ablak, másik a dialógus ablak. Az első tulajdonképpen az az ablak, amely az egyes felhasználói programokat tartalmazza, a másik az az ablak, ahol a Windows bekéri az adatokat. A felhasználói ablakokat szabadon nagyíthatod, kicsinyítheted, mozgathatod. Ez utóbbit a dialógus ablakokkal is megteheted, de ezek méretét nem változtathatod. A rajzon egy felhasználói ablakot láthatsz, a Program Manager ablakát.

A Windows kezelése

A Windows-t kezelheted egérrel, de billentyűzetről is. Ajánlatos egérrel kezelned, mert billentyűzetről igen nehézkes. Ha egérrel kezeled, akkor a képernyőn láthatsz egy nyilat, amely az egér mozgását követi. Ezzel a nyíllal tudod a Windows-t kezelni, valamint az egéren található két vagy három gombbal (ha egy sincs rajta, akkor nézd meg, nem egy igazi egeret csatlakoztattak-e a géphez). A Windows-ban **mindig** a bal gombot kell használnod, a jobb és esetleg a középső gombot soha, vagy csak igen ritkán. A kétszeri kattintás alatt azt értem, hogy az egér gombját kétszer lenyomod, gyorsan, egymásután. Míg a gombot kétszer lenyomod, lehetőleg az egér ne mozduljon el, mert akkor lehet, hogy más funkciót indítasz el. Ha nem sikerül, akkor a kétszeri kattintás sebességét lejjebb veheted, a Windows Control Panel nevű programjában, vagy használd az ENTER-t.

A Windows programokból, ablakokból leggyorsabban a Rendszer menü négyzetére kétszer rákattintva léphetsz ki. A menüket egy gombkattintással gördítheted le, szintén eggyel zárhatod be. A kicsinyítő / nagyító ikonokat is egy-egy kattintással aktivizálhatod. A felhasználói ablakok méretét egyszerűen változtathatod meg egérrel: ha a kurzorral pontosan a keretre állsz, a kurzor átváltozik, és a keletkező nyilakkal változtathatod meg az ablak méretét.

A Program Manager

Ez a program tulajdonképpen a Windows főprogramja. Ha elindítod a Windows-t, akkor a Program Manager szintén futni kezd. Ha kilépsz a Program Manager-ből, a Windows szintén eltűnik. Ezt a programot a Windows programok betöltésére, új programok konfigurálására, programcsoportok létrehozására, a programok csoportok közötti mozgatására használhatod. A Program Manager-ben a csoportokat egy-egy ikon jelzi. Ha az ikonok bármelyikére kétszer rákattintasz, az ikon eltűnik és egy felhasználói ablak lesz helyette látható, amelyben ott vannak a programok ikonjai, ezeket is kétszeri kattintással indíthatod el. Ha csak egy programikont akarsz kijelölni, akkor arra az ikonra csak egyszer kell rákattintanod. Ha egy programikont át akarsz helyezni egyik csoportból a másikba, akkor a kurzort rá kell vinned az ikonra, nyomvatartani a gombot és elhúzni az egeret. Ha át akarsz egy programikont másolni egyik csoportból a másikba, akkor az előbbi műveletet kell megismételned, de közben a Ctrl gombot kell lenyomva tartanod. Ha a Windows-ból akarsz kilépni, akkor a Program Manager Rendszermenüjére kell kétszer rákattintanod. Ha egy programot le akarsz törölni a csoportból, vagy ha egy csoportot akarsz letörölni, azt leggyorsabban a Del billentyű lenyomásával érheted el. Ekkor a biztonság kedvéért a Windows megkérdezi, biztosan le akarod-e törölni azt a programikont, vagy csoportot. Ha Yes-t válaszolsz akkor megteszi, No esetén nem.

A Program Manager alapbeállításában a csoportok a következők:

Main - főcsoport; ebben van a File Manager, a Control Panel, a Print Manager, a PIF Editor, a Windows Setup program, a DOS Prompt és a Clipboard.

Accesories - tartozékok; ebben a csoportban a kiegészítő programokat találhatod meg: szövegszerkesztő-, rajzoló-, táv-adatátviteli-, kartoték-, jegyzetfüzet-, naptár-, számológép-, óra programot.

Games - játékok; talán ez a Windows egyik legtöbbet használt csoportja (legalábbis egy ideig). Itt a Solitaire (passziánsz) és a Minesweeper (aknakereső) programokat találhatod meg.

Windows Applications - Windows felhasználói programok; itt a Windows alá írt programokat találhatod meg. Némelyiket neked kell beállítani, erről később még lesz szó.

Non-Windows Applications - nem Windows alá írt DOS programok; ezek azok a programok, amiket nem Windows alá írtak ugyan, de a Windows képes őket futtatni .

StartUp - a Windows elindításakor ebben a csoportban lévő program indul el (a Program Manager mellett).

A Main programcsoport programjairól

File Manager - ezzel a programmal tudsz a fájlokkal és a könyvtárakkal manipulálni, sajnos még mindig egy kissé primitív szinten (azért még mindig, mert a Windows 3.0-ban is primitív volt a File Manager. Igaz sokat javítottak rajta, de még mindig nem az igazi).

Print Manager - ez a program akkor aktív, ha valamelyik Windows program elkezd nyomtatni. Először a Print Manager-be küldi a nyomtatandó fájlt, majd ott hagyja, és azzal a programmal ismét lehet dolgozni. A Program Manager miután kinyomtatta a fájlt, a nyomtatót ismét üres állapotba állítja.

Clipboard - ez a program egy ún. köztes tároló. Ha egy programból valamit kimásolsz vagy kivágsz, akkor az ide kerül. Ezzel a programmal tulajdonképpen csak megnézni tudod, mi van a Clipboard-on (-ban). Ennek a tartalmát tudod bemásolni valamibe, de akár el is mentheted .CLP kiterjesztéssel lemezre.

PIF Editor - ezzel egy .PIF kiterjesztésű fájlt tudsz készíteni, ami a DOS programok futtatásához kell.

Windows Setup - a Windows és a programjai, valamint a rendszerbeállításokhoz szükséges program.

DOS Prompt - ezzel a programmal a DOS-ba tudsz ideiglenesen kilépni. Ekkor használhatod a DOS programjaidat majd az Exit szó begépelésével visszatérhetsz a Windows-ba.

Control Panel - ebben a programban a Windows-zal kapcsolatos dolgokat állíthatod be. Megnézheted a betűtípusokat, beállíthatod a munkafelülettel, a színekkel, az egérrel kapcsolatos paramétereket. A Windows 3.1 már hangkártyákat is tud kezelni, ezeket is a Control Panel-ben állíthatod be.

Programok és programcsoportok létrehozása

Ha új programokat akarsz egy programcsoportba beilleszteni, akkor a következőket kell tenned:

- legördíted a File menüt, kiválasztod belőle a New menüpontot;
- a megjelenő dialógus ablakban Program Item gombra rákattintasz, majd az OK gombra is;
- az ezután megjelenő dialógus ablakban lenyomod a Browse... gombot;
- ezután kiválasztod a kívánt programot (úgy mint az NC-ben);
- rákattintasz az OK gombra, majd az előző ablakban is.

Ha új programcsoportot akarsz létrehozni, ugyanúgy kell eljárnod, mint az előbb, de a legelőször megjelenő dialógus ablakban a Program Group gombra, majd az OK gombra kell rákattintanod. Ezután a megjelenő dialógus ablakba a villogó kurzorhoz be kell írnod a csoport nevét, majd az OK gombot kell lenyomnod.

Ha valamit elrontottál, vagy valamit meg akarsz változtatni, akkor a File menü, Properties menüpontjának kiválasztásával az előbbi ablakokat tudod előhozni, ahol akármit megváltoztathatsz azzal a csoporttal, vagy programmal kapcsolatban.

Ha egy programnak ki akarod jelölni, hogy melyik könyvtárat használja alapértelmezésben, a következőket kell tenned:

- kijelölöd a programikont, megnyomod az Alt + Enter gombokat (Properties);
- beírod a Working Directory szavak mögé a kívánt könyvtár elérési útját;
- lenyomod az OK gombot.

Ha egy billentyűkombinációt akarsz a programhoz hozzárendelni, akkor

- válaszd ki a kívánt programot;
- nyomd le az Alt + Enter gombokat (Properties);
- a Shortcut Key szavak mögé írj be egy karaktert (a Ctrl + Alt gombok automatikusan hozzárendelődnek);
- nyomd le az OK gombot.

Ezekután a programhoz a Ctrl + Alt + a beírt gombokat rendelted hozzá. Ha a Ctrl + Alt gombok nem szimpatikusak, akkor hozzárendelheted akár a Ctrl + Shift, Alt + Shift, Ctrl + Alt + Shift gombokat is, ha nem csak egy karaktert írsz be, hanem ezeket is lenyomod.

A Program Manager menüi

File -

New... - új programokat, programcsoportokat tudsz létrehozni.

Open Enter - egy programot tudsz elindítani.

Move... F7 - egy programikont tudsz átmozgatni, egyik csoportból, a másikba.

Copy... F8 - egy programikont tudsz átmásolni, egyik csoportból a másikba.

Delete Del - egy programikont, vagy egy üres programcsoportot tudsz letörölni.

Properties... Alt + Enter - egy programcsoport, vagy egy programikon tulajdonságait tudod megváltoztatni.

Run... - egy programot tudsz elindítani.

Exit Windows... - kiléphetsz a Windows-ból.

Options -

Auto Arrange - automatikus programikon elrendezés.

Minimize on Use - ha elindítsz egy programot, a Program Manager mindig lekicsinyítődik.

Save Settings on Exit - kilépéskor a beállítások elmentődnek.

Window -

Cascade Shift + F5 - programcsoport ablakok egymás fölötti elrendezése.

Tile Shift + F4 - programcsoport ablakok mozaikszerű elrendezése.

Arrange Icons - az ikonokat rendezi el.

Help -

Contents - a segítség tartalmát írja ki.

Search for Help on... - keresés a segítségek között.

How to Use Help - útmutató a segítség használatához.

Windows Tutorial - egy Windows oktató program.

About Program Manager - információk a Program Manager-ről.

A Task Manager

Ezzel a programmal elérheted az összes éppen futó programot. A Task Manager-t a Ctrl + Esc gombokkal hívhatod elő, vagy ha a kurzort olyan helyre viszed, ahol nincs egy ablak sem (csak a háttér látszik) és ott kétszer kattintasz az egérrel. Szintén ugyanazt az eredményt éred el, ha valamelyik futó program Rendszer menüjéből a Switch to... menüpontot választod.

Egy másik programra úgy tudsz átkapcsolni, hogy kiválasztod a felsorolt programok közül a Neked kellőt, majd lenyomod az Enter gombot.

Ha az ablakokat egymás felett akarod látni, akkor a Cascade gombot kell választanod, míg ha az összes ablakból ugyanannyit akarsz látni, akkor a Tile gombot kell választanod. Ha sok ikon van a munkaasztalon szétszórva, akkor az Arrange Icons gombot válaszd. Ha egy programot le akarsz zárni, akkor válaszd ki a program nevét, majd kattints az End Task gombra.

Az OLE eljárás

Míg a Windows 3.0-ban a DDE (dinamikus adatcsere) volt használható, addig a Windows 3.1-ben az OLE (objektum-hozzákapcsolás és -beágyazás) a nyerő. Ez abból áll, hogy ha a Write-ba beágyazott Paintbrush rajzot meg akarod változtatni, akkor csak kétszer rákattintasz a rajzra, a Paintbrush betöltődik és vele együtt a rajz is. Ezt szokták még 'forró drótos' kapcsolatnak is hívni. Ez sajnos csak már elmentett állományoknál alkalmazható.

A következőkben leírtak a Write-ra vonatkoznak, de más programok is támogatják az OLE eljárást.

Rajz beágyazása

A grafikus programból másold át a rajzot a Clipboard-ra, majd a Write-ban a kurzort vidd a beillesztés helyére. A Write Edit menüjéből válaszd a Paste pontot, vagy nyomd meg a Shift + Insert gombot.

Rajz hozzákapcsolása egy dokumentumhoz

Az eljárás hasonló mint az előbb: a rajzot másold át a Clipboard-ra, a Write-ban vidd a kurzort a beillesztendő helyre, de most az Edit menüből a Paste Link pontot válaszd.

Beágyazott rajz szerkesztése

Ha a rajzodat át akarod szerkeszteni, csak rá kell kattintanod a rajzra, a hozzákapcsolt program betöltődik.

Tippek

Ha valamelyik About menüpontot választod, és az ablakában kétszer a program ikonjára kattintasz a Ctrl és a Shift gombok nyomvatartása mellett, majd az OK gombra kattintasz, akkor ha ismét előhívod az About ablakot és ismét rákattintasz kétszer a programikonra a Ctrl és a Shift billentyű nyomvatartása mellett, akkor előjön egy kis Windows zászló. Ha a fentieket megismétled, akkor a kis Windows zászló helyén egy stáblista jelenik meg, Bill Gates bácsival (a Microsoft főnökével) együtt.

Ha a felhasználói ablakot fel akarod nagyítani, vagy le akarod kicsinyíteni, akkor kattints rá kétszer a címsávra.

Ha a képernyő tartalmát a Clipboard-ra akarod átmásolni, akkor nyomd le a Print Screen billentyűt. Ha csak az aktuális ablakot akarod eltárolni, akkor az Alt + Print Screen billentyűket nyomd le.

Ha gyorsan akarsz programot váltani, akkor tartsd nyomva az Alt gombot, és nyomd le a Tab gombot. Ha felengeded az Alt-ot, akkor dolgozhatsz tovább a megnyitott programmal, míg ha mégegyszer lenyomod a Tab-ot akkor a következő program jön, és így tovább.

Ha egy programot úgy akarsz megnyitni, hogy csak ikonként szerepeljen a munkaasztalon, akkor a Shift billentyű nyomvatartása közben kattints rá kétszer a program ikonjára.

Ha már unod a Windows 3.1 bejelentkező képernyőjét, akkor a Windows-t így indítsd el: WIN és utána egy szóköz, majd egy kettőspont (WIN :).

Az Windows-zal kapcsolatos kérdések, feladatok

1., Mi a Windows?

2., Hogyan indítod el a Windows-t?

3., Mire lehet használni a Program Manager-t?

4., Hogyan indítasz el egy programot a Program Manager-ben?

5., Mire lehet a Task Manager-t használni?

6., Mi az OLE eljárás, és hogyan használnád a Write szövegszerkesztőben?

7., Készíts a Program Manager-ben egy csoportot a saját neveden!

8., Tedd bele a legtöbbet használt programodat, de ha már valahol szerepel, akkor ne mozgasd el onnan, hanem másold!

9., Indítsd el az egyik programot, majd kicsinyítsd le ikonná!

10., Indítsd el újból a programot, de a Task Manager-t használd!

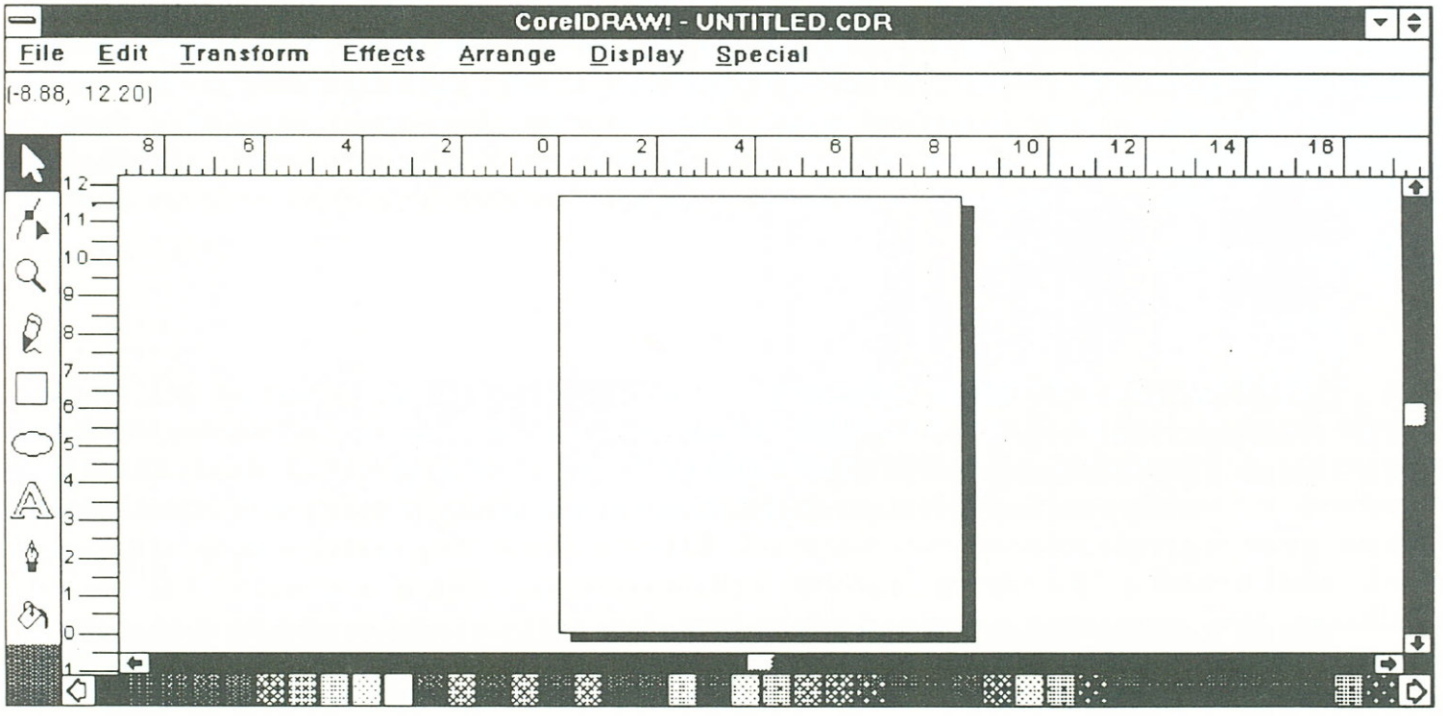
11., Töröld le a saját programikonjaidat, és a programcsoportodat is!

CorelDRAW! 2.00-ás verzió

A CorelDRAW!-ról

A CD nem bittérképes rajzolóprogram, mint a Paintbrush, hanem vektororientált. A CD-ban találhasz kb. 150 betűtípust, amelyeket egészen 1440 pont nagyságig nagyíthatsz. Ha a programhoz installálták, akkor megtalálod azt a sok ClipArt rajzot, amit szintén ehhez a programhoz adtak. Ehhez a programhoz adtak még egy rajznyilvántartó (MOSAIC), egy rajzkonvertert (CorelTRACE), egy fontkonvertert (Wfn Font Converter) és sok kész rajzot is. Egy fontos dolog: **mindig** a bal gombot kell használnod az egéren, ha nem akkor oda lesz írva, hogy mit lehet helyette használnod. A CD-ban a rajzodat mindig csak vázlatosan láthatod. Ez azért van, hogy a lassabb gépeken is az elviselhető szintre emelkedhessen a sebesség (különben csak a homokóráig lehetne eljutni).

A CD képernyője



A CD képernyőjén a következőket láthatod: legfelül a szokásos Windows fejlécet, alatta a legördülő menüket, azalatt a státuszsort, azalatt a vízszintes vonalzót, azalatt a rajzablakot. A rajzablaktól balra van a függőleges vonalzó, még balrább az ikonmenük. Az ablak alatt van a görgetőléc, azalatt a színpaletta. Az ablaktól jobbra a másik görgetőléc.

Az oldalsó ikonmenük

Az oldalsó ikonmenükben találhatod meg az alapvető rajzolási funkciókat, mint például a vonal, a kör, a négyzet, stb. Ezeket vagy ez egérrel, vagy a gombokkal tudod kiválasztani.

A nyíl

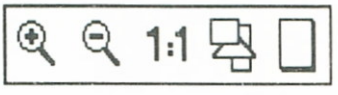
 Ez a legtöbbet használt szerkesztő funkció. Ezzel tudod a rajzelemeket (a továbbiakban tárgyakat) kiválasztani, ezzel tudod őket mozgatni, elforgatni és nagyítani. A kiválasztást az egérrel tudod megcsinálni, úgy, hogy a kiválasztott tárgyat vagy bekeretezed (mint ha egy négyzetet rajzolnál: odaviszed az egérkurzort az egyik sarkába, majd a gombot nyomvatartva leviszed a másik sarkába. A másik módszer, ha a kurzorral ráállsz a kívánt tárgyra és rákattintasz. Az egyik módszer akkor jó ha több tárgyat akarsz kiválasztani, a másik amikor csak egyet, vagy ha több tárgy van egymáson. Ha ki van választva egy tárgy akkor csak rá kell vinned a kurzort a tárgy körvonalára és rákattintani az egérrel. Fontos hogy ne a keretére vidd a kurzort hanem a tárgy körvonalára. Elforgatni úgy tudsz, hogy rákattintasz a kiválasztott tárgyra, ekkor megjelenik a közepén egy kör, a négy sarkában és az oldalai felénél egy-egy nyíl. A kör azt a pontot jelenti, amelyen forgatod a tárgyat, a nyilak a forgásirányt jelentik. A kört arrébbviheted, úgy, hogy a kurzort ráviszed és az egérgombot nyomvatartva elhúzod onnan. Ha az elforgatás után már nem kellenek a kis nyilak, akkor kattints rá egyszer a egérrel a tárgy körvonalára. Nagyítani úgy tudsz, hogy a kiválasztott tárgyon lévő fekete négyzeteket ragadod meg az egérkurzorral. Ilyenkor ha az egeret elhúzod, a státuszsorban láthatod a nagyítás mértékét.

A csomóponteditor (F10)

 A csomóponteditor (azért nem 'csomópontszerkesztő', mert az túl hosszú) szintén hasznos eszköz. Ezzel a vonalak, négyzetek, egyéb tárgyak csomópontjait tudod mozgatni, a vonalakat görbékké definiálhatod át. A mozgatáshoz csak meg kell ragadnod a csomópontot, és már mozgathatod is. Az átdefiniáláshoz a vonalra kell kétszer gyors egymásutánban rákattintanod. Ekkor a vonal megvastagodik és előjön egy ablak, ahol neked a 'To Curve' gombot kell lenyomnod. Ekkor a vonalon két kis kart találhatsz, amit mozgatva a vonalad görbe lesz. Ugyanígy vissza is alakíthatod a görbét vonallá: megint kattints kétszer rá a vonalra, ismét előjön az ablak, de most a 'To Line' gombot kell lenyomnod.

A nagyító (F2, F3, F4, Shift+F4)

 A nagyítóval megnézheted közelebbről a tárgyakat. Erre is többféle funkciót találhatsz: az egérrel ha rákattintasz a nagyítóra az ikonmenüben, akkor előugrik egy másik ikonmenü, amiben kiválaszthatod a nagyítás fajtáját:



Az első a normál nagyítás, ezzel ugyanúgy kell bánnod mint a kiválasztó nyíllal (F2). A második a kicsinyítés: a nagyítást megelőző nagyságot hozza vissza (F3). A harmadik akkorára nagyítja a képet, hogy minden tárgy látszódjon (F4). A negyedik a képet akkorára nagyítja, hogy a vonalzón a beosztások egy az egyben látszódjanak. Az ötödik a teljes papírt nagyítja ki (Shift+F4).

A ceruza (F5)

 A ceruzával rajzolhatsz szabadkézzel is és rajzolhatsz egyenesekkel is. Szabadkézzel úgy tudsz rajzolni, hogy az egérgombot lenyomva elhúzod az egeret. Egyenesekkel úgy tudsz rajzolni, hogy a kurzort odaviszed a kezdőpontra, ott megnyomod az egér gombját, majd a gombot felengedve elhúzod az egeret. Ha a kívánt pozícióba húztad az egeret, ott megnyomod mégegyszer a gombot, ekkor megrajzolja az egyenest. Ha Bèzier-görbékkel akarsz rajzolni, akkor a legördülő menüben kell átállítanod arra (Special menüpont, Preferences pontja; lásd később).

A négyzet (F6)

 A négyzettel - nem fogod kitalálni - négyzetet és téglalapot tudsz rajzolni. Ezt úgy teheted meg, mint a kiválasztásnál és a nagyításnál. Ha a rajzolás közben nyomvatartod a Ctrl gombot, akkor csak négyzetet rajzolhatsz, ha nyomvatartod a Shift gombot, akkor a kezdőpont nem a sarok lesz, hanem a négyzet vagy téglalap középpontja. Persze a Ctrl és a Shift gombot egyszerre is nyomvatarthatod, ekkor mindkét funkció érvényes.

A kör (F7)

 A körrel is ugyanúgy rajzolhatsz, mint a négyzettel, itt is használhatod a Ctrl és a Shift gombokat. Itt persze ha használod a Ctrl gombot, akkor nem négyzetet rajzol, hanem kört. A Shift gombra a kör középpontja köré tudsz kört rajzolni. Itt is nyomvatarthatod mindkét gombot és itt is mindkét funkció érvényes lesz.

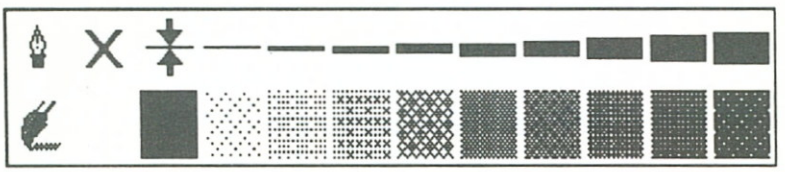
A betűk (F8)

 A betűkkel egy tetszőleges szöveget írhatsz ki a képernyőre. Kétféleképpen tudja a CD kezelni a szöveget: az egész szöveget vagy vázlatosan vagy grafikusan kezeli. A vázlatos funkciót úgy tudod használni, hogy a kurzort az egyik sarokba viszed, ott lenyomod és nyomvatartod az egér gombját, majd elhúzod az egeret. Ekkor egy téglalapot kereteztél be, amibe írhatod a szöveget. Ha a másik sarka is megvan, akkor elengedheted az egér gombját. Ekkor előjön egy ablak, ahol legfelül a szövegablakot láthatod, bal alsó sarokban a fonttípusokat, mellette a betűk nagyságát, azalatt láthatod magát a betűt. Ha kiválasztottad a betűtípust és a szöveget is beírtad, kattints rá az OK gombra. Ekkor ez az ablak eltűnik és a kiválasztott keretedben megjelenik a szöveg.

Ha grafikus betűket akarsz, akkor jelölj ki egy pontot a papíron, ez lesz a szöveg bal alsó kezdőpontja. Ezután ugyanúgy mint az előbb, beírhatod a szöveget, de ezután a szövegedet a program tárgyakként kezeli. Ha az egérgombot úgy nyomod le, hogy a Shift billentyű is le van nyomva, akkor a ClipArt rajzok közül válogathatsz (ezek nem biztos, hogy rajta vannak a lemezen, mert sok helyet foglalnak el).

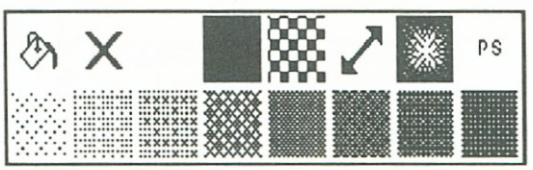
A toll (F12, Shift+F12)

 A tollal a tárgyak kihúzását tudod megcsinálni. Ez lehet vékony, lehet vastag, vagy akár nem látszó is. Tudsz színes tárgyakat is kihúzni, szinesen. Az almenü felső sorában a vastagságokat tudod beállítani, míg az alsó sorban a kihúzás színét.



A vödör (Shift+F11, F11)

 A vödörrel a tárgyakat tudod befesteni. Itt is, mint a tollnál, először a befestendő tárgyat kell kiválasztanod, utána rákattintani vagy a paletta egyik szinére, vagy erre a menüre. Ennek az almenüjében választhatod ki a szinezés tipusát, színét. A felső sorban a fajták közül a hetedik az érdekes. Itt színátmeneteket tudsz létrehozni. A megjelenő ablakban ki kell választanod, hogy milyen legyen a színátmenet típusa (Linear, Radial), majd azt a színt amiből kezdje az átmenetet és azt a színt amiben végződjön az átmenet. A lineáris szinezésnél megadhatod a szöget, amivel elforgatja a szinezést. A szinezést az alsó sorban lévő szinekkel is el tudod végezni, ott mintákkal szinezhetsz.



A szöveges menük

Sok beállítást csak a szöveges menükből tudsz elérni. Ezeket láthatod itt felsorolva. Az aláhúzott betűkkel és az Alt gombbal tudod őket használni. Néhány menüpontot a Ctrl és az ott írt betű lenyomásával is aktiválni tudsz. Ezek egy ilyen jellel vannak megkülönböztetve: '^'. Ha a menüpont neve mellett lévő billentyűkombinációt nyomod le, akkor az a menüpont aktivizálódik.

File

New - új fájlt tudsz létrehozni.
Open... ^O - egy meglévő fájlt tudsz behívni.
Save - a szerkesztett fájlt tudod elmenteni. Ha már volt egy neve, akkor azon a néven, ha csak Untitled volt felírva, akkor adhatsz neki egy nevet.
Save As... - a szerkesztett fájlt tudod egy másik néven elmenteni.

Import... - egy nem .CDR kiterjesztésű fájlt tudsz behívni.
Export... - egy fájlt tudsz nem .CDR kiterjesztéssel elmenteni.
Print... ^P - a szerkesztett fájlt tudod kinyomtatni.
Print Merge... - egy .TXT kiterjesztésű fájlt tudsz kinyomtatni.
Page Setup... - a papír nagyságát, színét tudod beállítani.
Control Panel... - a Windows Control Panel-jét tudod behívni.
Exit ^X - kiléphetsz a CD-ból.
About CorelDraw!... - a CD-ról tudhatsz meg információkat.

Edit

Undo AltBksp - a legutolsó lépést tudod visszacsinálni.
Redo AltRet - a visszacsinált lépést tudod nem visszacsináltá tenni (Juj!).
Repeat ^R - az előző parancsot tudod megismételni.
Cut ShiftDel - a kiválasztott tárgyat tudod a ClipBoard-ra kivágni.
Copy CtrlIns - a ClipBoard-ra tudod a kiválasztott tárgyat kimásolni.
Paste ShiftIns - a ClipBoard-on lévő tárgyat tudod bemásolni.
Clear Del - a kiválasztott tárgyat tudod kitörölni.
Duplicate ^D - a kiválasztott tárgyat tudod megkettőzni.
Copy Style From... - a kiválasztott tárgyra tudod rámásolni egy másik tárgy tulajdonságait.
Edit Text... ^T - a kiválasztott szöveget tudod megváltoztatni.
Character Attributes... - egy kiválasztott karakter tulajdonságait tudod megváltoztatni.
Select All - az összes tárgyat kiválaszthatod ezzel a menüponttal.

Transform

Move... ^L - a kiválasztott tárgyat tudod elmozgatni.
Rotate & Skew... ^N - a kiválasztott tárgyat tudod elforgatni.
Stretch & Mirror... ^Q - a kiválasztott tárgyat tudod felnagyítani, lekicsinyíteni, tükrözni.
Clear Transformation - a legutolsó transzformációt tudod semmissé tenni.

Effects

Edit Envelope - ezzel a csomópontszerkesztés fajtáját tudod meghatározni.
Clear Envelope - az utoljára átszerkesztett tárgyat tudod visszaállítani.
Copy Envelope from... - egy formát tudsz átmásolni a kiválasztott tárgyra egy másikról.
Add New Envelope - új formát tudsz adni a kiválasztott tárgynak.
Edit Perspective - a meglévő perspektívát tudsz változtatni.
Clear Perspective - a perspektívát tudod letörölni.
Copy Perspective from... - egy, már meglévő perspektívát tudsz átmásolni a kiválasztott tárgyra.
Add New Perspective - új perspektívát tudsz adni a kiválasztott tárgynak.
Blend ^B - két kiválasztott tárgy esetén használhatod. Az alul lévő tárgyból formálja meg a fölül lévő tárgyat, annyi lépésben, ahányban megadod.
Extrude ^E - a kiválasztott tárgyat három dimenzióssá tudod tenni.

Arrange

To *Front ShiftPgUp* - legfelülre hozza a kiválasztott tárgyat.
To Back ShiftPgDn - legalulra viszi a kiválasztott tárgyat.
Forward One PgUp - egy szinttel feljebb hozza a tárgyat.
Back One PgDn - egy szinttel lejjebb viszi a tárgyat.
Group ^G - több kiválasztott tárgyból, egy, együtt mozgatható, kezelhető csoportot csinál.
Ungroup ^U - felbontja a csoportot.
Combine ^C - a széttördelt tárgyat összerakja.
Break Apart ^K - széttördeli a tárgyat.
Convert To Curves ^V - az egyenesekből álló tárgyat görbékből állóvá változtatja.
Align... ^A - valamihez hozzáigazítja a tárgyat.
Fit Text to Path ^F - egy kiválasztott grafikus szöveg és kiválasztott egyenes vagy görbe esetén működik. A szöveget a vonalra vagy a görbére igazítja.
Align to Baseline ^Z - a tárgyat a bázisvonalra igazítja.
Straighten Text - kiegyenesíti a szöveget.

Display

Snap to Grid ^Y - a rácsozathoz állítja a lépésközt. A funkció lényege az, hogy a kurzor nem századonként megy össze-vissza, hanem akkora lépésekkel mint az a rácsozatnál van beállítva.
Grid Setup... - a rácsozatot tudod beállítani.
Snap to Guidelines - ha a kurzor a segédvonal közelébe ér, az automatikusan magához vonzza.
Guidelines Setup... - a segédvonalakat tudod elhelyezni.
Show Rulers - akarod-e látni a vonalzókat?
Show Statusline - akarod-e látni a státuszsort?
Show Color Palette - akarod-e látni a színpalettát?
Show Preview ShF9 - félképernyős látszatablak.
Show Full Screen Preview F9 - egész képernyős látszatablak.
Show Preview Toolbox - akarod-e látni a látszatablak ikonmenüjét?
Preview Selected *Only* - csak a kiválasztott tárgyat (tárgyakat) láthatod a látszatablakban.
Auto Update - automatikusan látható legyen-e a látszatablakban a változtatás?
Show Bitmaps - láthatóak legyenek-e a bittérképes rajzok?
Refresh Wire Screen ^W - felfrissíti a látszatablakot.

Special

Extract... - egy .TXT kiterjesztésű fájlt készít.
Merge Back... - egy .TXT kiterjesztésű fájlt olvas be.
Create Pattern... - egy mintát készít a rajzból.
Create Arrow... - egy nyilat készít a rajzból.
Preferences... ^J - itt az egérrel, a képernyővel, a vonalakkal kapcsolatos beállításokat tudod elérni.

A színpaletta

A színpalettán egyszerre nem láthatod az összes színt, mert nem férne ki a képernyőre. A paletta jobb és bal oldalán láthatsz egy-egy kis nyilat; ezekkel tudod jobbra és balra görgetni a palettát. Ha az egér bal gombját nyomod le a nyílon, akkor csak egyet megy arrébb, viszont ha a jobb gombot, akkor egy egész sornyit arrébb megy.

Tippek

Néhány, a rajzolás gyorsításához való tipp:

- ha a vonalzók bármelyikére, bárhol rákattintasz kétszer egymásután, akkor előjön a Grid Setup ablak;
- ha a vonalzók bármelyikén, bárhol nyomvatartod a gombot, és fel- vagy lehúzod a kurzort, akkor egy segédvonalat tudsz elhelyezni a papíron.
- ha a papír szélére kattintasz kétszer egymásután, akkor a Page Setup ablak jön elő.
- a Preferences menüben a Right Button beállításánál célszerű a Full Screen Preview-et beállítani, ugyanis akkor a jobb gomb lenyomására az egészképernyős látszatablak jön elő.

Kérdések és feladatok:

1., Mire való a mínusz jellel ellátott nagyító?

2., Hogyan tudod a ClipArt rajzokat előhozni?

3., Mi a Page Setup ablak előhozásának két fajtája?

4., Rajzolj egy négyzetet a papírra és töltsd ki színátmenettel!

5., Rajzolj egy egyenest és csinálj belőle görbét!

6., Az előző görbére illesz egy szöveget!

7., Mentsd el a rajzodat!

Számítógépek matematikája

Bevezetés

Ebben a fejezetben a számítógép matematikai hátterével fogsz megismerkedni. Lesznek nehezebben elsajátítható részek, amelyeket - ha még nem hallottál róla - többször kell elolvasnod, a példákat alaposan kell tanulmányoznod és önállóan is megoldanod.

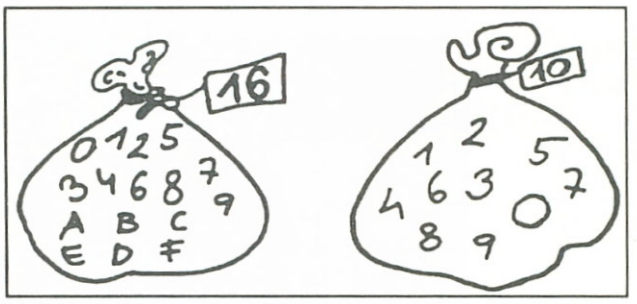
Megismerkedsz a számrendszerek közötti konvertálással, amely többek közt a gépi kódú programozáshoz is elengedhetetlenül szükséges, majd a számítógép belső számábrázolási módjáról olvashatsz.

Érdekes a fejezet végén ismertetett matematikai program, de helyszűke miatt alkalmazási lehetőségeit nem tudjuk megismertetni.

Számrendszerek, konvertálás

Számrendszerek

Mint bizonyára tudod, a számítógépek kettes (bináris) számrendszerben dolgoznak. A kettes számrendszer csak kétféle számjegyet ismer, a nullát és az egyet. Ha az 1998-at le akarjuk írni kettes számrendszerben, akkor a következő számsort kell leírni: 11111001110. Ez meglehetősen sok egyest és nullát tartalmaz, ezért számunkra kényelmesebb lesz a kettes számrendszerből könnyen képezhető nyolcas és tizenhatos számrendszer használata. A tizenhatos számrendszerben az általában használatos tizes számrendszer számjegyei (0..9) nem elegendők, hiszen itt a 10 és 15 közé eső számokat is egy karakterrel ábrázoljuk. Ezeknek a számoknak a jelölésére az ábécé betűit (A, B, C, D, E, F) használjuk.



Ezért az 1998 a tizenhatos (hexadecimális) számrendszerben így írható le: 7CE (Azt, hogy ez a szám a tizenhatos számrendszerben van ábrázolva, így jelöljük: $7CE_{16}$).

Konvertálás tizes számrendszerbe

A tizes számrendszerben a helyiérték fogalmát bizonyára ismered. A helyiérték alapján tudjuk eldönteni, hogy az egyes számjegyek milyen értéket jelölnek.

$135_{10} = 1*10^2 + 3*10^1 + 5*10^0$

A kettes számrendszerben a számjegyek a kettő megfelelő hatványával szorzódnak:

$101011 = 1*2^5 + 0*2^4 + 1*2^3 + 0*2^2 + 1*2^1 + 1*2^0$.

Ennek megfelelően:

$7CE_{16} = 7*16^2 + 12*16^1 + 14*16^0$.

A műveletek elvégzésével megkapod a szám tizes számrendszerbeli alakját:

$7CE_{16} = 7*256 + 12*16 + 14*1 = 1998_{10}$

$632_7 = 6*7^2 + 3*7^1 + 2*7^0 = 6*49 + 3*7 + 2*1 = 317_{10}$

Ilyen módon bármely számrendszerben lévő egész számot átválthatunk (konvertálhatunk) tizes számrendszerbeli számmá.

Konvertálás tizesből más számrendszerbe

Ha egy tizes számrendszerben ábrázolt egész számot akarunk átváltani tetszőleges számrendszerbe, akkor az eljárás a következő:

1. a tizes számrendszerben ábrázolt számot elosztjuk a számrendszert jelző számmal,
2. a hányados egész részét ismét elosztjuk az adott számmal,
3. a második pontot addig ismételjük, amíg a hányados egész része nulla nem lesz,
4. az osztási maradékokat fordított sorrendben leírjuk.

Pl.: A 7625-öt akarjuk átváltani nyolcas számrendszerbe.

1. 7625:8=953
   42
   25
   1

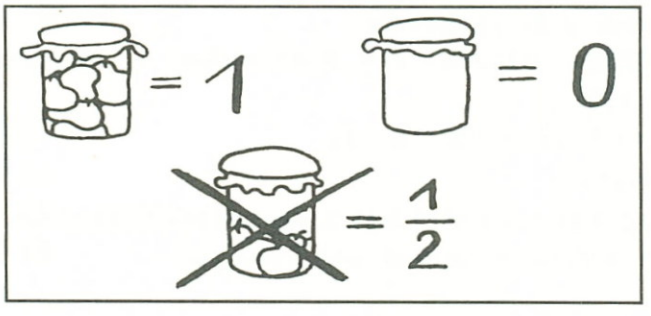
2. 3. 953:8=119 119:8=14
   15 39
   73 7
   1

   14:8=1 1:8=0
   6 1

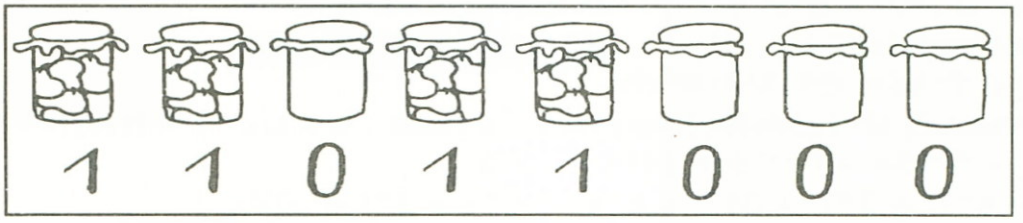
4. $7625_{10} = 16711_8$

A bit és byte fogalma, számábrázolás

A számítógép által használt egy darab egyest vagy nullát - tehát egy darab kettes számrendszerbeli számjegyet - bitnek nevezzük.



Nyolc ilyen számjegy (bit) alkot egy byte-ot. A számítógép egy byte-on tárol egy karaktert (betűk, számok, írásjelek). Például egy kódrendszerben (IBM-en is) a 01000001 byte az "A" betűt jelenti.



A számokat a számítógép általában nem számjegyenként ábrázolja -mert így sok helyet foglalna el és a műveletek elvégzése is nehézkes lenne- hanem tömörített formában. Az egész számokat kettes számrendszerbeli alakjukban ábrázolja a számítógép. Például 2 byte-on 0-tól 65535-ig vagy -32768-tól 32767-ig tud minden egész számot ábrázolni. Ebből is látszik, hogy az előjel tárolása egy bitet vesz igénybe.

A nem egész számokat a normálalakhoz ($482.123 = 4.8213*10^2$) hasonló formában ábrázolja, külön a mantisszát (pl. 5 byte-on) és a karakterisztikát (pl. 1 byte-on). A számok normálalakjáról már bizonyára tanultál. Ez egy olyan számábrázolási mód, amikor egy olyan tizedestörtet amelyiknek egészrésze egy és tíz közé esik beszorozzunk 10 valamelyik hatványával, és megkapjuk a kívánt számot. Pl.: a 23817.6973 normálalakja: $2.38176973 * 10^4$. Ennek ismeretében valószínűleg könnyebben megérted a számítógépek számábrázolását, ami - mint mondtuk - nagyon hasonlít a normálalakhoz. A 23817.6973-et a számítógép így ábrázolja:

+ .238176973 E +05.

Példánkban a mantissza a +.238176973, a karakterisztika a +05. Az E betű az angol exponent (kitevő) szó rövidítése.

Ezekből következik, hogy ebben az alakban minden számítógép csak véges tizedestörtet tud ábrázolni. Ez a számábrázolási mód (például 1/3-dal való számolás esetén) bizonyos számítási hibákat is okozhat, amelyek igen bosszantóak és elkerülésük esetenként komoly feladatot jelenthet. Ezenkívül hibát okozhatnak még olyan tizedestörtek is, amelyek végesek, de kettes számrendszerbeli alakjuk nem véges kettedes tört (ilyen például a 0.7). A tizedestörtek konvertálására a feladatoknál még visszatérünk.

Aritmetikai és logikai műveletek

A számítógép az egyes numerikus adatokkal különböző műveleteket tud elvégezni. A műveleteket két csoportba oszthatjuk:

- aritmetikai és
- logikai műveletek.

A főbb aritmetikai műveletek az összeadás, a kivonás, a szorzás és az osztás, vagyis a matematikából jól ismert négy alapművelet.

A logikai műveletek a következők:

- logikai ÉS (AND),
- logikai VAGY (OR),
- logikai KIZÁRÓ VAGY (XOR) és
- logikai NEM (NOT).

Az első három logikai művelet két kifejezést értékel ki, és ezek igaz vagy hamis voltától függően más-más értéket adnak eredményül.

A	B	A AND B
H	H	H
H	I	H
I	H	H
I	I	I

A	B	A OR B
H	H	H
H	I	I
I	H	I
I	I	I

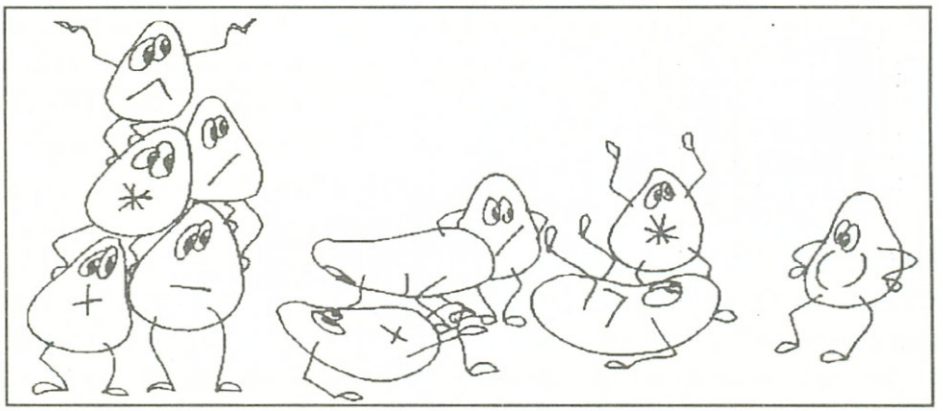
A	B	A XOR B
H	H	H
H	I	I
I	H	I
I	I	H

A logikai NEM egy kifejezés értékét változtatja az ellentettjére.

A	NOT A
H	I
I	H

A műveletek prioritása

Az egyes műveleteket prioritásuk sorrendjében végzi el a számítógép. A prioritás a műveletek magasabbrendűségét fejezi ki. A magasabb prioritású műveletek előbb, az alacsonyabb prioritásúak később kerülnek végrehajtásra. Az azonos prioritású műveletek végrehajtása balról jobbra történik. A végrehajtás sorrendje zárójelek alkalmazásával megváltoztatható.



Nézzünk erre egy példát. Gépeljük be a következő műveletsort:

3 + 6 * 4 - 2 ^ 3.

Ha a műveleteket balról jobbra haladva végezzük el, a következő eredményt kapjuk.

3 + 6 = 9
9 * 4 = 36
36 - 2 = 34
34 ^ 3 = **39304**

Ha figyelembe vesszük a műveletek prioritását, akkor az alábbi eredményt kapjuk:

2 ^ 3 = 8
6 * 4 = 24
3 + 24 = 27
27 - 8 = **19**

Ha ezt a műveletsort egy számítógéppel hajtatjuk végre, akkor végeredményként az utóbbit fogjuk megkapni. Ahhoz, hogy a végeredmény 39304 legyen, a következő módosítást kell elvégeznünk:

((3 + 6) * 4 - 2) ^ 3.

A számítógép a zárójeleket belülről kifelé haladva bontja fel, először mindig a zárójelben lévő műveletek kerülnek végrehajtásra.

A Graphic Calculus

A Graphic Calculus olyan grafikus matematikai program, amellyel az iskolában tanult függvényeket gyorsan, játszva megtanulhatod.

A program elindítása után megjelenik a képernyőn a program főmenüje, amelyből a megfelelő szám vagy betű leütésével választhatod ki a szükséges funkciót.

Az első menüpont játéklehetőséget kínál. Almenüjében az első pont kiválasztása után a program ábrázol egy függvényt grafikonon. A te dolgod, hogy ezt megfejtsd. Az általad beírt függvényt a program azonnal ábrázolja. Ha nem sikerült a megoldást eltalálnod, az ábrából következtethetsz a helyes megoldásra. A második almenüpont funkciója ugyanez, csak nehezebb függvényeket ad fel.

A harmadik almenüpont páros játékra való. Te írhatod be a függvényt, amit társadnak kell megfejtenie.

A főmenü harmadik menüpontjával grafikont lehet rajzolni. A hármas gomb megnyomása után kell a függvényt és a határértékeket megadnunk. Az ábra az F1 gomb megnyomása után jelenik meg a képernyőn.

Ezután kikérdezhetjük a pontos értékeket úgy, hogy a grafikonon lépegetünk a kereszttel. A kereszt lépésközét a <> gombokkal tudjuk változtatni. Háromféle nagyítási lehetőség is van. A LARGE a teljes képernyőre rajzolja a függvényt, a TIMES egy részletet nagyít ki a képernyő jobb oldalára. Ha a Z betűt nyomjuk meg, akkor ablakban jelölhetjük ki a nagyítani kívánt részt. Az ablak méretét az I és O gombokkal lehet változtatni.

Meg lehet keresni az ábrázolt függvénynek egy másikkal való metszéspontját. Erre való az EXTRA. A függvény javítására pedig a FUNCTION használható.

Ha a főmenüben a G gombot nyomod meg, akkor három dimenziós grafikonokat tudsz ábrázolni. Mivel matekból a kétváltozós függvényekről nem tanultatok, ezt a részt nem írjuk le részletesen, de próbáld ki, érdemes.

Konvertálási feladatok:

1. $10111111\ 10111101_2 = \ldots_{10}$

2. $772_8 = \ldots_{10}$

3. $FFFF_{16} = \ldots_{10}$

4. $832_8 = \ldots_{10}$

5. $0.11_2 = \ldots_{10}$

6. $372_{10} = \ldots_2$

7.** $0.6875_{10} = \ldots_2$

8.* $564_8 = \ldots_{16}$

9.* $342_{16} = \ldots_2$

10.** $0.123_{10} = \ldots_2$

Kérdések:

1., *Milyen számrendszerben dolgoznak a számítógépek?*

2., *Mi a bit és a byte?*

3., *Hogyan ábrázolja a számítógép az egész számokat?*

4., *Hogyan ábrázolja a számítógép a véges és a végtelen tizedestörteket?*

5., *A műveleteket milyen csoportokba osztjuk, és ezeken belül milyen műveleteket ismersz?*

6., *Mit jelent a műveletek prioritása?*

7., *Hogyan lehet megváltoztatni a műveletek végrehajtási sorrendjét?*

Feladatok:

Határozd meg a műveletek végrehajtási sorrendjét, és végezd el a műveleteket!

1., 3 + 2 * 6 = ...

2., 11 + 7 ^ 3 * 2 + 4 = ...

3., 7 + 9 ^ (8 - 5) + 4 * 3 = ...

4., 9 * (6 - 3 * (4 - 2)) + 1 = ...

5., *Állapítsd meg, hogy számológéped ismeri-e a műveletek magasabbrendűségét!*

Algoritmus

Az algoritmus fogalma

Az algoritmus tevékenységek, műveletek olyan sorozata, amely biztosan elvezet az adott feladat, probléma megoldásához. Tartalmazza a megoldás pontos leírását, a megoldáshoz vezető műveletek milyenségét, sorrendjét.

Nézzünk egy egyszerű, hétköznapi példát, mondjuk a teafőzés algoritmusát. Egy teljesen laikus ember számára valahogy így lehetne összefoglalni a teendőket:

> Végy egy kanna vizet és tedd föl melegedni. Várj amíg felforr, azután vedd le a gázról. Ezután mártogasd bele a teafiltert mindaddig, amíg kellően elszínezi a vizet. Ha ízesítve szereted, tégy bele citromot és cukrot ízlés szerint.

Ilyen és ehhez hasonló algoritmusok százait találhatod bármelyik szakácskönyvben leírva, vagy saját életed mindennapjaiban leíratlanul. Jól begyakorolt algoritmust hajtasz végre, amikor reggel mosakodsz, reggelizel, felöltözöl, iskolába utazol, telefonálsz és még sorolhatnám. Nem egy általad sosem hallott, érthetetlen dolog ez, hanem pusztán arról van szó, hogy egy nagyon is jól ismert fogalmat kell a hétköznapinál egy kicsit mélyebb szinten megismerned. Merüljünk tehát tovább az algoritmusok világában, mégpedig a teafőzés példáján keresztül.

Kérlek, olvasd el újra a fenti példát!

Találhatsz benne olyan egyszerű utasításokat, mint:

- végy egy kanna vizet;
- tedd föl melegedni;
- vedd le;
- mártogasd;
- tégy bele.

Ezek algoritmusaink legegyszerűbb, legelemibb összetevői, az **utasítás**ok.

Tovább vizsgálódva újabb érdekességet vehetünk észre:

- ...amíg felforr;
- ...amíg kellően elszínezi.

Vannak tehát olyan teendőink, utasításaink is, amelyeket nem elég egyszer végrehajtani, hanem valamitől függ, hogy meddig kell ismételni őket. (Példánkban a várakozás és a mártogatás az ismétlődő tevékenységek.) Az algoritmus ezen részeit **ciklus**nak nevezzük.

A következő újdonságot a példa utolsó mondatában találjuk:

- ha ízesítve szereted ...

Itt ugye nincs szó ismétlésről, mégis valami rafinációval állunk szemben. A "tégy bele ..." utasítást, vagyis a tea ízesítését nem biztos, hogy végre kell hajtani, ez attól függ, hogyan szeretjük a teát. Algoritmusainkban az ilyen és ehhez hasonló eseteket **elágazás**oknak nevezzük.

Mind a ciklusoknál, mind az elágazásoknál azt látjuk, hogy a bennük lévő utasítások végrehajtása valamitől függ. Ezt a valamit a számítástechnikában *feltétel*nek nevezzük (amíg felforr, amíg kellően elszínezi, ha ízesítve szereted).

Most már éppen elég újdonságot hallottál, itt az ideje, hogy összefoglald az algoritmusokról szerzett legújabb ismereteidet. Túltekintve a mondatok száraz jelentésén, algoritmusaink szerkezetileg a következő fő részekből állnak:

utasítások
ciklusok
elágazások

Gyakorló feladat:

1., Gondold végig napirendedet és válassz ki egy kisebb részt belőle, melynek algoritmusát a tea főzéséhez hasonlóan meg tudod fogalmazni. Ne válassz túlságosan bonyolultat, annak semmi értelme, de - ha fejlődni akarsz - nevetségesen egyszerűt se. A reálisan kiválasztott témára fogalmazd meg az algoritmusodat, majd ezután - a példához hasonlóan - szedd össze, hogy hol találsz benne utasításokat, ciklusokat, elágazásokat. Ha kételyeid vannak megoldásod helyességét illetően, fordulj tanárodhoz segítségért.

Eddig viszonylag egyszerű esetekkel foglakoztunk, amelyeket nem volt nehéz röviden és értelmesen szavakba foglalni. Más a helyzet, ha arra kérlek, hogy magyarázd el az 1-es busz újonc vezetőjének, hogy mi a teendője. Járművét ismeri, de se az útvonalat, se a várost nem. Ez ugye már egy más fajsúlyú feladat. Az ilyen bonyolultabb esetekben - melyeket nem kell majd lámpással keresned tanulmányaid folyamán sem - a számítástechnikában jártas emberek a mondatszerű leírás helyett más, áttekinthetőbb, szemléletesebb módszert alkalmaznak az algoritmus leírására.

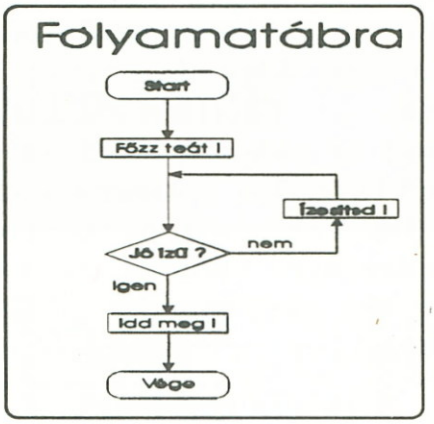
Algoritmus-leíró eszközök

Algoritmusaink leírására az alábbi lehetőségeink vannak:

- mondatszerű leírás,
- folyamatábra,
- struktogram.

A **mondatszerű leírás** az eddigi példák alapján már jól ismert.

A **folyamatábra** igen széles körben használt leíró eszköz, de olyan gondolkodásmódot kíván, amely a későbbiek folyamán nem segítené elő a Pascal nyelv könnyű megértését, ezért részletesen nem foglalkozunk vele.



A **struktogram** lesz az az algoritmus-leíró eszköz, amely a leginkább megfelel a céljainknak. A vele való munka gondolatmenete nagyon hasonlít a Pascal nyelvben kívánatos gondolkodásmódhoz, ezért a továbbiakban ezzel fogunk foglalkozni.

Az algoritmusról bővebben

Az egyszerű utasításokat gondolom sikerült az eddigiek alapján megértened, és sok problémát a jövőben sem fognak okozni. Ilyen utasítás lehet hétköznapi esetekben például, hogy:

- kelj fel,
- írd,
- olvasd stb.,

vagy a számítógép esetében:

- kiíratás a képernyőre,
- adatbeolvasás billentyűzetről,
- egy képlet kiszámolása.

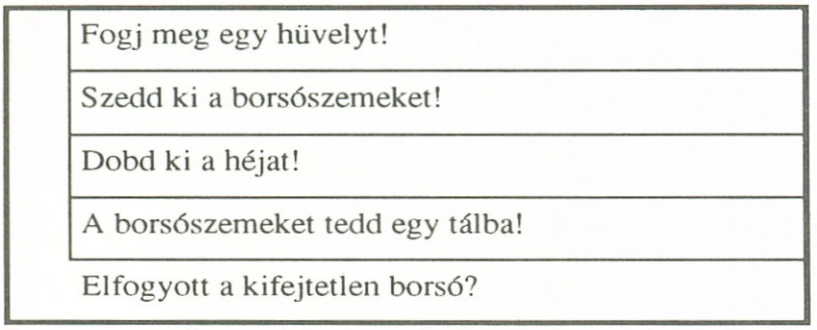
Struktogrammal történő leírás esetén az utasításokat egyszerűen beírjuk egy téglalapba:

Kelj fel!

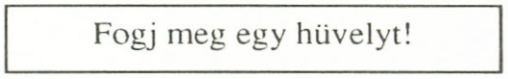
A ciklusokkal már egy kicsit több gondod lesz, de megijedni nem kell tőle. Gondold végig, hogy mit végeztetnél el azzal, aki egy tál hüvelyes borsót akar kifejteni. Én a következőket mondanám neki:

Fogj meg egy hüvelyt, szedd ki belőle a szemeket! A héjat dobd ki, a borsószemeket tedd vissza a tálba! Mindezt ismételd addig, amíg van kifejtetlen borsó.

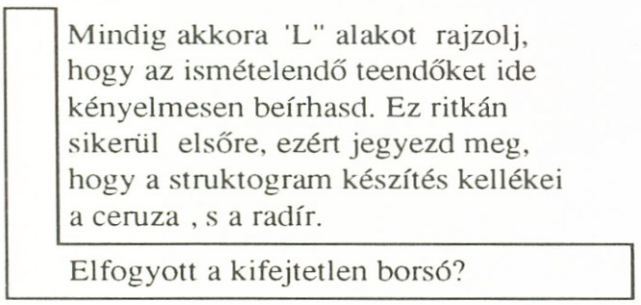
A következőképpen lehetne ezt struktogrammal leírni:



Az ugye itt is igaz, hogy az egyszerű utasításokat téglalapba írjuk:

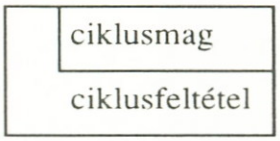


újdonság viszont a ciklus "L" alakú jelölése:
De nem ördöngősség.



Néhány elnevezés, amelyet érdemes megjegyezned:

Az ismétlődő tevékenységeket nevezzük ciklusmagnak, az "L" alakba írt feltételt pedig ciklusfeltételnek.



Az ilyen ciklusokat nevezzük ***hátultesztelő ciklus***nak.
Ennek az a jellemzője, hogy először végrehajtjuk a ciklusmagot, és csak utána vizsgáljuk, hogy szükség van-e további ismétlésre. Mert ugye semmi értelme sem lenne a munka megkezdése előtt figyelni, hogy végeztünk-e.

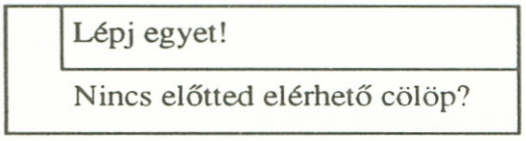
Egy jó tanács:

Hátultesztelő ciklusnál a ciklusfeltételt mindig úgy fogalmazd meg, hogy a kérdésfeltevés teljesülése esetén kelljen a munkát befejezni - vagyis a ciklusból kilépni , mert a Pascal nyelv úgyis ezt követeli majd meg. Szaknyelven ezt úgy mondjuk, hogy ***a ciklusból való kilépés feltételét adjuk meg.***

Nézzünk egy másik példát.

Piócáktól hemzsegő széles csatornán kell átkelned, de se híd, se csónak nincs a közelben. Viszont cölöpök állnak ki a vízből egymástól lépésnyi távolságra, ezeken kényelmesen átsétálhatsz.

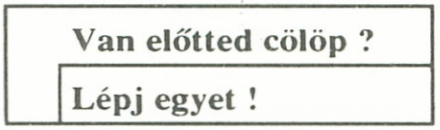
De milyen gondolatmenet kellene most?
Próbálkozzunk a "borsófejtő" stratégia alapján!



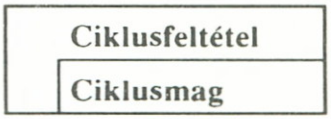
Ezzel az algoritmussal szerencsés esetben átjuthatsz a túlpartra, viszont nem szeretnék a helyedben lenni, ha a legelső cölöp véletlenül nincs a helyén. Ez esetben ugyanis csak a piócák járnának jól.

Mi lenne tehát a megoldás?

Az ilyen és ehhez hasonló esetekben javaslom, hogy előbb vizsgálódj, és csak azután cselekedj.
Vagyis struktogrammal leírva:



Az "L" alakú jelölés itt is megmaradt, csak fejjel lefelé fordítottuk.



Az ilyen ciklusokat nevezzük ***elöltesztelő ciklus***nak.
Ennek a ciklusnak az a sajátossága, hogy a ciklusmag végrehajtása előtt megvizsgáljuk a ciklusfeltételt, és ha az nem teljesül, akkor a ciklusmagot egyszer sem hajtjuk végre. Ezzel ellentétben a hátultesztelő ciklus magja egyszer biztosan végrehajtódik.

Még egy jó tanács:

Elöltesztelő ciklusnál a ciklusfeltételt mindig úgy fogalmazd meg, hogy nemleges válasz esetén kelljen a munkát befejezni - vagyis a ciklusból kilépni -, mert a Pascal nyelv úgyis ezt követeli majd meg. Szaknyelven ezt úgy mondjuk, hogy ***a ciklusba való belépés feltételét adjuk meg.***
Most már nagyon sok mindent tudsz az algoritmusokról. Itt az ideje, hogy néhány egyszerűbb példát önállóan is végig gondolj.

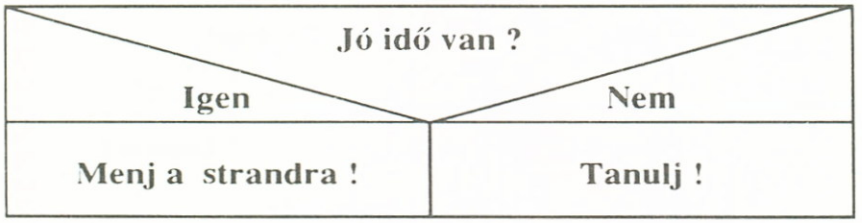
Gyakorló feladat:

1., Találj ki 1-1 mindennapi problémát a hátultesztelő és az elöltesztelő ciklusra! Gondold is végig, és írd le az algoritmust struktogrammal!

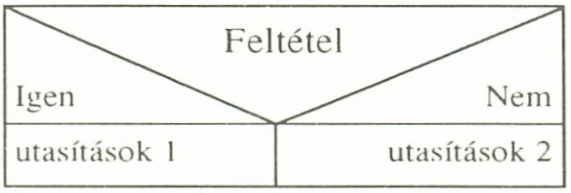
Ezek után nem maradt más hátra, minthogy az elágazásoknak egy kicsit a mélyére nézzünk.

Tegyük fel, hogy van ugyan tanulni valód, de nagyon szeretnél strandra is menni. A választást attól teszed függővé, hogy milyen az időjárás.

A döntés algoritmusa struktogrammal leírva a következő:



A feltétel (Jó idő van?) teljesülésétől vagy nem teljesülésétől függően vagy csak az egyik, vagy csak a másik ág utasításai hajtódnak végre.



Az ilyen elágazásokat kétirányú elágazásoknak nevezzük.

A tanulást befejezve, vagy a strandról hazaérve már csak az a dolgod maradt, hogy táskádat összekészítsd másnapra. Nem feltételezem, hogy mindig az összes tanszeredet magaddal hordod, ezért kénytelen vagy végiggondolni, hogy mi is kell másnapra, milyen nap lesz. Ez a kérdés már nem válaszolható meg igennel és nemmel, hanem pontosan hét válasz-eset lehetséges.

Hogyan írjuk ezt le struktogrammal?

Végső soron a már ismert kétirányú elágazások sokaságával is meg lehetne oldani a problémát, de van rá egyszerűbb lehetőség is. Ez pedig a következő:

Milyen nap lesz holnap ?						
Hétfő	**Kedd**	**Szerda**	**Csütörtök**	**Péntek**	**Szombat**	**Vasárnap**
Berak: hétfői órarend	**Berak: keddi órarend**	**Berak: szerdai órarend**	**Berak: csütörtöki órarend**	**Berak: pénteki órarend**	**Berak: szombati órarend**	**Berak: Vasárnapi órarend**

Attól függően, hogy a feltétel (Milyen nap lesz holnap?) melyik esete áll fönn, csak egy ág utasításai hajtódnak végre.

Feltétel			
Eset 1	**Eset 2**	**...**	**Eset n**
utasítás 1	**utasítás 2**	**...**	**utasítás n**

Az ilyen elágazásokat többirányú elágazásoknak nevezzük.
Foglaljuk tehát össze az algoritmusokról tanultakat!

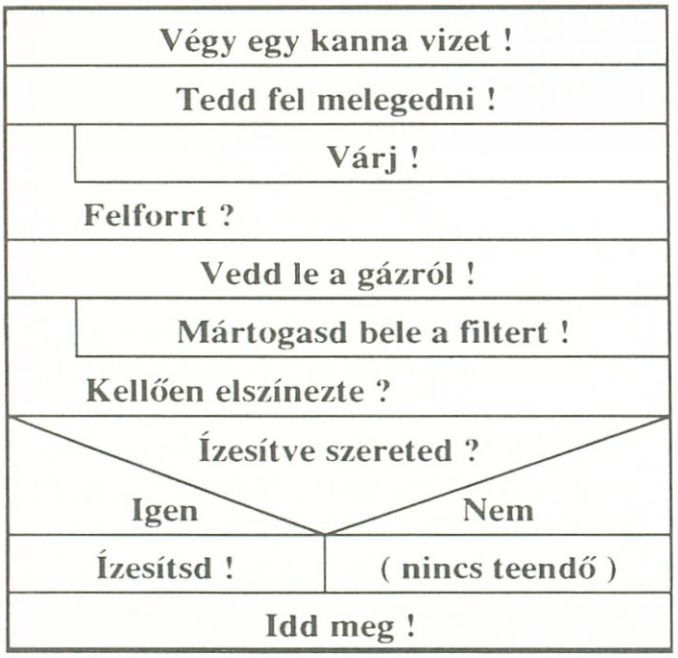
Algoritmusaink elemei:

- utasítások
- ciklusok
  - hátultesztelő
  - elöltesztelő
- elágazások
  - kétirányú
  - többirányú

Eddig az algoritmus egyes elemeit különálló részekként kezeltük, most nézzük meg, hogyan kapcsolódhatnak össze egy bonyolultabb probléma esetén. Térjünk vissza legelső példánkhoz, amely így szólt:

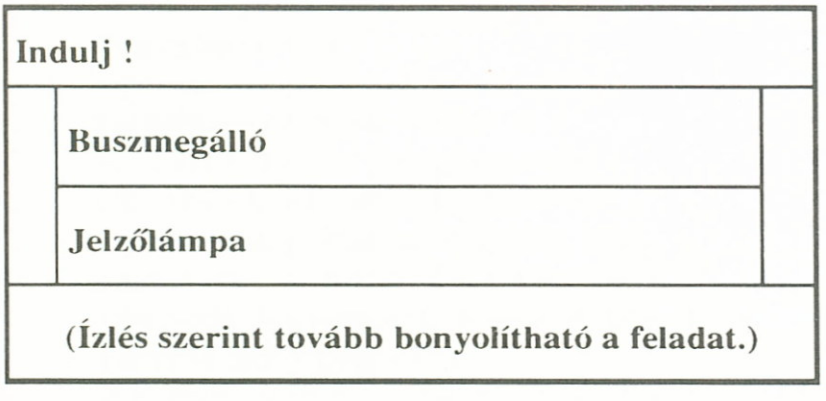
Végy egy kanna vizet és tedd föl melegedni. Várj, amíg felforr, azután vedd le a gázról. Ezután mártogasd bele a teafiltert mindaddig, amíg kellően elszínezi a vizet. Ha ízesítve szereted, tégy bele citromot és cukrot ízlés szerint.

A struktogram pedig:



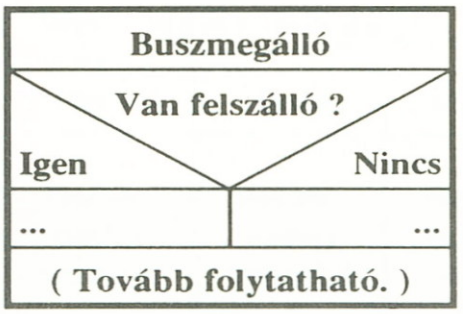
Ha valami nem világos az ábrán, javaslom, hogy lapozz vissza oda, ahol előzőleg részletesen foglalkoztunk a példával.
Vajon mi van akkor, ha olyannyira összetett a probléma, hogy a struktogram nem fér ki egy oldalra?
Bebizonyítom, hogy semmiféle problémát nem okoz ez sem. Ilyenkor részekre kell bontani a feladat egészét, és a részeket külön-külön kell kidolgozni. Nézzük az 1-es busz esetét.

Első közelítésben:



Ez a kettős oldalú téglalap azt jelenti, hogy valahol részletesen is ki van dolgozva.

Például:



Ezzel a módszerrel tetszőleges nehézségű feladatot áttekinthető méretű részfeladatokra tudunk bontani. Struktogram készítéshez tehát nincs is szükségünk a normál (A4-es) méretűnél nagyobb papírra.

Most már elméletben mindent tudsz, itt az ideje, hogy megbirkózz néhány valódi struktogram elkészítésével. A következő feladatok úgy vannak megfogalmazva, hogy a bonyolultságukat magad szabod meg. Igyekezz csak annyira elbonyolítani őket, hogy ne legyen megoldhatatlan, de a túl egyszerű sem sportszerű.

Gyakorló feladatok:

1., Készíts struktogramot reggeli teendőidre, a felébredéstől az iskolába való elindulásig!

2., Készíts struktogramot arra, hogy miként jutsz el otthonról az iskolába!

3., Készíts struktogramot kedvenc ételed elkészítésére!

A Turbo Pascal programnyelv

Bevezetés

Ha az előző fejezetet figyelmesen végigolvastad és elvégezted a gyakorló feladatokat is, akkor biztos lehetsz benne, hogy jó alapokat szereztél a Pascal programnyelv elsajátításához és a benne való programozáshoz. Egy kellemetlen újdonságra azonban már most szeretném felhívni a figyelmedet. Eddig kizárólag hétköznapi algoritmusokkal foglalkoztál, melyeket feltehetően értelmes emberek hajtanak végre. Vagyis megengedhettél magadnak kisebb megfogalmazásbeli pontatlanságokat, mert végrehajtás közben az emberi agy értelemszerűen áthidalja ezeket. Nem így tesz viszont a számítógép, amely önálló gondolkodásra képtelen. Mindent megtesz, amire a programon keresztül utasítást kap, de ezt és csak ezt hajtja végre. Mostantól kezdve tehát kínos precizitással vagy kénytelen utasításokat adni, mert ezek nem egy értelmes lénynek szólnak.

A nyelv történetének főbb állomásai

- 1968-ban készült el Nicklaus Wirth professzor terve, amelyet az addigi legfejlettebb programozási nyelvek - mint például az ALGOL - felhasználásával, azok továbbfejlesztésével készített.
- 1970-ben készült el az első fordítóprogram a nyelvre.
- 1973-ban született meg a Standard Pascal nyelv, amely azóta is a nyelv különböző változatainak az alapja.
- Több egyetemen is kifejlesztették a nyelv változatait. Az első változatok nagygépekre készültek. A nyelv rugalmassága, ereje akkor mutatkozott meg, amikor megjelentek a személyi számítógépek és kiderült, hogy ezekre is kiváló fordítóprogramokat lehet írni.
- A Turbo Pascal programozási nyelv is egyike az elsősorban IBM PC-re készült változatoknak, sőt, a nyelv és fordítóprogramja valószínűleg a legsikeresebb az összes eddigi Pascal-változat közül. Ez a nyelv a Standard Pascal továbbfejlesztett változata. A Borland Intézet fejlesztette ki az Egyesült Államokban. Az intézet 1982-től foglalkozik a nyelv állandó fejlesztésével, mindig újabb, s egyre jobb verziók kidolgozásával.
- Az első verzió -a Turbo Pascal 1.0- 1984-ben látott napvilágot, vagyis az IBM személyi számítógépek megjelenése után hamarosan forgalomba is került ez a nyelv.
- 1985 januárjában jelent meg a 2.0 verzió, mely főként az 1.0 hiányosságait igyekezett pótolni.
- Még ez évben, 1985 decemberére készült el a 3.0 verzió, amely jelentős bővüléseket hozott például a grafikában.

- A nagy áttörést a 4.0 verzió jelentette, amely hosszabb várakozás után, 1987-ben jelent meg. Ebben teljesen átalakították a nyelv kezelőrendszerét, bevezették a unit-okat, a Hercules, EGA, VGA stb. képernyők használatát, sok beépített eljárással kiegészítve ugrásszerűen megnövelték a Turbo Pascal nyelv színvonalát.
- Az ezután is fennálló problémákat már csak az 5.0 verzió oldotta meg, amely 1988 augusztusában látott először napvilágot, majd a kisebb hibáinak kijavítása utáni változat 1988 novemberében.
- Az 5.5 verzió 1989-ben készült el. A rendszer kezelése még egyszerűbb lett, és a sok kisebb fejlesztés mellett megjelent az objektum-orientált programozás, mely az 5.5 verzió legjelentősebb eredménye.
- a 6.0 verzió még az 5.5-höz viszonyítva is tartalmaz újdonságokat (ablaktechnika, egérkezelés stb.)
- legújabb a 7.0 verzió

A Turbo Pascal a többi programozási nyelv között igen előkelő helyet foglal el, nem véletlen tehát, hogy a tekintélyes szakemberekből álló zsüri a közelmúltban többször is az év nyelvének választotta. A magasszintű programozási nyelvek közt mindenképpen a legjobbak közé tartozik.

A Turbo Pascal egyaránt alkalmas matematikai számításokat, adatfeldolgozást, grafikus megjelenítést végző programok megírására, valamint sok más szerteágazó probléma megoldására. A nyelv rendelkezik többek közt saját editorral, amellyel a forrásprogram szövegét megszerkeszthetjük. Rendelkezik továbbá igen gyors fordító és futtató rendszerrel, jól használható hibakereséssel.

A programnyelv utasításai

Elemi apróságok

Magának a Pascal programnak a betöltésével és kezelésével itt nem foglalkozunk, ezt tanárod megmutatja a számítógépteremben. Inkább térjünk rá a programírásra.

A kezdés minden esetben ugyanaz:

```
program nev;
```

A "program" szó és a ";" kötelező, a név majdnem tetszőleges, akár el is hagyhatod. (Már most megemlítjük, hogy a Pascalban minden utasítást -de nem minden sort- pontosvesszővel kell elválasztani a következőtől.)
Ami ezután következik, az nagymértékben függ a megoldandó feladattól.
Ha például ki akarod iratni a képernyőre, hogy "Ez nem is nehéz.", akkor így kell folytatnod:

```
begin
  write ('Ez nem is nehéz.');
end.
```

A **begin** (kezdet) és az **end** (vége) szavak használata kötelező, és a program legvégére pontot (end.) kell tenni. Az összes többi végrehajtandó utasítást e két szó közé kell írni. Az, hogy a **write** utasítást beljebb írtuk egyéni dolog, így tesszük áttekinthetőbbé programjainkat.

A képernyőre való kiíratás

Ha rájöttél, hogy a **write** az egyik kiíratő utasításunk, akkor gratulálunk. Ha nem, akkor most közöljük.

Működéséről a példa sokat elárul:

A zárójelek és az idézőjelek használata szövegkiíratás esetén elengedhetetlen. Az utasítást általában pontosvesszővel kell lezárni.

A másik kiíratő utasítás a **writeln**. Alkalmazása megegyezik a **write**-éval. A különbség köztük pusztán annyi, hogy a **write**-ot követő kiíratás ugyanabba a sorba kerül, a **writeln**-t követő pedig új sorba.

Világítsuk ezt meg néhány példával:

Ha még mindig nem egészen érthető, akkor ülj le a számítógép mellé, és variáld addig e két utasítást, amíg meg nem érted működésüket.

Példaprogram	Kiiratás eredménye
program pelda1; begin write ('Ez nem is nehéz.'); end.	Ez nem is nehéz.
program pelda2; begin writeln('Ez nem is nehéz.'); end.	Ez nem is nehéz.
program pelda3; begin writeln('Ez nem is nehéz.'); write ('Ez nem is nehéz. '); end.	Ez nem is nehéz. Ez nem is nehéz.
program pelda4; begin write ('Ez nem is nehéz. '); writeln('Ez nem is nehéz. '); end.	Ez nem is nehéz. Ez ne m is nehéz.

Gyakorló-feladatok:

1., Irj programot, amely kiírja nevedet a képernyőre!

2., Készítsd el a képernyőre saját névkártyádat!

Most már tetszőleges szöveget ki tudsz iratni a képernyőre, de ez rövidesen unalmassá válna. Vegyünk hát egy érdekesebb feladatot.

Kérdezze meg a gép, hogy mi a neved, és üdvözöljön név szerint!

Ez már nem egy egyszerű kiíratás, hanem előtte be kell olvasni egy nevet, és azt el kell tárolni. Programjaink a különféle (beolvasott és kiszámolt) adatokat úgynevezett **változó**kban tárolják.

Mi a csoda lehet az a változó?

Úgy próbáld elképzelni, mintha poharak lennének a változók. Jobb helyeken ugye másfajta pohárba töltik a bort, a sört, a pezsgőt, a konyakot, a kólát és az egyéb italokat. Talpas pohár illik a pezsgőhöz, korsó dukál a sörnek stb. Nagy ínyenc ám a Pascal nyelv is, csak ő az adatok típusa szerint tesz különbséget.

A következő változótípusokat kínálja:

Elnevezés	legkisebb, legnagyobb érték	Megjegyzés
byte	0 ... 255	Ezekben a változókban csak egész számokat tudsz eltárolni.
shortint	-128 ... 127	
word	0 ... 65535	
integer	-32768 ... 32767	
longint	-2147483648 ... 2147483647	
comp	$-9.22*10^{18}$ ... $9.22*10^{18}$	
single	$1.401*10^{-45}$ ... $3.402*10^{38}$	Ezekben a változókban törtszámokat és egészszámokat is tudsz tárolni.
real	$2.939*10^{-39}$ ... $1.701*10^{38}$	
double	$4.941*10^{-32}$ ... $1.797*10^{30}$	
extended	$3.363*10^{-49}$ ... $1.189*10^{49}$	
boolean	logikai tipus, értéke igaz vagy hamis	
char	értéke egy tetszőleges karakter lehet	
string	maximum 255 karakter hosszú szöveg lehet	

Programíráskor még a kezdetet jelentő **begin** előtt fel kell sorolnod, hogy milyen változókat akarsz használni. A változók megadását mindig a **var** szóval kell kezdeni. Programban ez a következőképpen nézhet ki:

```
program pelda;
var
  a: integer;
  c, x: real;
  betu: char;
  nev, kod: string;
  szin: string[20];
begin
  ...
end.
```

Ebben a példában az "a" nevű változó csak egész értéket vehet fel -32768 és +32767 között (lásd táblázat). A "c" és "x" változók törtszámokat is tartalmazhatnak, a "betu" egyetlen karaktert, a "nev" és a "kod" maximálisan 255 karakterből álló karakterlánc lehet, a "szin" pedig maximum 20 karakter hosszú karakterlánc.

Megnyugtatlak, hogy bőven elég a példában szereplő változótípusokat ismerned, a táblázatot csak az "ínyencek" kedvéért tettük be a könyvbe. Azt viszont figyeld meg, hogy a változónév milyen karaktereket tartalmaz. A "betu" nem trehányságból lett ilyen, hanem itt a program nem fogad el ékezetes betűket és egyéb speciális karaktereket.

Ekkora kitérő után most már elkezdhetjük feladatunk megoldását.

Hogy is szólt?

Kérdezze meg a gép, hogy mi a neved, és üdvözöljön név szerint!

Ehhez ugye szükségünk van egyetlenegy változóra, amelyben a nevet rakjuk el. Típusát illetően karakterláncnak kell lennie, és elég, ha 20 karakter hosszúságú, mert ettől hosszabb neve csak az arab tevepásztornak, meg a kolumbiai kábítószercsempésznek van.

Tehát:

```
program udvozlet;
var
  nev: string[20];
begin
  ???
end.
```

De hogyan tovább?

A beolvasó utasítás

Most ugye be kellene kérnie a programnak a nevedet. Erre a következőképpen utasíthatod:

```
readln (nev);
```

Tehát a **readln** utasítással tudsz bármiféle adatot beolvasni a billentyűzetről. Hogy melyik változóba tegye el a gép, azt a **readln** után zárójelek között kell megadni. Az adat típusa attól függ, hogy ezt a változót előzőleg milyenre állítottad be (példánkban karakterlánc, vagyis szöveges típusú).

Ezek után a program:

```
program udvozlet;
var
  nev: string[20];
begin
  readln (nev);
end.
```

Működik ez a maga módján, de olyan, mint a süketnémák beszélgetése. A **readln** utasítás ugyanis nem ír ki semmit a képernyőre, csak várja az adatot. Főleg nagy programok esetén jelent ez kellemetlenséget, arról nem is beszélve, hogy legtöbbször nem az használja a programot, aki készítette.

Jegyezd meg tehát:

Ha értelmes programot akarsz írni, akkor minden egyes beolvasó utasítás előtt ki kell iratnod, hogy milyen adatot kér a program.

Valahogy így:

```
program udvozlet;
var
  nev: string[20];
begin
  write ('Hogy hívnak? : ');
  readln (nev);
end.
```

Programunk most már megkérdezi a nevet, és azt el is raktározza, de ezután semmit sem tesz. Célunk pedig az volt, hogy üdvözöljön név szerint. Most már ez nagyon egyszerű, csak egy **write** vagy **writeln** utasítás kell hozzá.

Például a következőképpen:

```
write ('Üdvözöllek kedves ',nev);
```

Ilyen ugye még nem volt? A kiírandó szövegre eddig azt mondtuk, hogy idézőjelek közé kell tenni. Ez igaz is, mint ahogy az 'Üdvözöllek kedves ' példája is mutatja. Azonban más a helyzet a név kiíratásával. A program írásakor nem tudjuk biztosan, hogy ki fogja futtatni, de nevét a nev változó tartalmazza. Tehát ennek a változónak az értékét kell kiíratnunk, és ilyen esetben ezt nem kell idézőjelbe tenni. A vessző csak azt jelzi a gépnek, hogy ezeket írja folytatólagosan (lehetne akár több tag is).

A kész program:

```
program udvozlet;
var
  nev: string[20];
begin
  write ('Hogy hívnak? : ');
  readln (nev);
  write ('Üdvözöllek kedves ',nev);
end.
```

Illetve van itt egy kis probléma. Ha készítesz a programról EXE kiterjesztésű - futtatható - állományt és azt indítod el, akkor tökéletesen fog működni. Viszont ha a Pascalból indítod, - például ellenőrzés céljából - akkor hibásnak fog tűnni. Ugyanis a kiíratő utasítás után nem vár a program, hanem megy tovább. Ennek az a következménye, hogy az utolsó kiíratások után kilép, és a Pascal visszaadja a programlistát. Mindez olyan gyorsan történik, hogy úgy tűnik, mintha ezek a kiíratások végre sem hajtódtak volna. Ennek a látszólagos hibának a kiküszöbölésére javaslom, hogy mindig helyezz el a program végén egy "csupasz" **readln** utasítást. Igy már nem lép ki rögtön a program, hanem vár az ENTER leütéséig, Te pedig nyugodtan nézegetheted a monitort.

Tehát, ha eddigi példáinkat ezzel is kiegészítjük, akkor így néznek ki:

1. példa:

```
program pelda1;
begin
  write ('Ez nem is nehéz.');
  readln;
end.
```

2. példa:

```
program udvozlet;
var
  nev: string[20];
begin
  write ('Hogy hívnak? : ');
  readln (nev);
  write ('Üdvözöllek kedves ',nev);
  readln;
end.
```

Most, hogy tudod hogyan lehet számokat és szövegeket beolvasni a számítógépbe, változók értékeit és tetszőleges szövegeket kiíratni a képernyőre, gondolkodj el azon, hogy mindennek mi az értelme! Igy önmagában ugye nem sok. Beírni egy adatot, és változtatás nélkül viszontlátni a képernyőn - ehhez nem kell korszerű számítógép. Igazi célunk az lenne, hogy munkára bírjuk a gépet. Beírt adatainkkal végezze el azokat a műveleteket, amelyek számunkra órákat, napokat jelentenének, és a kapott eredményeket írja ki a képernyőre.

Nézzünk egy egyszerű példát!

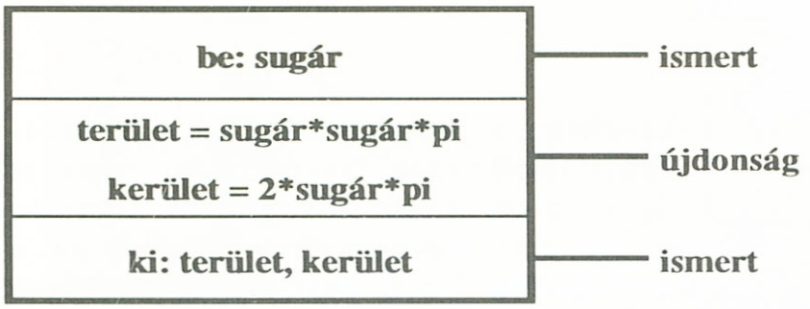
Számolja ki a gép tetszőleges sugarú kör területét és kerületét!

Eddigi tudásod alapján a következőket gondolhatod végig:

Milyen változókra lesz szükségem?

Kell egy változó a sugárnak, amit be kell olvasni a program elején, és kell egy-egy változó a területnek és kerületnek, amit a program fog kiszámolni. Mindhárom adat lehet akár egész, akár törtszám, tehát a változóknak megfelel mondjuk a real típus.

Milyen programlépések szükségesek?
Struktogrammal leírva:



Oldjuk meg azt, ami már elvileg ismert!

```
program kor;
var
  sugar, terulet, kerulet : real;
begin
  write ('Kérem a sugarat! r= ');
  readln (sugar);

  ???

  writeln ('terület= ', terulet);
  writeln ('kerület= ', kerulet);
  readln;
end.
```

Az értékadó utasítás

A számolásokat - amit most három kérdőjel jelez - a következőképpen lehet elvégezni.

terulet:=sugar*sugar*pi;

kerulet:=2*sugar*pi;

Ezt az utasítást nevezzük **értékadó utasítás**nak. A gép először kiszámolja a ":=" jeltől jobbra lévő kifejezés értékét, majd a ":=" jel bal oldalán álló változó felveszi ezt az értéket, függetlenül attól, hogy előzőleg mi volt benne. A **P** értékét nagy pontossággal ismeri a Pascal nyelv és a "pi" szóval hivatkozhatunk rá, a négy alapműveletet pedig így adhatod meg:

+ (összeadás),
- (kivonás),
* (szorzás),
/ (osztás).

És most tegyünk egy kis kitérőt a későbbi bonyodalmak elkerülése érdekében. Emlékszel még, hogy mihez hasonlítottuk a változókat? Ha nem, kérlek lapozz vissza,

és olvasd el újra a "poharas" hasonlatot! Ha most ebből a szemszögből nézzük az értékadó utasítást, akkor ez nem más, mint különböző italok adott arányú összekeverése. Én nagyon szeretem a vörösbort kólával, mások inkább gint isznak tonikkal, de nem láttam még olyat, hogy valaki sört öntött volna össze borral. Vannak tehát olyan italok, amelyek jól összeférnek, és vannak olyanok, amelyek összeöntésével garantált a gyomorrontás. A Pascal is nagyon kényes ezekre a dolgokra, ezért nem engedi, hogy egymással összeférhetetlen változókkal végezz műveleteket. Igy például nem tudsz összeadni szám jellegű (numerikus) és szöveges adatokat, mert ez értelmezhetetlen, sőt számos más ellentmondás is létezik.

Az értékadó utasítás formája tehát:

változó := kifejezés;

A kör területét és kerületét kiszámoló program pedig:

```
program kor;
var
  sugar, terulet, kerulet : real;
begin
  write ('Kérem a sugarat! r= ');
  readln (sugar);
  terulet:=sugar*sugar*pi;
  kerulet:=2*sugar*pi;
  writeln ('terület= ', terulet);
  writeln ('kerület= ', kerulet);
  readln;
end.
```

Programjaidban igen gyakran szükséged lesz arra, hogy felcseréld két változó tartalmát. (Például számok növekvő vagy csökkenő sorrendbe állításánál.) Kezdő programozóknak ez igen nagy gondot szokott okozni. Ha úgy érzed, hogy Te is közéjük tartozol, akkor figyelmesen olvasd el a következő példát, és egyszer s mindenkorra megoldódik ez a problémád.

Képzeld el, hogy a bal kezedben van egy üvegpohár tele vörösborral, a jobb kezedben pedig egy fehér műanyagpohár kólával megtöltve. Ki kell cserélned a két pohár tartalmát, vagyis az üvegpohárba kell átöntened a kólát, és a műanyagpohárba a vörösbort. Hogyan oldható ez meg? Ugye mindenképpen kell egy harmadik, egy "segédpohár" is. Ezután már egyszerű a dolog, mondjuk a vörösbort áttöltöd a segédpohárba, a kólát a most kiürült üvegpohárba, végül a bort a segédpohárból a műanyagpohárba. Ezzel a feladatot megoldottad, a segédpohárra nincs már szükséged.

Két változó értékének a felcserélésénél is ugyanezt a "játékot" kell végigjátszani. Ha például az "a" és "b" változók értékét kell felcserélni, akkor szükség van egy segédváltozóra is, ezt nevezzük például "s"-nek. A végrehajtás pedig:

```
s:=a;          s:=b;
a:=b;   vagy   b:=a;
b:=s;          a:=s;
```

Ciklusok

Az eddigi programutasítások az algoritmusoknál megismert egyszerű utasítások közé tartoznak. Segítségükkel már nagyon sok problémát meg tudsz oldani, de szintén sokat még nem. Nem véletlen, hogy az előző fejezetben annyit foglakoztunk a ciklusokkal és az elágazásokkal. A programozásban is nélkülözhetetlenek lesznek.

Kezdjük a hátultesztelő ciklusokkal.

Hátultesztelő ciklus

A borsó fejtéséhez hasonló algoritmus-szerkezetet Pascal programban is létre lehet hozni. Valószínűleg ez lesz a leggyakrabban használt ciklusfajtád. Segítségével többek között kiszűrheted a hibás adatbevitelt, kilépés nélkül újraindíthatod programodat, és még számtalan esetben nyújthat kényelmes megoldást.

Nézzünk egy példát!

Előző programunk ugye beolvassa a kör sugarának értékét, és ebből számolja ki a területet és a kerületet. Mennyivel szebb lenne, ha negatív sugarat eleve el sem fogadna. A terület és kerület képletébe ugyan ezt is be lehet helyettesíteni, sőt valami eredmény is kijön, de ennek geometriailag semmi értelme sincs. Ha tehát ezt a programot kiegészítjük a hibaszűréssel, akkor a következőképpen fog alakulni:

```
program kor;
var
  sugar, terulet, kerulet : real;
begin
  repeat
    write ('Kérem a sugarat! r= ');
    readln (sugar);
  until sugar >= 0;
  terulet:=sugar*sugar*pi;
  kerulet:=2*sugar*pi;
  writeln ('terület= ', terulet);
  writeln ('kerület= ', kerulet);
  readln;
end.
```

Hátultesztelő ciklust tehát úgy tudsz készíteni, hogy az ismételendő utasításokat a **repeat** és az **until** szavak közé írod, majd az *until után* megadod azt a *feltételt*, amelynek a teljesülése esetén ki kell lépni a ciklusból.

Néhány szó a feltételekről:

Ez annyira széles lehetőségeket rejt magában, hogy ha most mindent leírnánk róla, akkor úgy éreznéd magad, mintha egy porszívóval elküldtünk volna a Szaharába. Egyelőre tehát csak a legfontosabbakat vegyük sorra.

A példában szereplő feltétel (sugar >= 0) tartalmaz egy változót (sugar), egy konstans számértéket (0), és e két adat közötti összehasonlítást (>=). Az összehasonlítás eredményétől függően a feltétel vagy teljesül vagy nem, igaz vagy hamis.

A következő összehasonlításokat (relációkat) alkalmazhatod:

- > (nagyobb),
- < (kisebb),
- = (egyenlő),
- >= (nagyobb vagy egyenlő),
- <= (kisebb vagy egyenlő),
- <> (nem egyenlő).

Ezek segítségével már nagyon sokmindent megoldhatsz, de valamit még célszerű megjegyezned. Ha ugyanis a fenti programnak olyan adatbevitelt szeretnél csinálni, ami csak nemnegatív, de 100-nál kisebb sugarat fogad el, ezt az eddigiek alapján nem tudnád elérni. Az, hogy sugar>=0 és sugar<100, nem írható le egyetlen feltétellel. Ehhez szükség van az **and** (és) logikai operátorra. Ezzel a következőképpen nézne ki a ciklus zárósora:

```
until (sugar >= 0) and (sugar < 100);
```

Ennek a feltételnek akkor igaz az értéke, ha (sugar >= 0) is, és (sugar < 100) is teljesül. Ha csak az egyik teljesül, akkor az egész feltétel értéke hamis.

Van egy másik operátor is, az **or** (vagy). Ha ezt használod, akkor a vele összekapcsolt feltételek közül elég az egyiknek teljesülnie ahhoz, hogy együttes értékük igaz legyen.

Egy dologra azonban nagyon **vigyázz**, függetlenül attól, hogy a fentiek közül melyik relációt vagy operátort használod! Ciklusfeltételedben mindig szerepeljen olyan változó, melynek értékét a ciklusmag valamelyik utasításával megváltoztatod, mert ellenkező esetben programod úgynevezett **végtelen ciklus**ba kerül. Példánkban a sugar változó szerepel a feltételben, és beolvasás útján kap új értéket a ciklus minden egyes lefutásakor.

Elöltesztelő ciklus

Ezenkívül még további két ciklus létezik, amelyek az elöltesztelő ciklusokhoz tartoznak. Az egyik az úgynevezett **WHILE - DO** - ciklus, amelyben a feltétel megadása hasonló, mint a REPEAT - UNTIL ciklusnál, csak természetesen a ciklusmag előtt történik.

Formailag (Szintaktikája) a következő:

```
while (feltétel) do
begin
 ...
 (cilkusmag)
 ...
end.
```

Az eddigi két ciklusra jellemző, hogy az ismétlések száma meghatározatlan, a program írásakor nem kell tudnunk, hogy hányszor kell végrehajtódnia, ez csak a pro-

gram futtatásakor derül ki. Programjainkban azonban gyakran előfordul, hogy a ciklusfeltételt nem az eddig leírtak szerint tudjuk legegyszerűbben megadni, hanem egyszerű számoló ciklust szeretnénk szervezni. Nézzünk erre egy egyszerű példát!

Irja ki a gép egytől százig az egész számokat és négyzetüket. Megoldható ez az eddigi ciklusokkal is, csak van kényelmesebb megoldás, mégpedig a **FOR** - ciklus.

Ezzel a következőképpen nézhet ki a program:

```
program negyzet;
var
  x, y: integer;
begin
  for x:= 1 to 100 do
  begin
    y:=x*x;
    writeln ('X = ',x,'      X² = ',y);
    readln;
  end;
end.
```

Ebben a példában egy **növekvő ciklus**t láthatsz, ahol "x" a **ciklusváltozó**. **Kezdőértéke** 1, **végértéke** 100, e kettő között pedig egyesével változik. A FOR - ciklus kizárólag 1-es **lépésköz**t engedélyez, és ez automatikus. A ciklusmagot begin és end szavak közé kell írni, hacsak nem egyetlen utasításból áll.

Lehetőség van csökkenő ciklus szervezésére is. Nézzük az előző példát ilymódon.

Irja ki a gép száztól egyig az egész számokat és négyzetüket.

A program:

```
program negyzet;
var
  x, y: integer;
begin
  for x:= 100 downto 1 do
  begin
    y:=x*x;
    writeln ('X = ',x,'      X² = ',y);
    readln;
  end;
end.
```

Itt **egyetlen változás**t láthatsz az előző példához képest. A kezdőérték és a végérték közé a **TO** szó **helyett** a **DOWNTO** szót kell írni, ez jelzi a csökkenő ciklust.

Elágazások

Ezek után már csak a programelágazások okozhatnak problémát, de nem sokáig. Kezdjük a kétirányú elágazásokkal.

Kétirányú elágazások

Kérjen be a program egy számot, és írja ki, hogy nagyobb-e nullánál.
A program listája:

```
program elojel;
var
  a: real;
begin
  write ('Irj be egy számot!');
  readln (a);
  if a>0 then write ('Nagyobb nullánál.')
        else write ('Nem nagyobb nullánál.');
  readln;
end.
```

A kétirányú elágazást tehát az **IF** utasítással oldhatjuk meg, amely után a feltételt (a>0) kell írni. A feltétel teljesülése esetén a **THEN** utáni utasítás hajtódik végre, nem teljesülése esetén pedig az **ELSE** utáni. Az else-ág természetesen el is hagyható esetenként. Ha a feltételtől nem csak egy utasítás függ, hanem több is, akkor ezeket a **BEGIN** és **END** szavak közé kell írni a következőképpen:

```
if (feltétel) then begin
                     ...
                  end
            else begin
                     ...
                  end;
```

Többirányú elágazások

A többirányú elágazás a **CASE** utasítással oldható meg. Vegyünk egy példát! Kérjen be a gép egy osztályzatot, és írja ki betűkkel.
A program:

```
program jegyek;
var
  jegy: integer;
begin
  write ('Kérem az osztályzatot számmal!');
  readln (jegy);
  case jegy of
    1     : write ('Elégtelen');
    2     : write ('Elégséges');
    3     : write ('Közepes');
    4     : write ('Jó');
    5     : write ('Jeles');
```

```
    6..32767: begin
                write ('Ez nem osztályzat.');
                write ('Ilyen nincs.');
              end;
  end;
  readln;
end.
```

A **CASE** után annak a változónak (esetválasztó változó) a nevét kell beírni, amelytől függ az elágazás, majd ezt követi az **OF** szó. Ezután a példa szerint fel kell sorolni az esetválasztó változó számunkra érdekes eseteit, és azt is meg kell adni, hogy mi történjen ekkor. Természetesen, ha egynél több utasításunk van egy ágban, akkor ezeket a szokásos módon **BEGIN** és **END** közé kell írni. A case utasítást minden esetben **END** szóval kell lezárni.

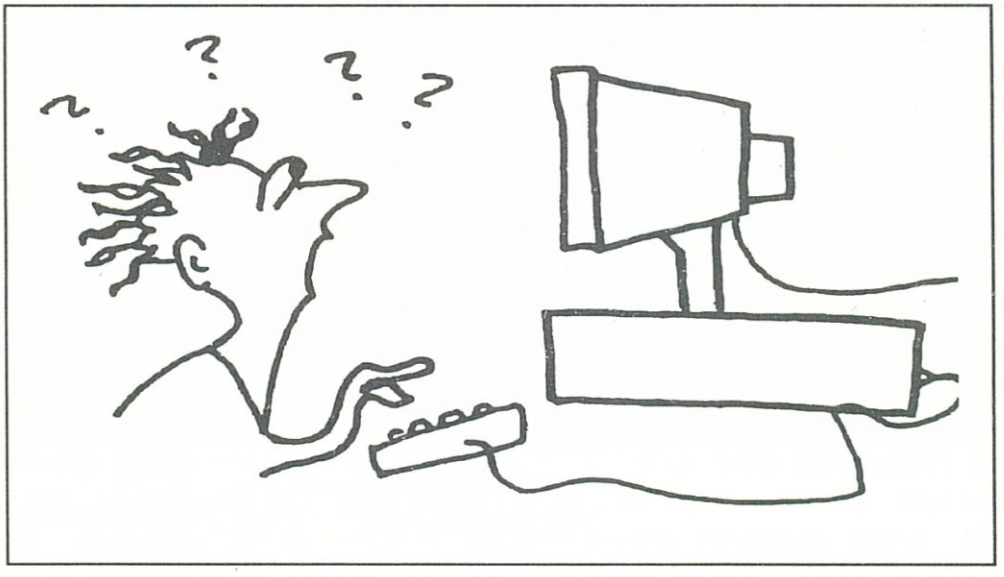
Kérdések, feladatok:

1. *Milyen egyszerű Pascal utasításokat ismersz?*
2. *Milyen ciklusszervező utasításokat ismersz?*
3. *Milyen feltételes utasításokat ismersz?*
4. *Mi a változó?*
5. *Hogyan lehet két változó értékét felcserélni?*
6. *Készíts Pascal programot, amely kiírja három beolvasott szám átlagát!*
7. *Írj programot, amely beolvas egy időtartamot másodpercben, és átváltja óra - perc - másodpercre!*
8. *Irasd ki programmal az első N db páratlan számot és négyzetüket!*
9. *Írj programot, amely kiszámolja N tetszőleges szám összegét és átlagát!*

Egy program elkészülésének lépései

A feladat meghatározása

Ez a legelső lépés a kész program felé, és mégis ezt szokta majdnem minden kezdő programozó elhanyagolni. Ennek oka az, hogy túlságosan örülnek a programozás lehetőségének. Pedig ezen áll vagy bukik a siker minden nagyobb programnál. Vegyük például egy épület felépítésének menetét: először meg kell határozni az épület pontos funkcióját, s ha ez megvan, akkor hozzá lehet kezdeni az épület megtervezéséhez. Majd be kell szerezni az anyagokat, ezután lehet csak elkezdeni az építkezést egy megadott sorrendben. Az építkezés fontosabb szakaszaiban pedig kijönnek megnézni az épületet, hogy biztonságosan, és a tervek szerint készül-e el. A legvégén pedig az épület tervrajzát, és minden hozzávalót (még a legapróbb irományokat is) egy mappában elhelyezik, hogy ha kell, akkor hamar hozzá lehessen jutni az összes információhoz. De ha nincs meghatározva az épület pontos funkciója, akkor nem lehet a tervrajzok elkészítéséhez sem hozzákezdeni. Pontosan ugyanez a helyzet a programozásnál is, mert a program megírása csak akkor történhet meg, ha már tudjuk, hogy egyáltalán mit kell tudnia a programunknak. De ha már eldöntöttük a programunk funkcióját, akkor azt rögzíteni is kell, hogy mindig elöttünk legyen a tervezés során. Ha ezt mégsem tennénk meg, akkor könnyen előfordulhat, hogy a programírás során kifelejtünk egy részt a programunkból, vagy ide-oda kapkodva tudunk csak a tervezésen átvergődni.



Kérdések:

1. *Miért hasznos feladat meghatározást készíteni ?*

2. *Mikor érdemes elkészíteni a feladat meghatározást ?*

A programtervezés

A programtervezés alatt a programunk algoritmusának elkészítését értjük. Persze ezt csak úgy lehet folyamatosan végezni, ha már elkészült a feladat meghatározás, mert a feladat meghatározás az "iránytűje" a programunknak.

A program tervezést papíron vagy szövegszerkesztőben készítjük el, és algoritmus segítségével fogalmazzuk meg a programunkat, nem pedig egy programnyelv utasításai segítségével. Van négy előnye is az algoritmussal való leírásnak:

1., Nem kell ügyelni, hogy milyen gépen dolgozunk, tehát gépfüggetlen.

2., Nem szükséges hozzá a programnyelv pontos ismerete.

3., Elkészítéséhez nem szükséges a számítógép.

4., A programunk bonyolultságának megfelelő minden programnyelvre és gépre le lehet kódolni a programunkat.

Persze a program tervezést sem lehet csak úgy összecsapni, hanem bizonyos (előnyös) taktikákat érdemes betartani.

Az első (meglehetősen régi mondásunk) egy nagyon fontos stratégia: "Oszd meg és uralkodj". A programozók ezt az elvet a lépésenkénti finomítás elvének nevezik. Értelme nagyon egyszerű: A feladatot oszd részekre, amit egymástól különállva meg tudsz oldani, de az egyes részeknek szigorúan illeszkedniük kell egymáshoz, és így részenként megoldva uralkodsz a feladat felett.

Kérdések:

1. Mi az előnye az algoritmussal való dolgozásnak?

2. Mit értünk a programtervezés alatt?

3. Mi az "Oszd meg és uralkodj" stratégia?

A megfelelő programnyelv kiválasztása

A programnyelv kiválasztására is sajnos gondot kell fordítanunk, de ezzel nem vagyunk egyedül, mert az építkezéseknél is nagy gondot kell fordítani, hogy milyen helyen építjük fel az épületünket (pl. felhőkarcolót nem érdemes földrengésekben gyakori területre építeni), és hogy milyen anyagokból építjük fel az épületünket (pl. felhőkarcolót fából nem nagyon érdemes felépíteni). A programnyelv kiválasztásakor pedig olyan funkciókat kell előtérbe helyezni, amikre a programunknak szüksége van (pl. jó grafikai felbontás, mert szép képeket akarunk kezelni, vagy gyors file-kezelés). Ha nem választjuk ki elég körültekintéssel a programnyelvünket, és a programírás felénél kiderül, hogy mégsem alkalmas a programunk feladatának megvalósítására, akkor kezdhetjük elölről a programunk írását egy másik programnyelvben. Ha ezt sem vizsgáljuk meg eléggé, akkor megint és megint kezdhetjük elölről egy másik programnyelven, míg végül inkább egy új program írásába kezdünk bele.

Kérdés:

1. Miért fontos a programnyelv helyes megválasztása?

Tesztelés

A tesztelésre például az építkezésnél azért van szükség, hogy ne fordulhasson elő hiba. Ha nem ellenőrzik folyamatosan az építkezést, akkor akár az is előfordulhat, hogy a ház alapja hibás, és nem bírja emiatt megtartani az épületet, és akkor elölről lehet az egész építkezést kezdeni. Mint eddig, most is megtalálható ez az építkezési feladat a programozásban is. A programozásnál is ugyanazt a funkciót látja el, mint az építkezésnél: a hibák időben való észrevételét.

A tesztelés fő célja az, hogy minél több hibát megtaláljunk a programunkban.

Célszerű a tesztelést olyan módszerrel végrehajtani, amellyel a próbák száma erősen csökkenthető, de ez nem azt jelenti, hogy nem alaposan végezzük el a tesztelést.

A programtesztelés módszereit két csoportba oszthatjuk aszerint, hogy a tesztelés során végrehajtjuk-e a programot vagy sem. Ha csak a program kódját vizsgáljuk, akkor STATIKUS, ha a programot végre is hajtjuk a tesztelés során, akkor DINAMIKUS tesztelésről beszélünk.

A tesztelést érdemes egy kívülálló személy segítségével végezni, mert általában a programozó a saját programját kevésbé hiszi hibásnak, mint amilyen valójában.

A statikus tesztelés fogalma

Ezt a tesztelést a program végrehajtása nélkül, a program szövegének vizsgálatával végezzük. A statikus tesztelésen belül kód-, szintaktikai (formai), és szemantikai (tartalmi) ellenőrzést különböztetünk meg. A kód ellenőrzésénél az algoritmust hasonlítjuk össze a kész kóddal. A szintaktikai ellenőrzésnél a program által megkövetelt parancsmegadás formátumát ellenőrizzük (a fordító programok is figyelmeztetni szoktak a szintaktikai hibákra). A szematikai ellenőrzésen olyan hibákat értűnk, mint pl.: egy változónak egymásután kétszer adunk értéket.

Dinamikus tesztelési módszerek

A dinamikus tesztelés fogalma

Ezt a tesztelést a program működése közben végezzük. Ha a tesztelő nem veszi elsődlegesen figyelembe a program belső szerkezetét, hanem a teszteseteket a feladatmeghatározás alapján választja meg, akkor úgynevezett fekete doboz módszerrel dolgozik. Ha pedig a teszt esetek megválasztásánál lehetőség van a program belső

szerkezetének figyelésére is, akkor úgynevezett fehér doboz módszerrel dolgozik a tesztelő. Ezek alapján két részre bonthatjuk fel a dinamikus tesztelést: fekete és fehér doboz módszerre.

Mivel sem a fekete-, sem a fehér doboz módszerrel nem lehetséges kimerítő tesztelést készíteni, a cél az, hogy az összes lehetséges teszteset csoport közül a leghatékonyabbat válasszuk ki.

A következő dinamikus tesztelési elveket minden programozó tudta nélkül is használja, mert segítségükkel elég biztosan lehet letesztelni a programunkat.

Utasítások egyszeri lefedésének elve

Lényege, hogy a programunkban szereplő minden utasítást legalább egyszer végrehajtunk.

Döntés lefedésének elve

Lényege, hogy a programunkban minden egyes elágazás igaz, ill. hamis ágát legalább egyszer be kell járnunk a tesztelés során.

Feltétel lefedésének elve

Lényege, hogy a tesztelés során olyan teszt eseteket készítünk, amelyek segítségével a döntésekben szereplő minden feltétel legalább egyszer hamis, ill. igaz értéket felvesz.

Kérdések:

1. *Mi a tesztelés fő célja?*
2. *Miért hasznos a tesztelést elvégezni?*
3. *Kivel érdemes a tesztelést végeztetni?*
4. *Mi a dinamikus tesztelés két típusa?*
5. *Mi a különbség a fehér és a fekete doboz módszer között?*

Hibakeresés és eszközei

Ha pl. egy építkezésnél már a villany bevezetésnél járunk, és kiderül, hogy egy lámpa nem ég, akkor meg kell keresni a hiba okát. Nagyon sok oka lehet a hibának, mint például: nincs felkapcsolva a biztosíték, kiégett a körte, zárlatos az aljzat, vagy például nincs is bekötve. De nem elég, ha tudjuk, hogy mi lehet a hibája, hanem meg is kell azt keresni. A programozásban is lehetnek hibák. Ezek megoldására lehet itt is ötletünk, de mint a villanyszerelésnél, itt is meg kell keresni a hiba helyes okát. A hibakeresésnek vannak alapelvei is, és eszközei is. Az alapelvei:

- addig nem szabad a hibajavítást elkezdenünk, míg a helyét és okát pontosan nem találtuk meg, mert különben elég szánalmas perceket tölthetünk el a programunk helyes visszaalakításával (ha nem volt elmentve egy előző változatunk), és mégsem jutunk előrébb
- ha a programba hiba csúszott, az a program más részeire is hatással lehet (pl. csináltunk egy hibás programrészt, viszont mindig hivatkozunk rá)
- a hibát kell kijavítani, nem csak a tüneteit kell megszüntetni (lehet, hogy a tüneteit egyszerűbb eltüntetni, de az nem segít azon a hibán, ami lehet, hogy most még nem okoz másik hibát, de később okozhat, akkor viszont már nem biztos, hogy elég jól meg tudjuk oldani a problémát).

A hibakeresési eszközök

Nyomkövetés

A nyomkövetés nyújt lehetőséget arra, hogy a program minden egyes utasítását végigkövethessük a programunkban (sokszor ezt beépítik a programnyelvekbe is).

Kiíratás

A lehető legegyszerűbb hibakeresési eszköz a kiíratás. Ez úgy működik, hogy a program megfelelő pontjain elhelyezünk kiíró utasításokat, és ilyen módon végigkövethetjük a programunk veszélyes zónáit.

Változófigyelés

A változófigyelés arra szolgál, hogy a kiválasztott változó illegális, vagy jogos megváltozásait észrevegyük.

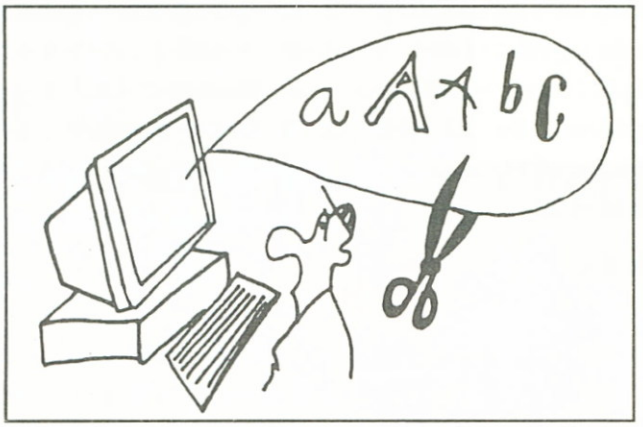
Kérdések:

1. *Mondj hibakeresési eszközöket!*
2. *Melyek a hibakeresés alapelvei?*

Dokumentáció

A dokumentáció a már elég sokszor említett építkezéseken is előfordul úgy, hogy az épülettel kapcsolatos minden információt összegyűjtik, és azt egy mappába rakják, ahonnan az összes információt könnyen előnyerhetünk, amikor csak kell. A programozásnál is hasonló a helyzet, mert itt is összegyűjtünk minden tudnivalót a programunkról, hogy a dokumentációba összefoglaljuk őket. A programunkról két fajta dokumentációt kell készítenünk: egy felhasználóit és egy fejlesztőit. Mind a két típusú dokumentáció más célt szolgál, és máshol használják is fel.

A dokumentációnak legfontosabb feladata az, hogy tájékoztassa a programról az azt megismerni kívánó felhasználót (erre szolgál a felhasználói dokumentáció), és fejlesztéskor információt adjon a program belső felépítéséről (erre pedig a fejlesztési dokumentáció szolgál).



A felhasználói dokumentáció

A felhasználói dokumentáció feladata az, hogy a felhasználót maximálisan el tudja irányítani a program útvesztőjében. De ne felejtsük el, hogy a dokumentáció sohasem pótolhatja a programon belüli segítségeket, mert nem támaszkodhatunk arra, hogy aki meg akarja ismerni a programunkat, annak mindig van a keze ügyében egy dokumentáció a programunkról.

De most térjünk rá a felhasználói dokumentáció lényegére. Nagyon fontos, hogy a dokumentáció megírásánál a legalapvetőbb számítástechnikai tudást tételezzük fel. Ez persze nem az, hogy a felhasználót lebecsüljük, hanem több segítséget próbálunk megadni neki.

Majd arra is kell gondolnunk, hogy a felhasználónak többféle program közül van esélye választani. Tehát mindent el kell követnünk, hogy a neki legalkalmasabb programra kerüljön a választása, emiatt a dokumentációnkat egy széles körű leírással kell kezdeni, ami a programunk felhasználhatósági területeit foglalja magába.

Majd a programunk hardver igényeit kell ismertetni a felhasználóval (például kell-e hozzá egér, vagy csak VGA monitoron fut, stb...).

Ezután leírhatjuk a programunk betöltésének a menetét, ami abban az esetben ad segítséget, ha nem jellemző az indítóprogram neve (például egy .bat file segítségével installáló paraméterekkel kell meghívni, vagy ha a programunk tartalmaz önmagában is indítható segédprogramokat és nem akarjuk szegény felhasználónkat belekergetni egy elmés játékba, aminek a lényege, hogy melyiket a három közül).

Ha már ezek alapján áldozatunk eljutott odáig, hogy elindítsa az évszázad programját, akkor még hátra maradhat egy gond, mikor a programunk alapján valami óhéber szövegnek látszó feliratot pillant meg a képernyőn. Tehát az a harci feladat, hogy felvilágosítsa a felhasználót arról, hogy mit vár ilyenkor a program a billentyűzetről (például mikor megkérdezi a zseniális programunk, hogy: Milyen idő van odaát? nem a szomszéd szoba időjárásjelentése felől érdeklődik, hanem a hawaii-i géptermekéről).

De persze ez még nem minden, mert még a hibaüzenetek is ide tartoznak. De ez még bonyolultabb is egy kicsit, mert a felhasználó a program kérdéseire még meg tudja adni a választ egy kis gondolkodás után, de a hibaüzeneteknél már nem elég az, hogy megérti, hanem azt ki is kell tudni javítani. Tehát minden hibát (még a legérthetőbbeket is) ki kell fejteni megoldási lehetőségeikkel együtt is (mert például, ha a 100 éves nagymama arra gondol, hogy mostantól a receptjeit számítógépen fogja tárolni, és a hosszas gépelés után kiderül, hogy nincs elég hely a háttértárolón, akkor még arra is képes lehet, hogy papírokat kezdjen tömködni a lemezes egységbe, hogy arra mentse ki a receptjeit, bezzeg, ha jól le van írva a hibamegoldás menete, akkor tudja, hogy csak egy üres lemezzel kell a szegény számítógépet kisegíteni).

A fejlesztői dokumentáció

A fejlesztői dokumentációnak az elkészítése lényegesen könnyebb, mint a felhasználói dokumentációé. A fejlesztői dokumentáció már nem a program használatához ad segítséget, hanem a program belső felépítéséről. Erre akkor van szükség, mikor a programunkat akarjuk fejleszteni, kb. egy fél év múlva. Ebben az esetben már nem biztos, hogy a programunk minden részletére pontosan emlékezni fogunk. Ilyenkor lép érvénybe a fejlesztési dokumentáció, mert ennek segítségével lehet csak megfelelően feleleveníteni tökéletesen és rövid idő alatt a programunk belső felépítését.

A fejlesztői dokumentációnak tartalmaznia kell a program feladatának pontos meghatározását, amit még a program megírása előtt kell elkészíteni.

Ezután a megoldás algoritmusa, a változók táblázata és a program kódja következik. Ezeket a program algoritmusának és kódjának elkészítésekor kell elkészíteni.

Majd a tesztelés leírása, és ezenbelül a teszteredmények leírása következik.

Ezután a program fejlesztési lehetőségeinek, ill. korlátainak leírása következik, ami tudatja a kedves fejlesztővel, hogy mennyire tudja fejleszteni a mi programunkat.

Ha valami megmagyarázhatatlan okból (pl. ha a nagymamának ilyen program volt az álma a receptjei részére) a megoldás folyamán sajátos tervezési vagy kódolási szabályt alkalmaz, akkor érdemes azt is leírni.

A dokumentáció szépségszalonja

Sajnos ez is kell, mert nem mindegy, hogy milyen dokumentációt adunk ki a kezünk közül.

Először is a dokumentáció ne legyen túl hosszú, de túl rövid se, mert a végletek csak végzetet okozhatnak.

A dokumentáció ne legyen túlságosan tagolt vagy cicomázott, mert ezek elterelik a véres verejtékkel összekotort munkánkról az olvasó figyelmét. De azért legyen eléggé tagolt, hogy az egyes részek legyenek rendesen elkülönítve és tisztán kivehetők legyenek az egyes fejezetek.

Kérdések:

1., Mi a dokumentáció legfőbb feladata?

2., Mi a feladata a felhasználói dokumentációnak?

3., Mi a feladata a fejlesztői dokumentációnak?

4., Milyen főbb dolgokra kell ügyelni a dokumentáció elkészítése közben?

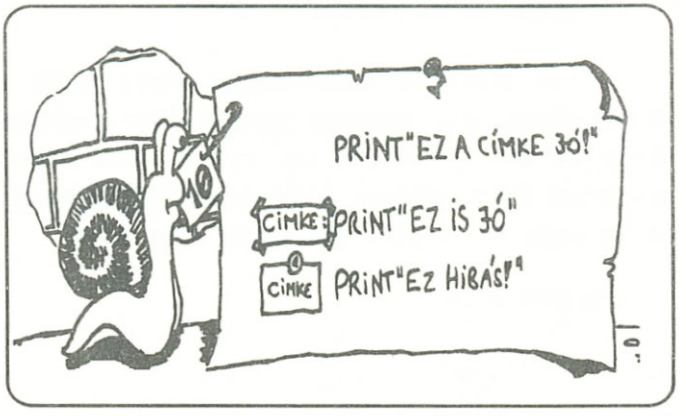
Bevezetés

Ebből a fejezetből megtudhatod, hogyan is épül fel egy Quick Basic program. Ha ezt már tudod, akkor nyugodtan ugord át, de azoknak, akik most ismerkednek a nyelvvel, nem árt elolvasni.

A Quick Basic programozási nyelv utasításkészlete nagyjából megegyezik az általánosan elfogadott BASIC nyelv utasításkészletével. Különbség csak abban van, hogy a Quick BASIC-ben speciális utasítások is vannak (ezek által kezelhetjük a file-okat, a grafikát, stb.), sőt, a fejlettebb verziókban alprogramokból (szubrutinokból) is össze lehet állítani programunkat, amelynek segítségével struktúrált programokat tudunk írni. A nyelv emiatt egy kissé hasonlít a Pascal nyelvre és a más struktúrált programozási nyelvre is.

Programjainkat - mint minden más fejlettebb nyelvben - egy szövegszerkesztő segítségével írhatjuk meg. Magára a szövegszerkesztő kezelésére most nem térnék ki, ezt a tanárod úgyis megmutatja. (Elég róla annyit tudni, hogy a begépelt programot a **SHIFT** és az **F5** billentyűk együttes lenyomásával futtathatod.)

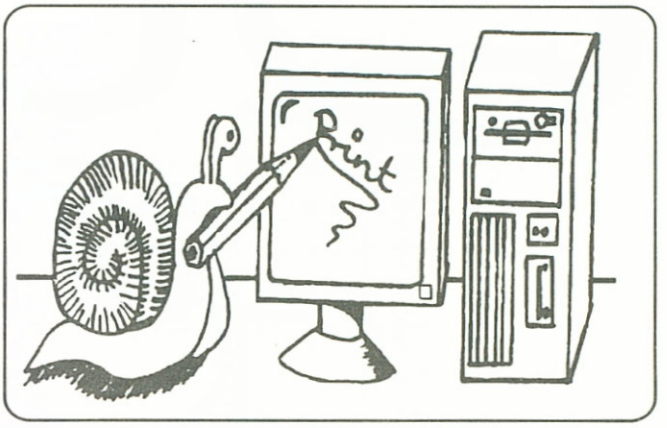
A Quick Basic nagy előnye a régebbi basic nyelvekkel szemben az, hogy a programsorokat nem kell sorszámozni, a gép enélkül is meg tudja különböztetni azokat. Ez persze jelentősen megkönnyíti a munkát. Mert gondoljunk bele - ha két sorszám között már nincs "hely" - pedig ide szeretnénk beszúrni - akkor az egész programot át kellene sorszámozni.



Ha egy programrészt mégis meg szeretnénk jelölni, elég csak "elnevezni" egy tetszőleges névvel, vagy esetleg egy sor-

számmal. Ezt hívják **címkézés**nek. Ha a címkének egy szót adtunk meg, azt kettősponttal kell befejezni. Két azonos címkenevet nem adhatunk meg, mert már a beírásakor a "Duplicate Label" (Duplázott címke) hibaüzenetet kapjuk.

PRINT - firkáljunk a képernyőre !



Egyszer mindenkivel előfordult már, hogy az unalmas órák egyikén elkezdte "kidíszíteni" az éppen előtte lévő könyvet. Ezek alól kivétel a számítástechnika-óra, ahol talán nincs is könyv amibe firkálni lehetne. Az élelmes diák ekkor új ötleteken kezdi törni a fejét: "Ott áll az asztalon egy szürke doboz, amin néha különféle szövegek jelennek meg. Az egész feltűnően hasonlít egy könyvre. Miért is ne próbálnám meg összefirkálni ?".

A monitor "összefirkálásához" csak egy egyszerű basic utasítást kell ismerni, és máris mindenféle szöveget jeleníthetünk meg a képernyőn. Ennek azonban megvan az az előnye is a könyvekkel szemben, hogy egy másik utasítással le is törölhetjük azt. Ez a **CLS** utasítás.

A képernyőre szöveget kiírni a **PRINT** utasítással lehet. A kiírandó szöveget idézőjelek közé kell tenni, mert a gép így különbözteti meg azt a változók nevétől. Például írjuk be a következő programot, majd futtassuk le :

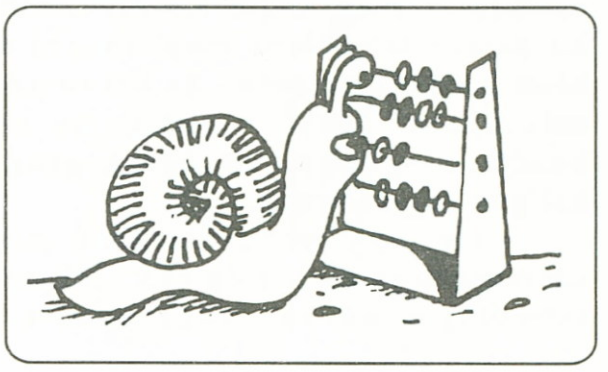
```
PRINT "QUICK BASIC"
```

A **SHIFT-F5** megnyomása után a képernyőre kiíródott a QUICK BASIC felirat, és a legalsó sorban kiíródott egy üzenet: "Press any key to continue". Ez azt jelenti, hogy a program futása befejeződött és a gép egy billentyű lenyomására vár. (Ha lenyomtunk egy gombot, akkor újból a szövegszerkesztőbe jutunk.)

Felfigyelhettünk arra, hogy ha volt már valami a képernyőn, akkor az is látszik, és ha a programot megint lefuttatjuk, akkor láthatjuk, hogy a következő felirat az előző alá íródott ki. Ezt elkerülendő, töröljük a képernyőt a **CLS** utasítással!

```
CLS
PRINT "QUICK BASIC"
```

A gép, mint számoló zseni



A **PRINT** utasítással nem csak szövegeket írathatunk ki, hanem számolási műveletek eredményét is. Például, ha valaki azt szeretné tudni, mennyi 5+5, akkor elég csak egy rövid programot megírnia, és máris okosabb lesz tőle:

```
PRINT 5+5
```

Persze ennél sokkal bonyolultabb számításokat is elvégezhet a gép, csak tudni kell, hogyan írjuk meg ezeket programsorban. Például:

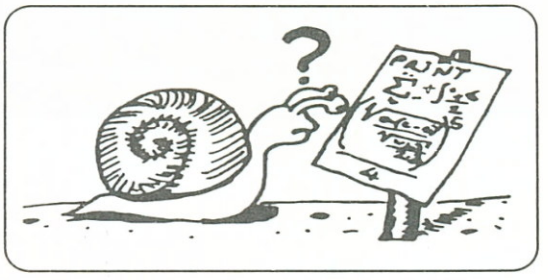
PRINT 7+(15-7)/2

Ugye ezt már nem lenne könnyű fejben kiszámítani? De ha a programot elindítjuk, a gép mégis villámgyorsan kiírja a végeredményt: 11. De vajon mi történik, ha a következő sort gépeljük be?

PRINT "7+(15-7)/2=";7+(15-7)/2

A program futtatása után nem csak az eredményt, hanem a számolást is kiírta a gép, valahogy így:

7+(15-7)/2= 11



Ez persze nem volt boszorkányság, hiszen csak annyit csináltunk, hogy a műveletet kétszer írtuk be, egyszer idézőjelekben, és egyszer simán. Ebből kiderül az, hogy a gép az idézőjelbe tett jeleket változatlan formában írja ki, az idézőjel nélkülieket -ha tudja- kiszámolja.

A gép által elvégezhető műveletek a következő táblázatban láthatók:

Jel	Művelet	
+	összeadás	Aritmetikai műveletek
-	kivonás	
*	szorzás	
/	osztás	
^	hatványozás	
MOD	maradékos osztás	
\	egész osztás	
=	egyenlő	Relációk
>	nagyobb	
>=	nagyobb egyenlő	
<	kisebb	
<=	kisebb egyenlő	
<>	nem egyenlő	
NOT	nem	Logikai műveletek
AND	és	
OR	vagy	
XOR	kizáró vagy	

Természetesen lehetőségünk van egy-két bonyolultabb függvény használatára is:

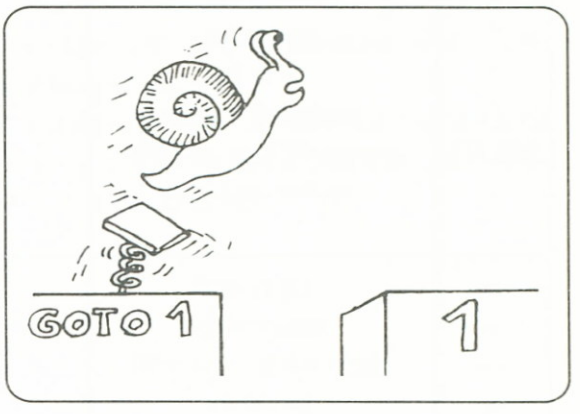
ABS() - abszolút érték számítása
COS() - koszinusz számítása
EXP() - exponenciális függvény
LOG() - természetes alapú logaritmus
RND() - véletlenszám generálása 0 és 1 között
SGN() - előjelfüggvény
SIN() - szinusz számítása
SQR() - négyzetgyök számítása
TAN() - tangens számítása

(Ezek használatára majd később rátérünk)

Az iménti programban felfigyelhetünk még egy furcsaságra is, a pontosvesszőre. Vajon mit jelenthet ? A magyarázat egyszerű: a **PRINT** utasításnak többféle kiíratási módja is van. Bizonyára mindenki észrevehette azt, hogy ha többször íratunk ki egy eredményt, az a következő sorba íródik ki. Ha viszont egy sorba többféle értéket is szeretnénk kiírni, akkor használhatjuk a ';' karaktert, ugyanis ennek hatása az, hogy a kiíratás után a kurzor nem ugrik a következő sor elejére. Ha pontosvessző helyett vesszőt használunk, akkor a kiírás után a kurzor 12 karakterrel továbbugrik. (Ezt hívják tabulálásnak.)

A programban még azt is észrevehettük, hogy a **PRINT** utasításnál egymás után többféle típusú adatot is kiírathatunk, azaz felváltva írathatjuk ki a szövegeket, a számolások eredményeit, és a változók értékeit.

GOTO - avagy ugráljunk egy kicsit !



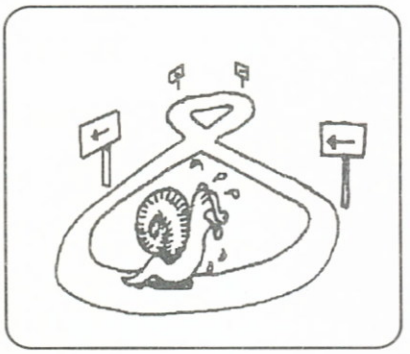
Ha valaki azt kérné tőlünk, hogy írjuk tele a képernyőt egy szöveggel, akkor bizony elég sok munkánk lenne, mert minimum 80*25, azaz 2000 karaktert kéne kiírnunk a képernyőre. Ez még a legegyszerűbb megoldásban is (amikor egy PRINT utasítással egyszerre 80 karaktert íratunk ki) 25 sorból állna. Persze ezeknek a beírását megspórolhatjuk úgy is, hogy begépelünk egy ilyen sort, és a szövegszerkesztő másolási funkcióival huszonötször lemásoljuk, de ennél van egy sokkal elegánsabb megoldás is.

A legegyszerűbb megoldás erre az, hogy egy karakter kiíratása után visszaugrunk ugyanerre a sorra, és újból kiírunk egyet. Erre az ugrálásra a **GOTO** utasítást használhatjuk. A **GOTO** utasítás után azt is meg kell adnunk, hogy milyen cimkére ugorjon a program. Tehát az egész képernyőt kitöltő program:

```
IDE: PRINT"*";
   GOTO IDE
```

Ha elhagyjuk a sor végéről a pontosvesszőt, akkor csak az első oszlopba íródik ki csillag.

A program futtatása után a gép nem jelzi ki; hogy befejezte volna a munkát, hanem fáradhatatlanul írja ki a képernyőre a csillagokat. Tehát a gép végtelen ciklusba jutott.



Ezt akkor is végezné, ha közben kivennénk évi rendes szabadságunkat, és elutaznánk a Bahama-szigetekre. Ha a kéthónapi nyaralás után megnéznénk, programunk akkor is írogatná a képernyőre a csillagokat (ha közben tönkre nem ment a gép a túlerőltetés miatt).

Ha valaki nem hinné el, akkor nyugodtan próbálja ki a következő programot, és láthatja, hogy a GOTO után következő sorhoz nem jut el a program végrehajtása:

```
IDE: PRINT"*";
   GOTO IDE
   PRINT"Kész vagyok!"
```

Ha viszont már kigyönyörködtük magunkat a csillagos képernyő látványában és szeretnénk folytatni a programozást, akkor nyomjuk meg a **CTRL** és a **PAUSE** billentyűket! Erre a programot megszakítva a szerkesztőbe kerülünk vissza. (Ezt a billentyűkombinációt bátran használhatjuk más esetben is, ha a programunkból szeretnénk visszalépni a szövegszerkesztőbe.)

CIKLUSOK

Persze az előző program nem igazán alkalmas feladatok elvégzésére. Ezért van a Quick Basic-nek egy utasítása, amellyel akárhányszor végrehajtathatunk egy utasítást, vagy egy utasítássorozatot. Ezt számláló ciklusnak nevezik. A **FOR ... NEXT** utasításpárral használhatjuk. Először is nézzünk egy példaprogramot:

```
CLS
FOR I=1 TO 2000 STEP 1
  PRINT "*";
NEXT I
```

Mit is csinál a **FOR...** és a **NEXT...** kezdetű sor a programunkban? A **FOR** után megnevezünk egy változót (egy olyan fogalmat, amelyhez különböző számértékek tartozhatnak), az **I**-t, és megmondjuk a számítógépnek, hogy ennek az **I**-nek az értéke 1-től 2000-ig terjedhet, és végül azt, hogy az I hányas lépésközzel számlálódjon (**STEP 1**) (persze ha csak egyesével számlálunk, akkor ezt el lehet hagyni). Valamint az is természetesnek tűnhet, hogy ha az első szám nagyobb, mint a második (a **TO** előtti és

utáni), akkor a lépésköznek negatívnak kell lennie. A **FOR**-ral kezdődő sor tehát így fordítható le magyar nyelvre: "I minden értékére hajtsd végre a **FOR** és a **NEXT** közötti utasításokat, miközben az **I** értékét növelgeted 1-től 2000-ig."

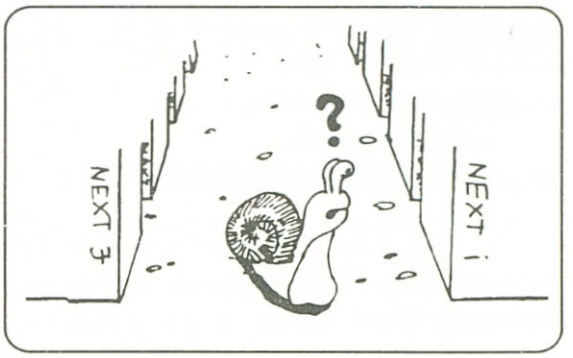
Erről legjobban akkor győződhetünk meg, ha begépeljük a következő programot, ami a páros számokat írja ki 2-től 100-ig:

```
FOR I=2 TO 100 STEP 2
  PRINT I,
NEXT I
```

A ciklusok és a számolási műveletek ismeretével már meg tudunk oldani olyan feladatot is, amikor mondjuk egy függvény értékeit kell kiszámoltatni a géppel bizonyos határok között. Erre egy példa: számoljuk ki a 2X+5 függvény értékeit 1 és 100 között!

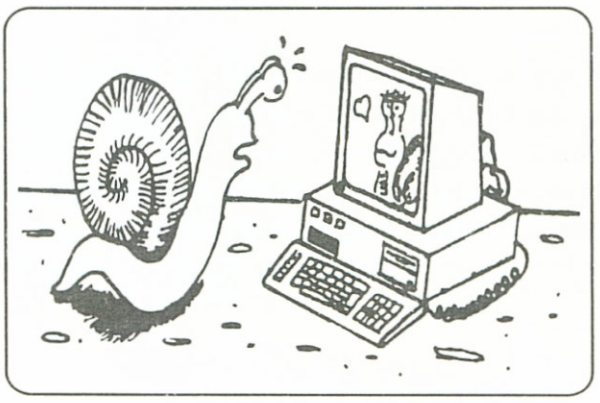
```
FOR X=1 TO 100
   PRINT 2*X+5,
NEXT X
```

A ciklusokat egymásba is ágyazhatjuk. Ez azt jelenti, hogy egy cikluson belül még egy ciklus van, azon belül még egy, és így tovább. Persze ilyenkor ügyelni kell a cikluskezdetek és ciklusvégek helyes sorrendjére, mert esetleg a gép eltévedhet a ciklusok útvesztőjében. A durvább hibákat persze már futtatás előtt felfedezi a gép, és kiírja a "NEXT without FOR" (NEXT FOR nélkül) hibaüzenetet. De most lássunk az egybeágyazott ciklusra egy rövid példát: Írjuk ki a képernyőre a 10*10-es szorzótáblát!




```
CLS
FOR I=1 TO 10
  FOR J=1 TO 10
    PRINT I*J;
  NEXT J
  PRINT
NEXT I
```

A változó gyönyörködtet



Az előbbi ciklusfajta ismertetésénél feltűnt egy új fogalom, a változó. Mik is azok a változók? A változók mindenféle adat tárolására alkalmasak. Képzelünk el sok kis dobozt, amit ha kinyitunk, beletehetünk egy tetszőleges számmal vagy szöveggel ellátott cédulát. A cédula tartalmát a program futása során bármikor leolvashatjuk, vagy megváltoztathatjuk. Ezeket A fiókokat elnevezhetjük tetszőleges (viszonylag rövid) névvel, de két azonos nevű változó nem lehet a programban.

Persze a dolog nem ennyire egyszerű, ugyanis nem mindegy, hogy mi van a cédulára írva. A szövegeket tartalmazó cédulákat ugyanis a gép más típusú fiókokban tárolja, mint amiken számok vannak. A fiókok típusát a nevük után írt speciális jellel adhatjuk meg.

A változók típusait a következő táblázatban láthatod:

Változótípus	Jelölés	Érték	Byte
INTEGER	%	-32768...32767	2
LONG	&	-2147483648...2147483647	4
SINGLE	!	7 számjegy egyszeres pontos	4
DOUBLE	#	15 számjegy kétszeres pontos	8
STRING	$	max. 32768 karakter	32768

A változóknak az egyenlőségjellel adhatunk értéket a programban, ezenkívül mindenféle műveletet is végezhetünk velük. Ennek bemutatására szolgál a következő példa:

```
A=5
B=3+2
C=A+B
PRINT C
```

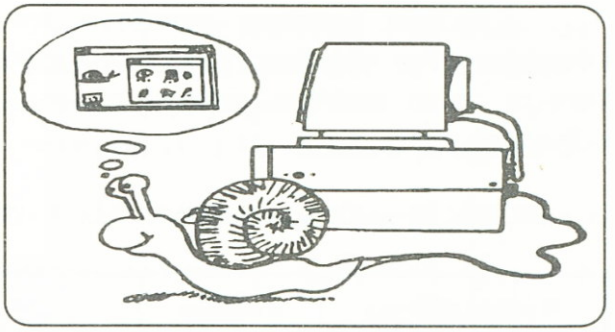
A program futtatása után a gép kiírta a C változó értékét (10). Ez még mindig nem több, mint amit egy átlagos nyolcéves gyerek egy zsebszámológéppel ki tud számolni, de csak nyugalom, ez csupán az alap, ennél sokkal bonyolultabb művelet is lesz még:

```
A$="QUICK"
B$="BASIC"
C$=" "
D$=A$+C$+B$
PRINT D$
```

Az előbbi programra kissé értetlenkedve tekinthetünk. Az itt használt változókat hívjuk karakterláncoknak, angolul STRING-eknek. Mint látszik, itt is az idézőjel használatával különböztethetjük meg a sima szöveget a változók nevétől. Ahogy a táblázatban láthattuk, ezeket a szövegeket sztring típusú változóban tárolhatjuk el, amelyek jelölése a $ jel.

A karakterláncokat ugyanúgy adhatjuk össze, mint a más változókat, azzal a különbséggel, hogy míg más esetben a végeredményt számolja ki a gép, a szöveges változóknál csak a tartalmukat olvassa össze, fűzi egybe (**D$="QUICK BASIC"**)

Persze kényelmetlen mindig kézzel beírogatni a programlistába a változók értékeit, mi lenne ha maga a gép kérné be azokat? Erre használhatjuk az INPUT utasítást:



```
INPUT A
INPUT B
C=A+B
PRINT C
```

Látható, hogy programunk használata így már sokkal kényelmesebb. Azonban mi van akkor, ha a programot más is használni szeretné, akinek fogalma sincs arról, hogy mit írjon be? Mennyivel elegánsabb lenne, ha a gép kiírná a kérdést is:

```
INPUT "Kérem adja be az A értékét:";A
INPUT "Kérem adja be a B értékét:";B
PRINT "A+B=";A+B
```

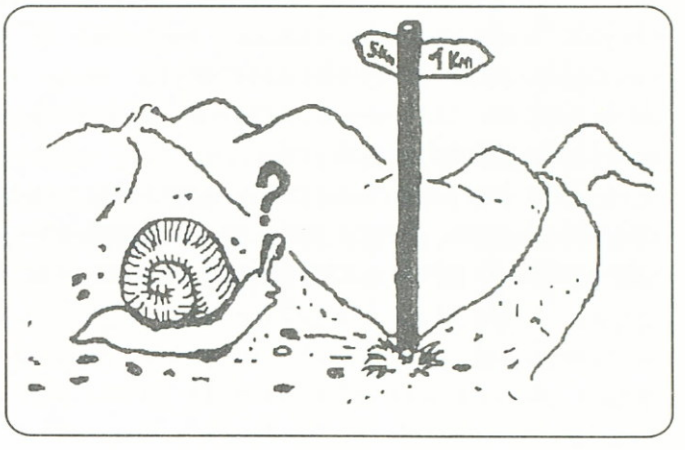
Ennyi ismerettel már nyugodtan elkészíthetünk egy névjegykártyaíró programot:

```
CLS
INPUT "Hogy hívnak ?";N$
INPUT "Melyik városban élsz ?";V$
INPUT "Melyik utcában ?";U$
INPUT "Mi az irányítószám ?";I$
INPUT "Mi a házszám ?";H$
CLS
PRINT "Névjegykártyád:"
PRINT
PRINT N$
PRINT V$;" ";U$;" ";H$
PRINT I$
```

(Itt megjegyezném azt, hogy a PRINT utasítást lerövidítheted, helyette kérdőjelet (?) használhatsz.)

Feltételes módban - az elágazások

Sokmindent ismerünk már a Quick Basic sajátosságairól, de egy igen fontos lehetőségéről még szót sem ejtettünk.



Eddig kétféle módon dolgoztattuk a gépet: vagy úgy, hogy sorra végrehajtotta a programsorokat, vagy pedig "külön kérésre" kilépett ebből a folyamatból és a **GOTO** utasítás hatására átugrott egy más programrészre. A gépet azonban feltételes módban is utasíthatjuk. Hogy ez hogyan megy, arra nézzünk meg egy rövid kis programot:

```
IDE: A=A+1
   PRINT A;
   IF A<100 THEN GOTO IDE
```

Na, ezen aztán van mit nézni. Vegyük például az első sort. Itt található egy furcsaság (**A=A+1**), ami ellentmond mindazzal, amit eddig matematikából tanultunk. Hogy lehet **A** egyenlő (**A+1**)-gyel? Hát úgy, hogy ez nem matematika, hanem **BASIC**, és ez nem egyenlet, hanem a gépnek adott utasítás. Azt jelenti pontosan: "az **A**-nak nevezett változó értékét növeld meg eggyel."

Az igazi érdekesség a harmadik sorban van. Az a talán nem ismeretlen jel azt jelenti "kisebb, mint", és ott van még két ismeretlen szó: **IF** és **THEN**. Ezeknek pontos magyar megfelelője: **HA** és **AKKOR**. E két szó már sugallja azt a tényt, hogy a harmadik programsor jelentése: "Ha az **A** változó értéke kisebb, mint 100, akkor ugorj az '**IDE**' címkére."

A gép tehát egyesével növelgeti a változó értékét, és mindig ellenőrzi, hogy elérte-e a 100-at, ha nem érte el, akkor visszaugrik a program elejére. Ezt egyébként hátultesztelős ciklusnak is hívják.

A hátultesztelő ciklusnak van azonban a Quick Basic-ben egy külön utasítása is. Erre csak egy példát mutatok, talán nem kell bővebben magyarázni:

```
DO
  A=A+1
  PRINT A
LOOP UNTIL A > 99
```

Ennek van egy másik változata is, amikor elöl tesztelünk (elöltesztelő ciklus):

```
WHILE A<100
  A=A+1
  PRINT A
WEND
```

Most pedig játsszunk egy kicsit! A programmal lemérheted, hogy mennyire jól tudsz fejben számolni:

```
    INPUT "Hogy hívnak ?";N$
    PRINT "Hello kedves ";N$
    FOR I=12 TO 60 STEP 3
L1: PRINT "Üss be két számot, amelyek összege:";I
    INPUT A
    INPUT B
    IF A+B=I THEN PRINT "Oké, helyes a válasz!":GOTO L2
    PRINT "Sajnos ez rossz ! Próbáld újra!":GOTO L1
L2: NEXT I
```

(Ebben a programban észrevehető, hogy az utolsó előtti két sorban kettőspontot (:) alkalmaztunk az utasítások elkülönítéséhez, így még egy kicsit takarékoskodhatunk is a sorok számával.)

Nincsenek véletlenek - vagy mégis?

Most a BASIC egy igen érdekes függvényével ismerkedhetünk meg. Mint láthattuk, a fejszámoltató programnak az volt a hátránya, hogy többszöri futtatás után

is csak ugyanazokat a számokat kérdezi. Ez érthető, mivel a számokat egy egyszerű számlálós ciklus adta meg. Vajon mi lenne, ha nem mi adnánk meg a számokat, hanem maga a gép találná ki azokat? Erre az **RND** függvényt használhatjuk. Ilyenkor az embernek az az érzése támad, mintha a gép önállóan döntene bizonyos dolgokban. Pedig sajnos ez nem így van, ugyanis a gép az **RND** függvény meghívásakor egy véletlenszámot generál **0** és **1** között. Ezt a számot egyébként különféle matematikai módszerekkel állítja elő. Ennek kipróbálásához nézzük a következő programot:

```
IDE: PRINT RND(1)
   GOTO IDE
```

Ezt úgy hasznosíthatjuk, hogy a generált számnak megadjuk a határait. Ezt egy kicsit furcsán kell megtenni, inkább egy példa: generáljunk 3 véletlenszámot 1 és 10 között!

```
FOR I=1 TO 3
 A=INT(RND(1)*10)+1
 PRINT A
NEXT I
```

Gondolom némi magyarázatra szorul a második sor. Tehát: az **A** változóba szeretnénk rakni a véletlenszámot. Az **INT** függvény az egész részét képezi egy törtszámnak (azaz **INT(5.3)** eredménye **5**). A függvénynek be kell adni egy paramétert (ez **0** vagy **1**), ami megadja, hogy a véletlenszám-generálásban szerepe legyen-e a gép belső órájának is. Azt tudjuk, hogy a szám 0.0 és 0.9999999 között van, tehát meg kell szorozni tízzel. Végül pedig hozzá kell adni egyet, hiszen nullát nem szeretnénk eredményül kapni.

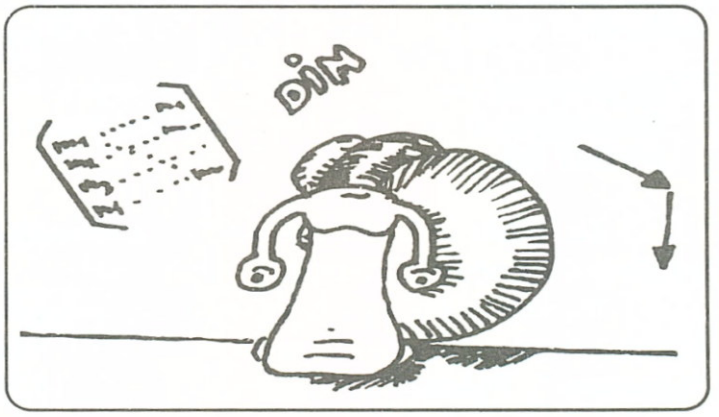
Most pedig következzen még egy kis játék. Gondolom, mindenki ismeri azt a játékot, hogy a gép kitalál egy számot, és nekünk rá kell jönnünk:

(a gép 1 és 100 között találja ki a számokat)

```
KEZD: CLS                    <- ez törli a képernyőt
   INPUT"Hogy hívnak";N$
   PRINT"Hello ";N$;", örülök, hogy megismertelek."
   INPUT"Kezdhetjük (i/n)";B$
   IF (B$="n") OR (B$="N") then goto kezd
   P=0
   V=INT(RND(0)*100)+1
   WHILE (P<=10)
    P=P+1
    INPUT"A te tipped";T
    IF V<T THEN PRINT"Az én számom kisebb!"
    IF V>T THEN PRINT"Az én számom nagyobb!"
    IF V=T then goto VEGE
   WEND
```

```
VEGE: IF V=T THEN PRINT "Szuper! Kitaláltad ";P;". próbálkozásra!":goto L
      PRINT "Sajnos ez most nem ment,talán legközelebb..."
L:    INPUT"új játék (i/n)";B$
      IF (B$="i") OR (B$="I") then goto kezd
```

Vektorok, mátrixok és más riadalmak



Ha valaki vektorokat emleget előttünk ne szaladjunk el messzire - ezek sem veszélyesebb fogalmak, mint amilyenekkel eddig megismerkedtünk, ugyanis a vektorok olyan változókból álló csoportok, amelyek azonos néven vannak nyilvántartva, de megszámozzuk őket, hogy áttekinthetőbb legyen a programunk. Például képzeljünk el egy programot, ami egy osztály taulóinak névsorát olvassa be. Az ugye teljesen világos, hogy ha minden egyes névnek egy új változót hoznánk létre, akkor előbb-utóbb összezavarodnánk. De mi van akkor, ha csak egy olyan változót hozunk létre, aminek rengeteg eleme lehet? Ilyen változókat a **DIM** utasítással készíthetünk el. Például csináljuk meg az előbb említett osztálynévsort beolvasó programot:

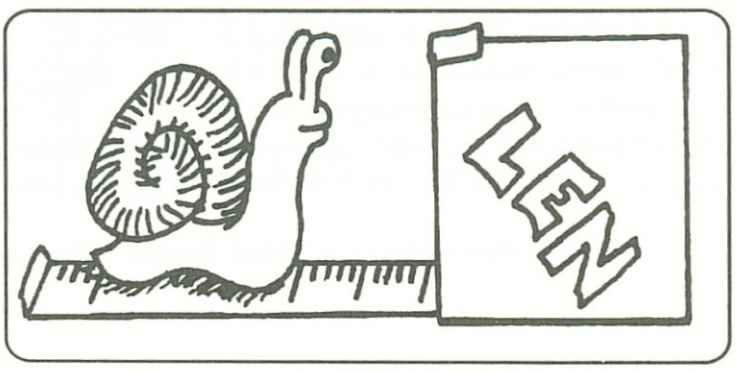
```
CLS
INPUT "Hány tanuló ?";T
DIM A$(T)
FOR I=1 TO T
 PRINT I;".név:";
 INPUT A$(I)
NEXT I
FOR I=1 TO T
 PRINT A$(i)
NEXT I
```

Az egydimenziós vektorok mellett előbb-utóbb szükségünk lesz két- és többdimenziós tömbök kezelésére is. Ebben a könyvben azonban nem megyünk tovább a kétdimenziósnál.

A kétdimenziós tömböt (mátrixot) a legegyszerűbben úgy képzelhetjük el, mint egy táblázatot, amely számozott oszlopokra és sorokra van felosztva. Például csináljunk egy 8*8 elemből álló kétdimenziós tömböt:

```
DIM A(8,8)
```

Sztringműveletek



Korábban már megismertük azt, hogyan lehet a sztringeket "összeadni". A sztringekkel még sok más műveletet is elvégezhetünk. Például ha arra lennénk kíváncsiak, hogy egy szöveg hány karakter hosszú, azt könnyen megtudhatjuk a **LEN** függvény segítségével:

```
A$="EZ HUSZ BETUBOL ALL!"
A=LEN(A$)
PRINT A
```

Ha viszont szeretnénk megtudni, hogy egy sztringben egy bizonyos pozíción, vagy egy bizonyos pozíciótól kezdve milyen karakterek állnak, azt a **MID$** függvénnyel tudhatjuk meg. Például szedjük ki a "HELLO" szövegből azt, hogy "LO":

```
A$="HELLO"
B$=MID$(A$,4,2)
PRINT B$
```

Itt tehát először azt kell megadnunk, hogy az **A$** sztringből akarunk kivenni a **4.** pozíciótól kezdve **2** betűt.

Ennek van egy egyszerűbb változata is, ezt akkor használhatjuk, ha csak a karakterlánc *bal vagy jobb oldalától* szeretnénk egy bizonyos mennyiségű elemet "kiszedni". Például szedjük szét a **"QUICKBASIC"** szöveget kétfelé :

```
A$="QUICKBASIC"
B$=LEFT$(A$,5)
C$=RIGHT$(A$,5)
PRINT B$
PRINT C$
```

Most pedig egy rövid példa a sztringfüggvények használatára: a program bekér egy szöveget, és fordítva írja ki azt a képernyőre:

```
INPUT"SZÖVEG";A$
FOR I=LEN(A$) TO 1 STEP -1
  PRINT MID$(A$,I,1);
NEXT I
```

A "nagyobb", "kisebb" jelekkel végezzünk el most egy meglepő kísérletet ! Gépeljük be ezt a kis programot :

```
INPUT "Első név";A$
INPUT "Második név";B$
IF A$<B$ THEN PRINT A$,B$:END
PRINT B$,A$
```

(Az **END** utasítás a program befejezésére szolgál)

Ez a program ABC-sorrendbe állít két bármilyen nevet (vagy más szót, de a sztringekben csak a normál ABC betűi lehetnek benne, ékezetes és speciális karaktereket nem tartalmazhatnak). Szokatlan gondolat szöveges változók esetében a < jelet használni, hiszen hogyan kisebb **A$** a **B$**-nél? Hát úgy, hogy a bennük lévő karakterek **ASCII-kód**ja az ABC sorrendjében növekszik, és a gép ezeket a kódszámokat hasonlítja össze.

Átalakító (konverziós) függvények

Mivel a változóknak többféle típusa van, ahhoz, hogy egyik típusú változó értékét használhassuk egy másik változóban, át kell alakítani, konvertálni kell. Erre valók a konverziós függvények. Hogy ezeket mire is lehet használni? Bizonyos feladatoknál nélkülözhetetlen. Például ilyen feladat az is, amikor billentyűzetről begépelt neveket rendezünk abc-sorrendbe, mert itt ismernünk kell a betűknek az abc-beli sorrendjét (ezt a legegyszerűbben úgy tudhatjuk meg, hogy a betűt átalakítjuk ASCII-kóddá, és a szám nagysága határozza meg az abc-beli sorrendjét).

A konverziós függvények közül csak a leggyakrabban használt függvényekről írok, akit érdekel, az utánanézhet a QUICK BASIC című könyvben.

Tehát:

Függvény neve:	Mit konvertál:	Mire:
ASC	betűt	ASCII-kódra
CHR$	ASCII-kódot	betűre
HEX$	hexadecimális számot	decimális számra
INT	LONG,DOUBLE típusú számot	INTEGER típusú számra
VAL	szöveget	számmá
STR$	numerikus értéket	szöveggé

Végül fejezzük be egy könnyű utasítás megtanulásával. Ha programjainkat áttekinthetőbbé akarjuk tenni, akkor a **REM** utasítás segítségével megjegyzéseket írhatunk bele. A megjegyzések semmiben sem befolyásolják a program futását, mert a gép nem is dolgozza fel őket, csak a listában vannak.

Kérdések:

1. Milyen változótípusokat ismersz ?

2. Melyik függvény generál véletlenszámokat ?

3. Mit nevezünk számlálós ciklusnak ?

4. Milyen utasításokkal változtathatjuk meg a program futásának menetét ?

Feladatok:

***1.** Írj egy programot, ami kiszámolja az (2X-3)+5 függvény értékeit 1 és 100 között !*

***2.** Készíts el egy olyan programot, amely ha beírod a neved, kiírja a beadott név kezdőbetűit!*

3. *Írj egy programot, ami egy beírt szövegben összeszámolja azt, hogy milyen betűből hány darab van.(Ha ügyesebb vagy, megpróbálhatod táblázatban vagy grafikonon is megjeleníteni.)*

4. *Írj egy programot, ami egy osztály névsorát abc-sorrendbe rendezi. (az osztály tanulói max. százan lehetnek...)*

5. *Írj egy programot, amivel ki lehet számolni azt, hogy a derékszögű háromszög megszerkeszthető-e. (Ha megszerkeszthető, egyből a területét és a kerületét is kiszámolhatod).*

Az MS DOS 5.0 parancsai

A Parancsok begépelését segítő billentyűkombinációk

ESC: parancs törlése
Ctrl-S vagy PAUSE: parancs vagy program felfüggesztése
Ctrl-C vagy Ctrl-BREAK: parancs megszakítása
F1: vagy jobbranyíl egy karakter másolása az előző parancsból
F2 + karakter: másolás a megadott karakterig
F3: összes hátralévő karakter másolása
F4 + karakter: törlés a megadott karakterig
F5: előző parancs helyettesítése az újonnan beírttal, végrehajtás nélkül

Beállítást végző parancsok

PATH=C:\DOS;C:\NC;C:\WINDOWS: futtatható állományok keresési útjának beállítása
PROMPT füzér: parancs értelmező promptjának beállítása
Füzér:
- $$ = $
- $_ = soremelés
- $b = |
- $d = a rendszerdátum, elöl a hét napjával
- $e = balra mutató kis nyíl, illetve az ESC karakter
- $g = >
- $h = visszatörlés, BackSpace
- $p = az aktuális alkönyvtár útvonala
- $q = =
- $t = a rendszeridő tizedmásodpercekkel
- $v = a futó DOS neve és verziószáma

TIME: kiírja vagy beállítja a rendszeridőt
DATE: kiírja vagy beállítja a rendszerdátumot
VERIFY ON/OFF: háttértárolóra való írás utáni ellenőrzés be/ki kapcsolása

Egyéb parancsok

DIR: egy könyvtár tartalmának kiiratása
- pl.: DIR C:\DOS a C:\DOS direkttori tartalmának kiiratása
- /W öt oszlopba sűrítve listáz
- /P ha betelik a képernyő, egy gombnyomásra vár

CD: aktuális könyvtár megváltoztatása, vagy elérési útjának kiiratása
- CD.. visszalépés az előző könyvtárba
- pl.: CD DOS a DOS alkönyvtárba való belépés

MD könyvtárnév: alkönyvtár létrehozása

RD könyvtárnév: létező alkönyvtár tőrlése
COPY honnan mit hová: file vagy file-ok másolása
pl.: COPY C:\DOS*.COM A:\DOSCOM
DEL mit: file vagy file-ok tőrlése
REN mit mire: ile átnevezése
XCOPY honnan mit hová: file vagy file-ok másolása
/S alkönytárakkal együtt
/P megerősítés kérése minden egyes állománynál
/D:dátum - a dátummal megadott napon vagy utána készült file vagy file-ok másolása
DISKCOPY: teljes lemez másolása
pl.: DISKCOPY A: B: az A: meghajtó tartalmának másolása a B:-re
DISKCOPY A: A: másolás egy lemezegység esetén
FORMAT drive meghajtó formázása
/S system file-ok másolásával együtt
/4 360 kb-os lemez készítése 1.2 Mb-os meghajtó esetén
/Q gyorsformázás
/V:név az új neve a formázott lemeznek
CLS: képernyőtörlés
CHKDSK: lemezállapot ellenőrzése, kiiratása és tárhasználat kijelzése
/F - állományhibák kijavítása az ellenőrzés során
CHKDSK file a megadott file ellenőrzése
UNDELETE: letörölt file vagy file-ok visszahozása
/ALL az összes file visszaállítása megerősítés nélkül
/LIST törölt file-ok listázása visszaállítás nélkül
UNFORMAT: formázott lemez állományainak visszaállítása
/TEST file-ok kiirása helyreállítás nélkül
MEM: tárhasználat ellenőrzése
/P kiírja a memóriában lévő programok nevét
/D kiírja a programok és szoftvermeghajtók nevét, valamint egyéb információkat a memória felosztásáról
/C osztályozza a programokat memória felhasználásuk szerint, felsorolja a méretüket, összegzi a használatban lévő memóriát, és kiírja a rendelkezésre álló legnagyobb memóriaterület adatait
HELP: parancs segítő információt ad a DOS egyes parancsiról
PRINT file: file nyomtatása
/T törli az összes file nevet a nyomtatókra várók sorából
/C a megadott file-t vagy file-okat törli a nyomtatási sorból
SET: kiírja az MS DOS környezetleíró változóit
SETVER: lehetővé teszi, hogy a különböző programok más és más verziójú DOS-ról tudjanak
VER: MS DOS verziószámának kiíratása
SYS: system file-ok átmásolása másik lemezre
pl.: SYS C: A:
A C: meghajtóróról átmásolja a system file-okat az A:-ra.

Norton Utilities 6.00

Alapműveletek

A Norton Utilities (NU) egy segédprogramokból álló programgyűjtemény. A segédprogramokat a DOS-ból, vagy az NU menüjéből tudod aktivizálni. Az NU menüjét a NORTON parancs beírásával tudod elindítani.

Unerase (törlés-helyreállítás)

A letörölt fájlokat az első betűjük nélkül láthatod felsorolva egy ablakban. A visszaállításhoz az Unerase gombra kell kattintanod, majd a fájl első betűjét beírni.

Ha több fájlt akarsz visszaállítani egy könyvtárban, akkor a File menü Group menüpontjára kell rákattintanod. Ezután Unerase, majd Prompt for mising 1st letters, majd egymás után add meg az első betűket.

Az alkönyvtárakat a fájlokhoz hasonlóan tudod helyreállítani.

Ha egy ismeretlen fájlt akarsz visszaállítani, vagy nem tudod, hogy a fájl hol van, akkor a Search menü, for Lost names menüpontját kell választanod, majd kiválasztani a fájlt és az Unerase gomb lenyomása után beírni az első betűt.

Norton Disk Doctor (Lemezdoktor)

Ha a lemez könyvtár és fájlszerkezetében akarsz egy hibát megkeresni, akkor a Diagnose Disk gombot kell lenyomnod, majd a meghajtó(ka)t kell kijelölnöd, és a Surface Test képernyőjén a Cancel gombot lenyomnod.

Ha logikai és fizikai hibákat akarsz felkutatni, akkor az előbbieket kell megismételned, de a Surface Test képernyőjén a Begin Test gombot kell lenyomnod.

Ha a teszteredményt akarod egy fájlba kinyomtatni, akkor a Summary képernyőn a Report gombot kell lenyomnod, de ha nyomtatón akarod látni az eredményt, akkor a Print gombot (persze az sem árt, ha ilyenkor papírt fűzöl a nyomtatóba).

Speed Disk (Töredékmentesítés)

Ha a lemezedet töredékmentesíteni akarod, akkor először a meghajtót kell kijelölnöd, majd az Optimize gombot lenyomnod.

Ha a beállításokat akarod megváltoztatni, akkor a meghajtó kijelölése után a Configure menüpontot kell kiválasztanod.

Ha másfajta optimalizálást akarsz kérni, az Optimize menü, Optimization method pontjában kell azt kiválasztanod.

Ha az írás utáni ellenőrzést akarod bekapcsolni, akkor az Other options menüpont, Read-after-Write pontját kell kiválasztanod.

NDOS (Norton DOS)

A Norton DOS (NDOS) sokkal hatékonyabb és több parancsot kínál mint a DOS COMMAND.COM-ja. Ha az NDOS-t akarod használni, az összes NDOS-sal kezdődő fájlt másold át a főkönyvtárba, a CONFIG.SYS fájlba pedig a

shell=c:\ndos.com /p

sort írd be.

NDOS szolgáltatások

Ha egy fájlhoz szeretnél egy 40 karakterből álló szöveget hozzárendelni, ami a DIR parancsnál kiíródik a képernyőre, akkor a

describe *fájlnév "szöveg"*

sort kell beírnod, ahol a fájlnév a leírandó fájl nevét jelenti, a szöveg a leírást jelenti.

Ha egy fájlt akarsz kilistázni előre-hátra görgethető formában, akkor a

list *fájlnév*

sort kell beírnod, ahol a fájlnév a kilistázandó fájl neve.

Ha egyszerre akarsz könyvtárat és meghajtót változtatni, akkor a

cdd *meghajtó:\könyvtár*

sort kell beírnod, ahol a meghajtó a kívánt meghajtó betűjele, a könyvtár meg a célkönyvtár neve.

Ha a korábbi parancsokat akarod kilistázni, akkor a

history

parancsot kell begépelned.

Ha a menóriaállapotot akarod kiiratni, akkor a

memory

parancsot kell használnod.

Ha az időmérőt akarod bekapcsolni vagy kikapcsolni, akkor a

timer

parancsot kell beírnod.

Az NDOS alatt is használhatod az NU - Batch Enchancer parancsának a legtöbbjét. Ha csak ki akarod próbálni az NDOS-t akkor csak az **NDOS** szót kell beírnod és ha vissza akarsz lépni a DOS-ba, akkor az **Exit** parancsot.

Szövegszerkesztők

A szövegszerkesztők szöveges állományok bevételére, formázására, javítására alkalmasak.

A szövegszerkesztőket három kategóriába soroljuk. A legkisebb tudású szövegszerkesztők a hagyományos szövegszerkesztők. Ezek gyors editálásra alkalmasak; például BAT file-ok írására. Ennek ellenére ezeknek is tudniuk kell szöveget kijelölni és ezekkel műveleteket végrehajtani.

A kijelölt szövegeket blokkoknak hívjuk. Blokkokkal végezhető műveletek: a blokkok másolása, törlése, mozgatása.

A legismertebb hagyományos szövegszerkesztők a Norton Editor, a Personal Editor, a Side Kick, a Dosedit.

A következő kategória a nagy tudású szövegszerkesztők kategóriája. Fő különbség az eddigi szövegszerkesztőkhöz képest a formázási lehetőség. Általában ismer különböző betűtípusokat és -méreteket is, bár ez a nyomtatótól is függ. A bekezdések formázása itt is alapvető követelmény. Be lehet állítani a margókat és a bekezdések igazítását.

A nyomtatás előtt be kell állítani a lapméret margót.

A kiadványszerkesztők nem valódi szövegszerkesztők. Inkább újságok, könyvek szerkesztésére alkalmasak. A rengeteg formázási lehetőség mellett képesek tartalomjegyzék, tárgymutató, fejléc, lábjegyzet, oldalszám készítésére is.

Word

File

Ebben a menüben a file megnyitására, elmentésére és printelésére van lehetőség.

New: Új file létrehozására van lehetőség. Ez a file lehet szöveg- és stílus file.

Open: Egy már régebben elmentett szöveg, stílus vagy egyéb file megnyitására alkalmas. Lehetőség van arra, hogy egy file-t csak olvasásra nyissunk meg.

Save és Save As: A fájl elmentésére e két menüpont szolgál. A különbség a kettő között, hogy míg a Save menünél a már létező nevet nem lehet felülírni, az utóbbinál meg lehet adni a file nevét és útvonalát. Ha nincs neve a szövegnek, akkor a két menü lényegileg ugyanaz.

Close: A fájl lezárására szolgál. A biztonság kedvéért megkérdezi, hogy ne mentse-e el.

Print preview: A nyomtatás előtti megtekintésre szolgál. Célja a formázások ellenőrzése.

Printer setup: A printer beállítását végezhetjük ezzel a menüvel.

Print: Ebben a menüben indíthatjuk el a printelést. Be kell állítani, hogy mely oldalakat és hány példányban akarjuk kinyomtatni.

Quit: A Word program elhagyása.

Edit

Paste: Szöveg beszúrása a zsebből.
Copy: Szöveg másolása zsebbe.
Cut: Kijelölt szöveg törlése; a szöveg a zsebbe kerül.
Undo: Utolsó művelet érvénytelenítése.
Goto: Ugrás megadott szövegrészre.
Search: Szöveg keresése.
Replace: Szöveg cseréje.

Format

Character: A kijelölt szöveg karaktereit formázza át. Megadható a betűk mérete, stílusa és attribútuma.
Paragraph: Bekezdés formázására alkalmas. Megadhatók a margók és a kizárások.
Section: A szekció a dokumentum formai tagoltságát jelenti. Ide tartozik az oszlopszám (columns), a lap- és sorszámozás, valamint a margók.
Margins: Lapméret (Page Size) és margók beállítására alkalmas.

Néhány szó a stíluslapról

A stíluslap egy formai beállításokat tartalmazó file. Ez a formai beállítás vonatkozhat bekezdésre, szekcióra és karakterre.

Minden szöveghez hozzárendelhetünk egy stíluslapot a Format menü Attach Style Sheet segítségével.

Minden stílus rendelkezik egy két karakteres kóddal. Ezzel a kóddal lehetséges a szöveg formázása a Shift+Ctrl+kód billentyűkkel. Ha változtatni akarunk rajta, akkor a Format, Define Styles paranccsal léphetünk ebbe a file-ba. Rá kell menni a módosítandó stílusra, és a Format menü megfelelő pontjával lehet beállítani a kívánt állapotot.

Stílust az Edit, Cut-tal törölhetünk, és a Format, Define Styles paranccsal hozhatunk létre.

Stíluslapot a File, New paranccsal hozhatunk létre. A Style Sheet opciót kell választani a Style keretből.

CorelDraw! 3.00 (kiegészítés a CorelDraw! 2.00-hoz)

Bevezetés

Ebben a kiegészítésben a CorelDraw! 3.00-ról (a továbbiakban CD3) olvashattok. Mivel ez csak kiegészítésként van benne a könyvben, csak az újdonságokról, változásokról lesz benne szó.

Változások, újdonságok

Az egyik újdonság 'csak' egy felirat a menüben: Help. Ez az egyik olyan dolog ami az előző verzió(k)ból hiányzott. Ezzel a funkcióval egy részletes programleírást találhatsz a Windows Help-jének formájában.

A másik újdonság(ok) az ún. úszómenük. Ezek olyan menük, amelyek csak akkor tűnnek el a képernyőről, ha kétszer rákattintasz a bal felső sarkukban lévő Rendszer menüre, vagy kikapcsolod őket a szöveges menüből (vagy kilépsz a CD3-ból). Ezeknek a menüknek a leírását később még megtalálod.

Változott a szövegszerkesztő is: szóelválasztást, szöveg oszlopra való felosztását, bekeretezését tették lehetővé.

Változtattak a látszatablakon is: mostmár az eddigi drótvázas megjelenítés helyett akár a főablakban lehet megszemlélni a rajzot teljes színpompájában.

Beépítették a réteges rajzolási módot is: különböző rétegekre tudsz rajzolni, mint egy-egy fóliára; ezekkel a fóliákat ki tudod nyomtatni, különböző színeket tudsz hozzájuk rendelni, stb.

A rajzos ikonmenük nem változtak sokat, a nagy részük változatlan maradt. Amit megváltoztattak, az a kényelmi szempontokat szolgálja: eddig is használni lehetett néhány ilyen funkciót, csak elő kellett őket halászni a menük mélyéről.

A billentyűparancsok nem változtak, vagy ha igen, akkor csak az új funkciók beépítése miatt, az új parancsokat megtalálhatod az aktuális funkciónál.

Az úszómenük

Az úszómenüket bárhova elvontathatod a képernyőn, ugyanúgy, mint a Windows ablakokat. Ha valamit megváltozatsz az úszómenüben, az a változtatás csak az Apply gomb lenyomása után aktivizálódik. A jobb felső sarokban egy kis kép van: ha a rajta lévő nyíl lefelé mutat, az úszómenüt lenyithatod, ha felfelé, akkor felcsukhatod.

Layers (Rétegek): Ctrl-1

Itt hozhatsz létre új réteget, törölhetsz le egy meglévőt, módosíthatsz valamelyiken. Beállíthatod látható legyen-e, nyomtatható legyen-e, le legyen-e zárva, vagy milyen színű legyen. Ha a MultiLayers feliratot a menüjében kikapcsolod, akkor csak az aktuális réteget láthatod.

Extrude (Kihúzás): Ctrl-E

Ez a funkció egy objektumot 'három dimenzióssá' tesz. Ez a többdimenziósság csak látszat, valójában csak síkban kezelhetők a koordináták is. A tárgyat el lehet forgatni, különböző irányból lehet megvilágítani, árnyékolni, kiszínezni.

Blend (Áttűnés): Ctrl-B

Két tetszőleges tárgyat olvaszt egymásba, úgy, hogy a közöttük lévő színt, alakot, keretszínt is fokozatosan változtatja. Te adhatod meg a lépések számát, az elforgatás mértékét, a követendő görbe útvonalát, a szinezés mértékét.

Fill (Kitöltés)

Ezt az úszómenüt az ikonmenü felső sorában lévő második ikon nyitja ki. A kiválasztott tárgyat lineárisan és radiálisan (vonalasan és körkörösen) töltheted ki. Te általad rajzolt bittérképes mintával is kitöltheted a tárgyadat. Az Update From... gombbal egy másik tárgyról tudod a kitöltést átmásolni.

Pen (Körvonal készítése)

Ezt az úszómenüt is az előbbihez hasonlóan tudod előhozni. Itt a körvonal színét, vastagságát, a vonal végén lévő lezárást tudod beállítani. Itt is az Update From... gombbal tudod átvenni egy másik tárgyról a körvonal tulajdonságait.

Text (Szöveg készítése)

Itt a szöveget a régi módon tudod elkészíteni, azon a két fajtán nem változtattak. Változtattak viszont a kis ClipArt ábrák behozatalán. Ha a szöveg ikonjára kattintasz, előjön egy kis menü, ahol az 'A' betűt választva szöveget, míg a csillagot választva ClipArt ábrát tudsz a képernyőn elhelyezni.

Szövegen be tudod állítani a kiigazítást (jobbra, balra, középre, mindkét oldalra), a betűtípust, a betűk méretét, elhelyezkedését, keretet tehetsz köréjük, oszlopokba tördelheted, bekapcsolhatod az elválasztást.

ASCII táblázat

0:	32:	64: @	96: '	128: Ç	160: á	192: └	224: α
1: ☺	33: !	65: A	97: a	129: ü	161: í	193: ┴	225: ß
2: ☻	34: "	66: B	98: b	130: é	162: ó	194: ┬	226: Γ
3: ♥	35: #	67: C	99: c	131: â	163: ú	195: ├	227: π
4: ♦	36: $	68: D	100: d	132: ä	164: ñ	196: ─	228: Σ
5: ♣	37: %	69: E	101: e	133: à	165: Ñ	197: ┼	229: σ
6: ♠	38: &	70: F	102: f	134: å	166: ª	198: ╞	230: µ
7: •	39: '	71: G	103: g	135: ç	167: º	199: ╟	231: γ
8: ◘	40: (	72: H	104: h	136: ê	168: ¿	200: ╚	232: Φ
9: ○	41:)	73: I	105: i	137: ë	169: ⌐	201: ╔	233: θ
10: ◙	42: *	74: J	106: j	138: è	170: ¬	202: ╩	234: Ω
11: ♂	43: +	75: K	107: k	139: ï	171: ½	203: ╦	235: δ
12: ♀	44: ,	76: L	108: l	140: î	172: ¼	204: ╠	236: ∞
13: ♪	45: -	77: M	109: m	141: ì	173: ¡	205: ═	237: φ
14: ♫	46: .	78: N	110: n	142: Ä	174: «	206: ╬	238: ε
15: ☼	47: /	79: O	111: o	143: Å	175: »	207: ╧	239: ∩
16: ►	48: 0	80: P	112: p	144: É	176: ░	208: ╨	240: ≡
17: ◄	49: 1	81: Q	113: q	145: æ	177: ▒	209: ╤	241: ±
18: ↕	50: 2	82: R	114: r	146: Æ	178: ▓	210: ╥	242: ≥
19: ‼	51: 3	83: S	115: s	147: ô	179: │	211: ╙	243: ≤
20: ¶	52: 4	84: T	116: t	148: ö	180: ┤	212: ╘	244: ⌠
21: §	53: 5	85: U	117: u	149: ò	181: ╡	213: ╒	245: ⌡
22: ▬	54: 6	86: V	118: v	150: û	182: ╢	214: ╓	246: ÷
23: ↨	55: 7	87: W	119: w	151: ù	183: ╖	215: ╫	247: ≈
24: ↑	56: 8	88: X	120: x	152: ÿ	184: ╕	216: ╪	248: °
25: ↓	57: 9	89: Y	121: y	153: Ö	185: ╣	217: ┘	249: ∙
26: →	58: :	90: Z	122: z	154: Ü	186: ║	218: ┌	250: ·
27: ←	59: ;	91: [	123: {	155: ¢	187: ╗	219: █	251: √
28: ∟	60: <	92: \	124: |	156: £	188: ╝	220: ▄	252: ⁿ
29: ↔	61: =	93:]	125: }	157: ¥	189: ╜	221: ▌	253: ²
30: ▲	62: >	94: ^	126: ~	158: ₧	190: ╛	222: ▐	254: ■
31: ▼	63: ?	95: _	127: ⌂	159: ƒ	191: ┐	223: ▀	255:

Felhasznált és javasolt szakirodalomjegyzék:

H.M.Goldstein: A számítógép Pascaltól Neumannig
Számítástechnika középfokon
Zsakó László - Szlávi Péter: Programozási Módszertan
Pirkó József: TURBO PASCAL 5.5
Inotai László - Lázár László: IBM XT/AT rendszerprogramozás
Kelemen Gáspár - Golenczki István: Novell Netware felhasználói ismeretek I.
James Martin - Kathleen K. Chapman: Lokális hálózatok
Bartha Attila: NORTON
Molnár Mátyás: WORD .5.5
Cserhalmi Zsolt: WINDOWS 3.1
Hold Gábor: PAGEMAKER 4.0 for Windows
Dr. Barakonyi Károly: A FRAMEWORK II használata kezdőknek
Bill Harrison: Bevezetés a FRAMEWORK III használatába
Gál István - Dallos Endre: QUICK BASIC

Szószedet

R

S

T

U

V

W

Kiadó és nyomda
TIKETT Nyomdaipari Kft.
9700 Szombathely, Szinyei Merse Pál u. 41.

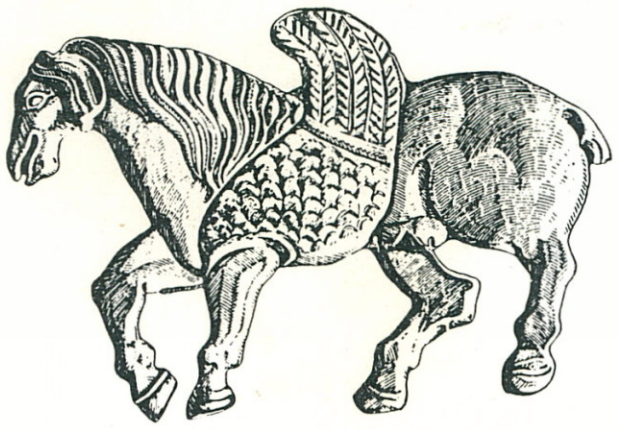
Ára: 235 Ft

KIK VAGYUNK? HONNAN JÖTTÜNK?
AZ ÚJ MAGYAR ŐSTÖRTÉNET VÁZLATA

a címe BAKAY KORNÉL neves régész és őstörténész új könyvének, amely a Karácsonyi Könyvvásár újdonsága lesz.

A szerző az első fejezetben az olvasó elé tárja, hogyan lettünk finnugorok. Nyelvünk rokonsága az ún. finn- ugor népek nyelveivel nem tárgya vizsgálódásainak, ám a magyarság nem az obi-ugorok csoportjából "vált ki", hanem ellenkezőleg, az obi-ugorok és egyéb finn nyelvű csoportok vették át az ősi magyar nyelv rendszerét és szókincsének egy részét.

A második fejezet a hún-magyar rokonság kérdéskörét elemzi, elutasítva a történetírás megkövült közhelyeit, s arra hívja fel a figyelmet, igenis lehetséges a hún- avar-magyar rokonság.



A könyv harmadik fejezetében tesz kísérletet a szerző a könyv főkérdéseinek megválaszolására. Történeti adatok, régészeti emlékek alapján felvázolja a magyarság őstörténetét a Közép-Ázsiai őshazától (az Aral-tó, a Kászpi-tenger, az ókori Oxus [Amu-Darja folyó] melléke) a Kárpát-medencéig.

A források rendkívüli tömegét felsorakoztató mű nemcsak izgalmas, hanem szórakoztatóan olvasmányos és közérthető, noha a kutatók igényeinek is eleget tesz hatalmas jegyzetanyagával és gazdag irodalmával.

A (SZ)ÁMÍTÁSTECHNIKA 1.1 ÉS A KIK VAGYUNK? HONNAN JÖTTÜNK? CÍMŰ KÖNYVEK MEGRENDELHETŐK A

TIKETT KIADÓNÁL

SZOMBATHELY, SZINYEI MERSE PÁL U. 41.

TELEFON/TELEFAX: (06-94) 323-616