

Jegyzet a

Bevezetés az informatikába

című tárgyhoz



Készítette: Nyakóné Juhász Katalin

Lektorálta:

Tartalomjegyzék

Számrendszerek	5
A számolás története, a számrendszerek kialakulása	5
A számítástechnikában használatos számrendszerek	10
Aritmetikai műveletek különböző számrendszerekben	13
A számítógép mint adatfeldolgozó eszköz	14
Történeti áttekintés	14
Számítógép generációk	18
Első generációs számítógépek	18
Második generációs számítógépek	20
Harmadik generációs számítógépek	20
Negyedik generációs számítógépek	20
Ötödik generációs számítógépek	21
Adatábrázolás a számítógépen	22
Számábrázolás	23
Fixpontos számábrázolás	23
Lebegőpontos számábrázolás	24
Kódolt ábrázolás	27
Binárisan kódolt decimális (Binary Coded Decimal - BCD)	27
Nem-numerikus karakterek, kód táblázatok	27
Műveletek a számítógépen	30
A számítógép felépítése	32
A memóriák	36
Perifériák	38
I/O vezérlő	38
Párhuzamos és soros adatátvitel	38
FireWire	39
USB	39
Háttértárolók	40
A mágneses háttértárolók	40
Az optikai háttértárolók	41
RAID	41
Pendrive	41
Bemeneti perifériák	42
Billentyűzet	42
Egér	43
Egyéb beviteli eszközök	44
Kimeneti perifériák	44
Monitor, videokártya	44
Nyomtatók	46

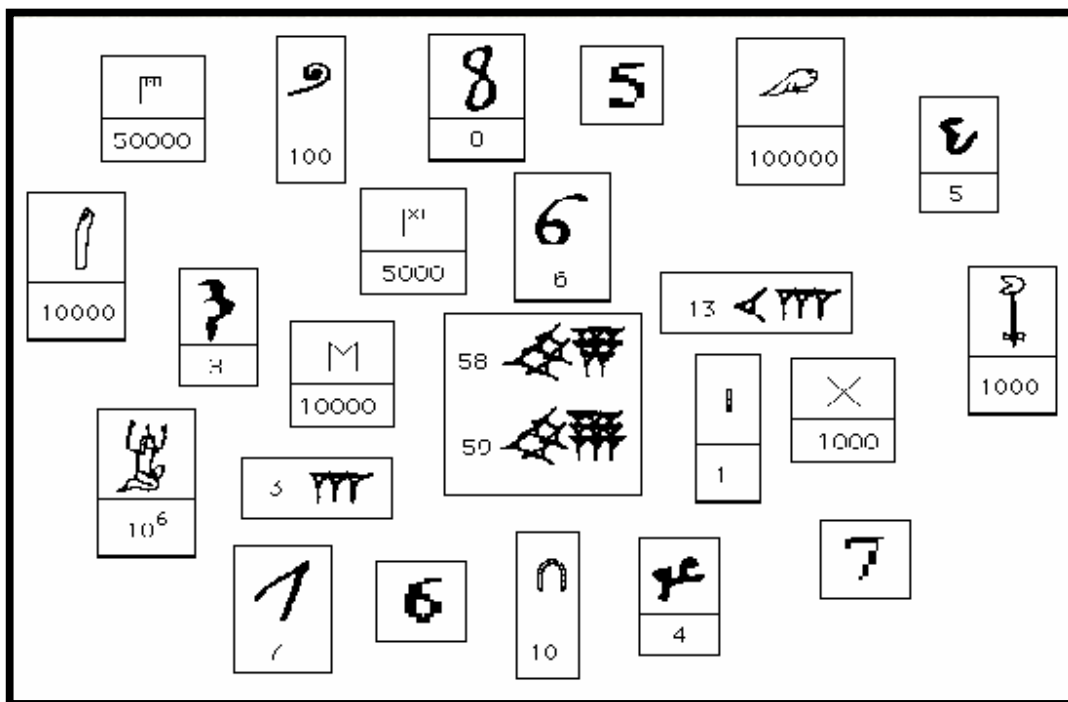
Egyéb kiviteli eszközök	48
Plotter	48
Hangszóró	48
Számítógéphálózatok	49
Sín vagy busz topológia	49
Csillag topológia	50
Gyűrű topológia	51
Fa topológia	52
Vezetékes adatátviteli közegek	55
Csavart érpár	55
Koaxiális kábelek	55
Üvegszál kábel	55
Vezeték nélküli adatátviteli közegek	56
Infravörös, lézer átvitel	56
Rádióhullám	56
Szórt spektrumú sugárzás	56
Műhold	56
Adatátviteli vezérlő egységek	57
Repeater	57
Bridge	57
Router	57
Gateway	57
Az internet	58
Az internet története	58
Csomagkapcsolt átvitel	59
RFC	59
TCP/IP	60
Fontosabb szolgáltatások	60
A szoftver	63
A szoftverek csoportosítása	63
Rendszer- és rendszerközel szoftverek	63
Felhasználói vagy alkalmazói szoftverek	64
Szoftverek osztályozása kereskedelmi szempontból	65
Az operációs rendszer	66
A megszakítás (interrupt) kezelése	66
Spooling technika	68
Perifériák ütemezése	69
Tárkezelési problémák	69
Processzor időbeosztás	70
Személyi számítógépek operációs rendszerei	71
Parancsvezérelt (DOS alapú) operációs rendszerek	71
Grafikus felületű operációs rendszerek	71
UNIX rendszerek	72
Algoritmusok	74
Nevezetes algoritmusok	78
Kereső algoritmusok	78
Rendező algoritmusok	79

<i>A programozás alapjai.....</i>	<i>82</i>
<i>Alkalmazások.....</i>	<i>91</i>
<i>Szövegszerkesztés.....</i>	<i>92</i>
<i>Tömörítés</i>	<i>97</i>
<i>Vírusok.....</i>	<i>97</i>
<i>Ajánlott irodalom.....</i>	<i>98</i>

Számrendszerek

A számolás története, a számrendszerek kialakulása








Ma Magyarországon arab számokat használunk, és tízes számrendszerben számolunk. De a történelem ennél többféle számrendszert és különböző írásmódokat ismer. Ezekből ad ízelítőt az alábbi ábra:

















Az ősember az ujjait használta a számoláshoz. (Az *ujj* latin neve *digitus*, innen származik a számjegy angol *digit* neve.) Nagyobb számok kezelésére az ókorban köveket használtak. (A *kövecske* latin neve *calculus*, innen származik a mai *kalkulátor* szó.)

Egyiptomban az i.e. 2000 körüli időkben már jól kialakult tízes számrendszer volt, melynek elterjedését a mezőgazdaság és a csillagászat szükségletei mozdították elő. Minden magasabb tízes egységre külön jelet használtak, tehát a helyi érték fogalmát még nem ismerték. Egy pálcika: 1; két pálcika: 2; három pálcika: 3; és így tovább. Egy hajtű: 10; két hajtű: 20; három hajtű: 30; és így tovább. Egy csavar: 100; két csavar: 200; három csavar: 300; és így tovább. Egy




























































lótuszvirág: 1000; két lótuszvirág: 2000, és így tovább, millióig. A milliót a csodálkozó, térdeplő emberalak fejezte ki. Ezek a jelek láthatók az alábbi ábrán:

						
1	10	100	1000	10000	100000	10 ⁶

													
1	2	3	4	5	6	7	8	9	10	100	1000	10000	100000

Mezopotámiában, a késői sumér korszakban, ahol a csatornázás és építkezés bonyolult számítást kívánt, fejlett helyi értékes hatvanas számrendszert találunk, amely még árulkodik az előző tízes számrendszer használatáról, hiszen 1-től 60-ig a régebbi tízes számrendszer segítségével írták le a számjegyeket. Ékírásos jeleiket az agyagba nyomták, az agyagtáblát tüzes kemencében kiégették, s ezzel olyan időtállóvá tették azt, hogy csak meg kellett találni a táblát, megfejteni az ékírás titkát, és az az idők végezetéig olvasható marad. Egy agyagtábla képét és a számok ékírásos megfelelőit láthatjuk az alábbi ábrákon:



1 	11 	21 	31 	41 	51 
2 	12 	22 	32 	42 	52 
3 	13 	23 	33 	43 	53 
4 	14 	24 	34 	44 	54 
5 	15 	25 	35 	45 	55 
6 	16 	26 	36 	46 	56 
7 	17 	27 	37 	47 	57 
8 	18 	28 	38 	48 	58 
9 	19 	29 	39 	49 	59 
10 	20 	30 	40 	50 	

A számolásra utaló legrégebb kínai jelek az i.e. XIV. - XI. századból származó – jósláshoz használt – csontokon, valamint az i.e. X. - III. századi cserép- vagy bronztárgyakon és pénzekon

maradtak fenn, és nem helyi értékes, de tízes számrendszerről tanúskodnak. A számpálcikás számrendszer a tízes alapú helyi értékes rendszerek legrégebbike, azonban az ebből kialakult írásbeli rendszert nem egészítették ki a nulla jelével, ezért a számítások többségét még a papír feltalálása után is számológéptáblán végezték. Az idők folyamán többször átdolgozott és kibővített kínai matematikai értekezés, a *Matematika kilenc könyvben* VIII. könyvében a tudomány történetében először találkozunk a pozitív és negatív számok megkülönböztetésével, és itt fogalmazták meg a negatív számokkal végzett műveletek legegyszerűbb szabályait is. A táblázat pozitív elemeit piros pálcikákkal ábrázolták, a negatívokat feketével. Az ábrázolás ilyen módját a könyvnyomtatásban is alkalmazták.

Régi számábrázolási formákat láthatunk az alábbi ábrán:

A számok Sang-Jin-kori alakja:

Modern alak:

Indiai-arab számmal:

一	=	三	三	五	八、八	+	九	九
一	二	三	四	五	六	七	八	九
1	2	3	4	5	6	7	8	9

Az ókori görögök számírása az i.e. V. század tájékán nem helyi értékes tízes számrendszerben történt. Az első 9 számjegyet ábécéjük első 9 betűje, a 9 darab tízest a következő 9 betű, és a 9 százast a további 9 betű jelentette. A 999-nél nagyobb számok leírására betű-számjegyeik mellett külön jeleket használtak. Ilyen alfabetikus számjegyírást találunk az ószláv, a héber és az arab népeknél is:

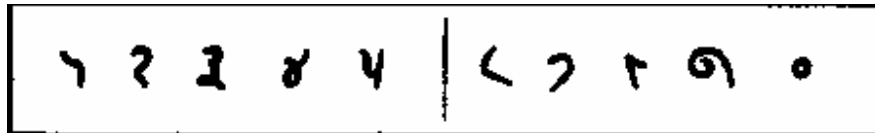
1	2	3	4	5	6	7	8	9	10	20	30	40	...	100	...	500	...	900	...	1000
α	β	γ	δ	ε	ς	ζ	η	θ	ι	κ	λ	μ		ρ		φ		χ		,α

Tízes számrendszerre használatára utalnak a római számjegyek is. A tízes számrendszerre mutató 1, 10, 100 és 1000 jeleket kibővítették az 5, 50 és 500 jelével, így az általuk használt számjegyek:

$$I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000,$$

amelyekből a többi számot a következő módon tudták előállítani: a számok írásánál az egymás mellé írt egyenlő jegyeket össze kell adni; az egymás mellé írt különböző jegyeknél a kisebb számot a nagyobbhoz kell adni, ha ettől jobbra áll, és levonni belőle, ha balra áll tőle. Például: IX=10-1=9, XI=10+1=11. Mivel a rómaiak nem ismerték a számok helyi értékét, a nagy számok leírása kényelmetlen, és a számolás reménytelenül nehéz volt.

Az indiai népek legnagyobb tudományos és általános kultúrtörténeti vívmánya a helyi érték elvén alapuló tízes számrendszer megteremtése volt, amelynek kiteljesedése hosszú időt vett igénybe, és még távolról sem ismert fejlődésének minden állomása. Az indiai számolási mód már ősidők óta tízes alapú volt, bár egyes időszakokban és egyes vidékeken a négyes alapszám nyomai is megtalálhatók. A mi arab számjegyeket használó helyi értékes tízes vagy dekadikus számrendszerünk arab közvetítéssel, de Indiából származik. A régi tízes számrendszer és a helyi érték használata itt forrt össze, valószínűleg a III. - IV. században. Brahmagupta-hoz fűződik a kis körrel jelölt 0 feltalálása és használata a számok írásmódjában (lásd a következő ábra), valamint a negatív számokkal végzett műveletek kiterjesztése is az ő műveiben szerepel először.



Az ősmagyarok a történelmi időkben már tízes számrendszert használtak. Ez azonban az előző idők hatos és hetes számrendszerén át, hosszú fejlődés eredménye volt. A hetes számrendszerre lehet következtetni például a mesék hétfejű sárkányáról, a hetedhét országról, a hét rőf hosszú szakállról, a hétmérföldes csizmáról, a hétpecsés titokról. A később keletkezett nyolc és kilenc számneveinkben a szóvégi c, régiesen írva z, valószínűleg a tíz számnév végződése. Ebből úgy sejtjük, hogy a nyolcat és a kilencet a tízből származtatták ők. A nyelvészeti kutatások szerint a finnugor nyelvekben közös gyökere van a két, három, négy, öt, hat és száz tőszámneveknek. Ezek kialakulásakor még a finnugor népek együtt voltak és hatos számrendszert használtak. A hét számnév már a szűkebb ugor családra utal (magyar, vogul, osztják). E népek nyelvében a hét szó nemcsak számnév, hanem jelenti a hétnapos időtartamot is. Az ősi rovásírás számjegyei és azok írásmódja (lásd a következő ábra) már a tízes számrendszerre utalnak. Nemcsak a számok alakját, hanem elnevezését is az indiaiaktól vették át. Később az arab számok alakja folyamatosan változott.

•	1	2	3	4	5	6	7	8	9
0	1	2	3	4	5	6	7	8	9

Egy számrendszer (vagy számábrázolási rendszer) egységes szabályok alapján határozza meg, hogy a számjegyek sorozata milyen számokat jelenít meg. Például, a tízes számrendszer azt

jelenti, hogy egy adott mennyiség kifejezésekor a csoportosítást 10-esével végezzük. Először tízes csoportokat hozunk létre az adott mennyiségből, majd az így kapott csoportokat is tízesével csoportosítjuk mindaddig, amíg újabb nagyobb csoport létrehozható. Ebből elsősorban az következik, hogy 10 különböző jelet kell használnunk a számok leírására: a 0-t a „nem maradt” kifejezésére, az 1, 2, 3, 4, 5, 6, 7, 8, 9 jeleket pedig a csoportosításból kimaradt 9 lehetséges állapot jelzésére. A lényeges újítás abban állt, hogy a számjegyek a helyüktől függően más értéket vesznek fel. Az első csoportosítás végén megmaradtak száma kerül a szám jobb szélére, tőle balra a második csoportosítás végén megmaradtak száma, és így tovább.

A számrendszerek lényegét a helyi érték fogalma alapján lehet megérteni. A szám értékét úgy kapjuk, hogy az egyes számjegyek értékét szorozzuk a helyi értékükkel, és mindezt összeadjuk. Tízes számrendszerben:

$$123,45 = 1 \cdot 10^2 + 2 \cdot 10^1 + 3 \cdot 10^0 + 4 \cdot 10^{-1} + 5 \cdot 10^{-2}.$$

Általában egy

$$a_n a_{n-1} \dots a_1 a_0 a_{-1} a_{-2} \dots a_{-m}$$

alakú szám felírása polinom alakban tízes számrendszerben

$$\sum_{i=-m}^n a_i \cdot 10^i.$$

Tetszőleges p alapú ($p > 1$) számrendszerben a használt számjegyek

$$0, 1, \dots, p-1,$$

a helyi értékek pedig a p szám hatványai:

$$\sum_{i=-m}^n a_i \cdot p^i$$

A számítástechnikában használatos számrendszerek

A számítástechnikában leggyakrabban a tízes (decimális), a kettes (bináris) és a tizenhatos (hexadecimális) számrendszerrel dolgozunk. A kettes számrendszerben csak kétféle jelet használunk (0,1), míg a tizenhatosban 16 különbözőt, ezért betűkre is szükség van (0, ..., 9, A, B, ..., F).

A tízes számrendszer alkalmazása a más területen való mindennapi használatból nyilvánvaló. A kettes számrendszer használata a digitális számítógép tulajdonságaiból adódik. A tizenhatos számrendszer pedig a tömörebb írásmódot teszi lehetővé, hiszen szoros kapcsolatban van a kettes számrendszerrel. A tizenhatos számrendszer számjegyei négy kettes számrendszerbeli számjeggyel írhatók fel, ugyanis $2^4 = 16$.

Kettes számrendszer	Tizenhatos számrendszer	Tízes számrendszer
0000	0	0
0001	1	1
0010	2	2
0011	3	3
0100	4	4
0101	5	5
0110	6	6
0111	7	7
1000	8	8
1001	9	9
1010	A	10
1011	B	11
1100	C	12
1101	D	13
1110	E	14
1111	F	15

A számokat különböző számrendszerekben írhatjuk fel. A felírásnál konverziót alkalmazunk. Alábbi példáinkban csak a számítástechnikában használatos számrendszerekre szorítkozunk.

Példák

1. Mi lesz $123,45_{10}$ kettes, illetve tizenhatos számrendszerbeli alakja?

Megoldás:

A $123,45_{10}$ konverziója kettes, illetve tizenhatos számrendszerbe az alábbiak szerint történik:

Egészrész				Törtrész			
/2		/16		*2		*16	
123	1	123	B	0,45	0	0,45	7
61	1	7	7	0,9	1	0,2	3
30	0	0		0,8	1	0,2	3
15	1			0,6	1	...	
7	1			0,2	0		
3	1			0,4	0		
1	1			0,8	1		
0				...			

Tehát

$$123,45_{10} \rightarrow 1111011.0111001..._2 \text{ és } 123,45_{10} \rightarrow 7B.733..._{16}.$$

A kapott értékeket kettes, illetve tizenhatos számrendszerből tízesbe az alábbiak szerint írjuk vissza:

$$1111011.0111001..._2 \longrightarrow$$

$$2^6 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0 + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-7} = 64 + 32 + 16 + 8 + 2 + 1 + 1/4 + 1/8 + 1/16 = 123,4453125 \sim 123,45_{10}$$

$$7B.733..._{16} \longrightarrow$$

$$7 \cdot 16^1 + B \cdot 16^0 + 7 \cdot 16^{-1} + 3 \cdot 16^{-2} + 3 \cdot 16^{-3} = 112 + 11 + 1/16 + 1/256 + 1/4096 =$$

$$123,449951171875 \sim 123,45_{10}$$

2. Mi lesz a tizenhatos számrendszerbeli alakja az 110100111111010_2 számnak?

Megoldás:

A kettes számrendszerbeli szám számjegyeit balról jobbra haladva négyes csoportokra osztjuk, és megadjuk a csoportokhoz tartozó tizenhatos számrendszerbeli számokat:

$$1101,0011,1111,1010_2 \longrightarrow D3FA_{16}$$

3. Mi lesz a kettes számrendszerbeli alakja az $AC95_{16}$ számnak?

Megoldás:

Megadjuk a tizenhatos számrendszerbeli szám számjegyeinek megfelelő kettes számrendszerbelieket:

$AC95_{16} \longrightarrow 1010110010010101_2$

Feladatok

1. Konvertáljuk tízes számrendszerbe az alábbi számokat:

1011.01_2 ; 123.45_{16} ; $1A9.DB_{16}$.

2. Konvertáljuk kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat:

$BABA_{16}$; $ABBA_{16}$; $DADA_{16}$; $ECCE_{16}$;

101101110011_2 ; 1110111100010111_2 .

3. Konvertáljuk bináris számrendszerbe az alábbi decimális számokat:

$3492,326$; 1000 ; $1512,1533$; $112,3$.

4. Konvertáljuk hexadecimális számrendszerbe az alábbi decimális számokat:

$12438,964$; $3096,123$; $12345,678$; 9977 .

Aritmetikai műveletek különböző számrendszerekben

Az aritmetikai műveleteket a tízes számrendszerben megszokott módon végezzük minden más számrendszerben.

Példák

1. Végezzük el az alábbi műveleteket a bináris számok körében:

$$1001.01 + 1001.10; 1001.11 - 1001.10.$$

Megoldás:

$$\begin{array}{r} 1001.01 \\ +1001.10 \\ \hline 10010.11 \end{array}$$

$$\begin{array}{r} 1001.11 \\ -1001.10 \\ \hline 0.01 \end{array}$$

2. Végezzük el az alábbi műveleteket a hexadecimális számok körében:

Megoldás:

$$\begin{array}{r} ABCD.EF \\ +1923.7A \\ \hline C4F1.69 \end{array}$$

$$\begin{array}{r} 1AB2C.23 \\ -AB3C.25 \\ \hline FFEF.FE \end{array}$$

Feladatok

1. Végezzük el az alábbi műveleteket a bináris számok körében:

$$10111.01 + 1111.11;$$

$$100010.111 + 101110.111;$$

$$1000.11 - 111.00;$$

$$10000.1110 - 1001.1111.$$

2. Végezzük el az alábbi műveleteket a hexadecimális számok körében:

$$CCC.CC + DDD.DD;$$

$$1000.010 + A111.013;$$

$$AAA.AA - AA.AB;$$

$$10000.100 - 1111.111.$$

A számítógép mint adatfeldolgozó eszköz

Történeti áttekintés

Az ember mindig arra törekedett, hogy életét technikai segédeszközökkel megkönnyítse. Így volt ez a számlálás és a számolás esetében is. A nehézkes számrendszerük miatt a rómaiak használtak először számolólécet. Később jelentek meg a *saun-pan*, *soroban*, *scso ti*, stb. Az abakusz pedig a számítógép őseinek tekinthető.

Európában még a középkorban is számolóléccel számoltak. *Adam Riese* (1492-1559) német matematikus fejlesztette ki a számolóléce vonalain való számolást, a vonalak közötti számolás helyett. Ő fedezte fel, hogy a negatív hatványok segítségével tíznek a törtrészei is képezhetők.

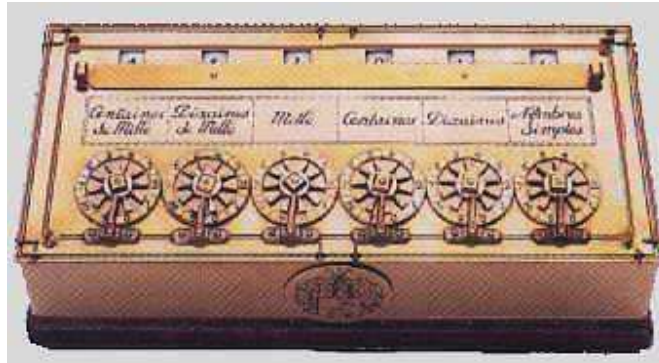
Talán az első igazi újkori matematikai felfedezés, amely a görögök és az arabok számára elképzelhetetlen lett volna, a logaritmus felfedezése volt. Egyszerre két tudós is felfedezte, egymástól függetlenül: *John Napier* (1550–1617) skót báró és *Jobst Bürgi* (1552-1623) svájci polgár. 1588-ban Jost Bürgi elkészítette az első logaritmustáblákat.

A logaritmus felfedezésének köszönhető mechanikus segédeszköz a logarléc. 1622-ben *William Oughtred* (1574-1664) alkalmazott először logaritmus skálát a két, egymáson elcsúsztatható vonalzókon. 1650-ben készítette *Patridge* az első mai formájú logarlécet. 1851-ben vezették be a csúszóablakot, amelynek segítségével több skálát is lehetett egyszerre használni.

1623-ban *Wilhelm Schickard* (1592-1635), tübingeni professzor egyszerű, négyalpműveletes masinát szerkesztett. A gép működésének elve a John Napier által készített Napier-csontok számolási eljárásait követi. A gép (lásd a következő ábrát) számtárcsákkal tárolja a részeredményeket, és a túlcsoordulást egy kis csengő megszólaltatásával jelzi.



Blaise Pascal (1623-1662) 1642-ben összeadó-kivonó gépet (lásd a következő ábrát) készített. A kivonáshoz komplementst kellett képezni.

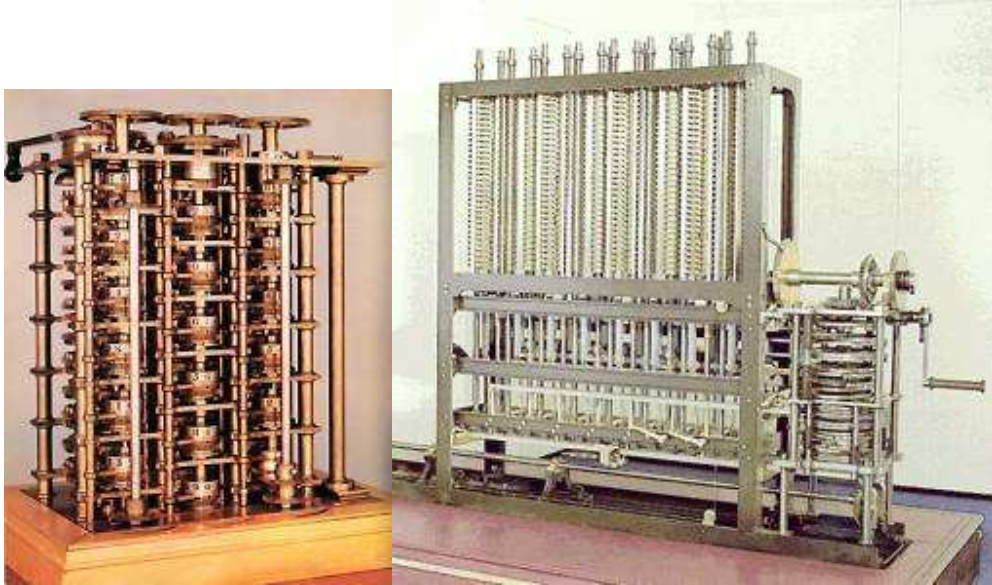


Gottfried Wilhelm Leibnitz (1646-1716) német tudós 1672-ben mechanikus számológépet (lásd a következő ábrát) épített. Az első, valódi négyalpműveletes gépet alkotta meg kézi forgató meghajtással, mozgatható beállító művel. Leibnitz nevéhez még két felfedezés fűződik, melynek nagy szerepe van a számítások korszerűsítésében: 1666-ban bebizonyítja, hogy egy számolási művelet egymás után elvégezhető egyszerű lépések sorozatára bontható; 1679-ben pedig ismerteti a kettes számrendszert.

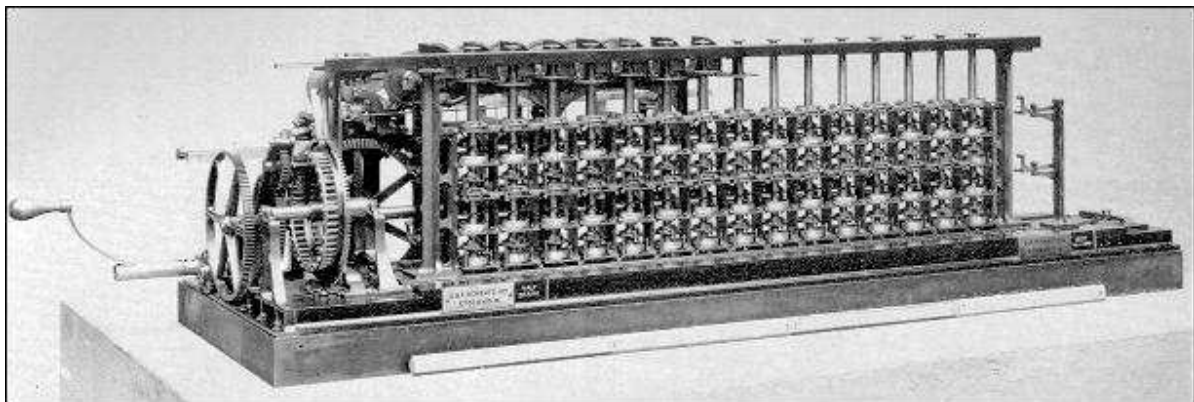


Az olasz *Giovanni Polenus* és *Antonius Braun* a bordáskerekes (mozgatható fogú fogaskerék) gép feltalálói. Ennek segítségével 1774 és 1790 között készített számítógépet *Philipp Matthäus Hahn* plébános.

1820-as évek elején *Charles Babbage* (1782-1871) megtervezte a Difference Engine-t (differenciagépet), amely logaritmus táblázatok pontos és gyors elkészítését tette lehetővé. A bal oldali képen a Differencial Engine számológépének 1832-ben összeszerelt részlete látható, a jobb oldalin pedig a londoni Science Museum-ban látható replika, mely Babbage eredeti tervei szerint épült:



Az első működő gépet azonban csak 1853-ban *Pehr Georg Scheutz* (1785 - 1873) svéd nyomdász és fia, *Edvard Scheutz* készítette el (lásd a következő ábrát), mert Babbage a szükséges közel 50000 alkatrészt nem tudta legyártatni. A differenciagépet egészen 1940-ig használták matematikai táblázatok elkészítéséhez.



1833-ban Babbage megtervezte az Analytical Engine-t (analitikus gépet), ami a történelem első számoló automatája lett volna. 1847-ig ezen a gépen dolgozott, bár az építése már kezdetben megakadt: a kor finommechanikai lehetőségeivel ezt a gépet nem lehetett elkészíteni.

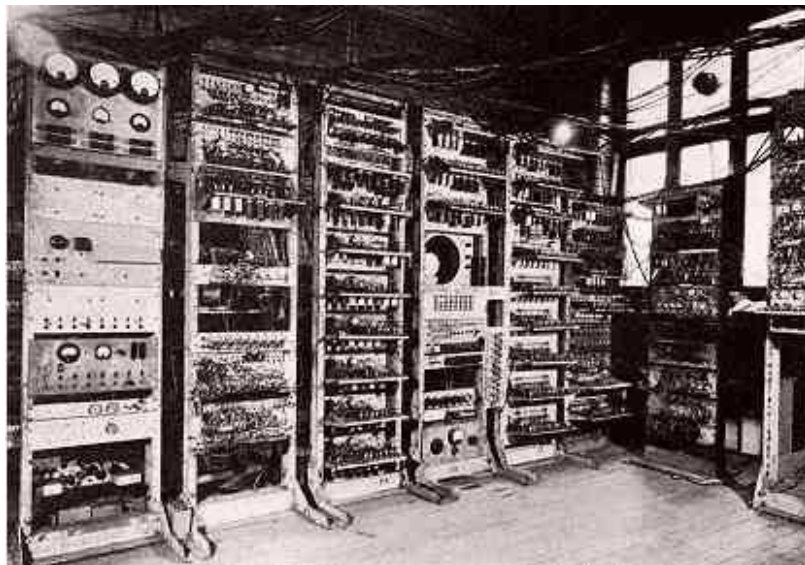
Augusta Ada King (született Lovelace Byron) grófnő (1815–1852) főként arról ismert, hogy leírást készített a Charles Babbage által tervezett Analytical Engine-hez.

1850-ben szabadalmaztatták az első billentyűs vezérlésű összeadó gépet. 1885-ben *Stevens Borroughs* (1857-1898) elkészítette az első billentyűzettel, nyomtatóval ellátott összeadó gépet.

Konrad Zuse (1910-1985) német építészmérnök 1938-ban elkészített Z1 nevű elektromechanikusnak mondható gépe már kettes számrendszerben számolt, és egy úgynevezett fénymátrixon (szintén kettes számrendszerben) jelenítette meg az eredményeket. A Z1 24 bites szavakkal dolgozott, memóriájában 16 adatot tudott tárolni, és decimális-bináris átalakítót is tartalmazott.

Konrad Zuse készülékeit Z2 (16 bites fixpontos adatokkal dolgozott és 16 szavas tárolója volt) és Z3 néven fejlesztette tovább. Az 1941-ben elkészült Z3 egy jelfogókból felépített gép, amely lebegőpontos aritmetikai egységgel, program- és adattárolási lehetőséggel rendelkezett.

Az első teljesen automatikusan működő számítógépet az Egyesült Államokban, a Harvard Egyetemen készítették el *Howard H. Aiken* (1900-1973), az egyetem professzora vezetésével, és 1944-ben az egyetemnek adományozták a Harvard Mark I nevű elektromechanikus gépet (lásd a következő ábrát), amely Babbage elvei alapján épült.



Számítógép generációk

Az elektronikus számítógépeket felépítési elvük, az alkalmazott logikai elemek működési elve, illetve az alkalmazott áramkörök integráltsági foka alapján **generációkba** soroljuk. A generációkhoz tartozó időintervallumokat csak hozzávetőlegesen lehet meghatározni, ezért a szakirodalomban többféleképpen adják meg ezeket. Az alábbiakban egy lehetséges besorolást ismertetünk.

Első generációs számítógépek

A számítógépek **első generációi** az elektroncsöves digitális gépek. Kialakulásukat az tette lehetővé, hogy *Lee de Forest* (1873-1961) 1906-ban feltalálta az elektroncsövet. Az első generáció időszaka 1940 és 1954 közé tehető. A háború és a háborús kutatások nagy lendületet adtak a számítógépipar fejlődésének.

1939-ben az Egyesült Államokban az Iowa State College-ban *John Atanasoff* (1903-1995) és *Cliffor Berry* (1918-1963) megépítették egy elektronikus gép prototípusát. Az építők nevének kezdőbetűiből a számítógép az ABC (Atanasoff-Berry Computer) nevet kapta.

1943 decemberére a britek elkészítették a Colossus nevű számítógép első, 1944-ben pedig a második verzióját, melyek a németek kódoló gépén elküldött üzenetek megfejtésére szolgáltak a II. világháború alatt.

1946-ban fejezték be az ENIAC (Electronic Numerical Integrator and Computer) építését az amerikai Pennsylvanai Egyetemen. *John William Mauchly* (1907–1980) vezetésével végezték a fejlesztést, amiben részt vett *John Presper Eckert* (1919-1995) az egyetem, valamint *Hermann Heine Goldstine* (1913-2004) a hadsereg részéről. Az ENIAC-ot ballisztikai és szélcsatorna-számításokra használták, és 1955-ig működött sikeresen.



A korszak egyik legjelentősebb tudósa az alábbi képen látható **Neumann János** (Budapest, 1903. december 28. – Washington D. C., 1957. február 8.) magyar származású matematikus volt, aki több tudományterületen is kimagasló eredményeket ért el.



Neumann és Goldstine személyesen először 1944-ben találkozott. 1948-ban megfogalmazták az elektronikus digitális számítógépekkel, az úgynevezett Neumann-elvű gépekkel szembeni követelményeket.

A Neumann-elv:

- A számítógép legyen soros működésű, teljesen elektronikus. A gép egyszerre csak egy műveletet vesz figyelembe és hajt végre, és mindezt igen gyorsan.
- A gép a bináris számrendszert használja.
- Az adatok és a programok a gép belső tárolójában helyezkedjenek el.
- A vezérlőegység emberi beavatkozás nélkül értelmezze és hajtja végre az utasításokat.

- A számítógép tartalmazzon egy olyan egységet, ami képes elvégezni az alapvető logikai műveleteket.

Második generációs számítógépek

A **második generációs** számítógépek építésének időszaka az 1955 és 1965 közötti évek. Tranzisztorokat, ferritgyűrűs tárokat tartalmaztak. Az előzménye az volt, hogy *Walter Houser Brattain* (1902-1987), *John Bardeen* (1908-1991) és *William Bradford Shockley* (1910-1989) amerikai fizikusok feltalálták a tranzisztort (1948). Ebben az időben jelent meg az operációs rendszer őseinek tekinthető MONITOR. Ez egy, a memóriában tárolódó program volt, amely a számítógépet vezérelte, az operátor csak a perifériákat kezelte. Megjelentek az első programnyelvek is (1954 - FORTRAN, 1958 - ALGOL, 1959 - COBOL, 1964 – *Thomas E. Kurtz* (1928-) és **Kemény János** (1926-1992) megalkották a BASIC (Beginner's All-purpose Symbolic Instruction Code) nyelvet).

Harmadik generációs számítógépek

A **harmadik generációs** számítógépek már integrált áramköröket használtak. Az integrált áramkör feltalálását 1959-ben jelentették be. Kialakult a multiprogramozás és a párhuzamos működtetés, melynek segítségével lehetőség nyílt egy számítógépet egy időben több feladatra is használni. A harmadik generáció korszakát az 1965-1974-es évekre lehet tenni. Erre az időszakra az SSI, MSI (Small & Medium Scale Integration) áramkörök használata volt jellemző.

Negyedik generációs számítógépek

A számítógépek **negyedik generációját** az 1970-es évek elejétől napjainkig számíthatjuk. (Vannak, akik az 1990-es évek elejére teszik e korszak végét, és a miniatürizálást már új korszaknak tekintik.) A gépek igen nagy integráltságú (LSI, VLSI – Very Large Scale Integration) áramkörökből épülnek fel. Nincsenek alapvető változások a számítógépek szervezésében. A korábban bevett megoldásokat tökéletesítik. A negyedik generáció jellemzője, hogy a szoftvergyártás óriási méretűvé válik. A szoftverek árai elérik, egyes esetekben meg is haladhatják a hardverét.

Akik a számítástechnika, informatika iránt valamilyen formában érdeklődnek, tudják, hogy mennyi feltáratlan területe van még ennek a tudománynak. A kutatásokban a jövő felé vezető út a mesterséges intelligenciához kapcsolódik.

Ötödik generációs számítógépek

Az **ötödik generációra** való előrejelzések elég sok bizonytalanságot hordoznak, mert ezek a változások épp csak megkezdődtek. 1981-ben, egy Japánban tartott konferencián új állami kutatási tervet jelentettek be, aminek a célja egy ilyen számítógép elveinek lerakása volt, melynek fontos alkotórésze a mesterséges intelligencia, a szakértői rendszerek, a szimbólumokkal való műveletvégzés. A cél tehát olyan intelligens számítógép létrehozása, mely lát, hall, beszél és gondolkodik. A számítógép felépítése is változni fog: a többprocesszoros, párhuzamos, elosztott (grid) adatfeldolgozású gépek veszik át lassan a Neumann-típusú gépek szerepét.

Adatábrázolás a számítógépen

Az **adat** az objektumok mérhető és nem mérhető tulajdonsága, vagy - ahogyan az értelmező szótár definiálja - valakinek vagy valaminek a megismeréséhez, jellemzéséhez hozzásegítő (nyilvántartott) tény, részlet. Az adatnak önmagában nincs sem jelentése, sem bármilyen szövegösszefüggése. Tengernyi adat születik minden egyes intézményben, és az adatok nyilvántartása, feldolgozása, továbbítása igen sokféle eszközt igényel.

Az **információ** az értelmezett adat, amelynek legfontosabb jellemzője, hogy bizonytalanságot, határozatlanságot oszlat el. Az adatból akkor lesz információ, ha valamilyen jelentést kap, s annak alapján valamiféle ítélet alkotható. Információnak nevezünk mindent, amit a rendelkezésünkre álló adatokból nyerünk. Az információ olyan tény, amelynek megismerésekor olyan tudásra teszünk szert, ami addig nem volt a birtokunkban.

A számítástechnikában az információ legkisebb egysége a *bit* - binary digit. A programok is 1 bites információkból épülnek fel. A bit lehet *0* vagy *1*, *hamis* vagy *igaz*; azaz bármely kettő, egymást kölcsönösen kizáró *állapot*.

A bit ugyanakkor az információt hordozó közlemény hosszának egyik alapegysége is. Egy hírforrás valamely p valószínűséggel (relatív gyakorisággal) kibocsátott h hírének az információtartalma: $I(h) = -\log_2 p$ bit. Egy eldöntendő kérdésre adott válasz információtartalma 1 bit, ha mindkét válasz egyformán valószínű.

A **byte** bitek csoportja, leggyakrabban 8 bit. A számítógépi adattárolás legkisebb, címezhető eleme, illetve a tárolókapacitás mértékegysége.

A számítógép az adatokat kódolt formában tárolja, kezeli és képezi. A számítógépnek tudnia kell, hogy az adott adat az éppen szám, szöveg, utasítás vagy valami más. A következőkben ehhez meg kell ismerni a különféle adatok tárolási módját, vagyis a belső adatábrázolást.

Számábrázolás

Ha az adatokkal aritmetikai műveleteket akarunk végezni, akkor azok reprezentálására fixpontos vagy lebegőpontos számábrázolási formát kell választanunk.

Fixpontos számábrázolás

Ez a számábrázolási mód minden számot tizedes (kettedes) pont nélküli egész számként kezel. Az előjeles abszolútértékes ábrázolást két byte segítségével mutatjuk be. Minden bithez a kettes számrendszer helyi értékeit rendeljük. A legnagyobb helyi értéken álló bit az előjelbit. Negatív szám esetén értéke 1, pozitív számoknál 0. Például:

2^{15} (előjelbit)								2^0							
0	0	0	0	0	1	1	1	1	0	0	0	0	0	0	0

A példában ábrázolt szám tízes számrendszerbeli alakja:

$$2^{10} + 2^9 + 2^8 + 2^7 = 1024 + 512 + 256 + 128 = 1920$$

Törtszámokat is ábrázolhatunk így, ha valamelyik helyi érték elé kettedes pontot képzelünk:

2^{15}					2^0				
előjel	egészrész				tötrész				

virtuális bináris pont

A kettedes ponttól jobbra eső helyi értékek rendre $1/2$, $1/4$, $1/8$, $1/16$, stb. Így, ha az előbbi számot 11.11 törtként értelmezzük, akkor annak tízes számrendszerbeli értéke $2 + 1 + 1/2 + 1/4 = 3,75$.

Az eltolt nullpontú ábrázolásnál a kettes számrendszerbeli értékből ki kell vonnunk egy megállapodás szerinti értéket. Például 8 bit esetén:

$$\begin{aligned} 00000000 &= -128 \\ 00000010 &= -126 \\ \mathbf{10000000} &= \mathbf{0} \\ 11000000 &= 64 \end{aligned}$$

Komplemens vagy egyes komplement ábrázolás esetén a pozitív számot binárisan adjuk meg, a negatívot bitenként negáljuk. Például 8 bit esetén:

$$11111110 = -1$$

A kettes komplementum ábrázolás esetén a pozitív számot binárisan adjuk meg, a negatívot bitenként negáljuk, majd hozzáadunk 1-et. Például 8 bit esetén:

$$11111111 = -1$$

Lebegőpontos számábrázolás

A fixpontos számábrázolás hátránya, hogy a nagy és a kis számok is sok biten ábrázolhatók. A lebegőpontos számábrázolás alkalmazásánál a számokat

$$\text{szám} = (-1)^S M p^k$$

alakban adjuk meg, ahol S az előjel, M a mantissza, p az alap és k a karakterisztika, illetve $1/p \leq M < 1$.

A lebegőpontos számábrázolás a különböző architektúrák esetén különböző módokon történhet. Napjaink számítógépein az Institute of Electrical and Electronics Engineers (IEEE) által a nyolcvanas években kiadott IEEE 754 nevű szabvány a meghatározó. E szabvány szerint az egyszeres pontosságú (32 bites) számokat például

$$\text{szám} = (-1)^S (1.M) (2^{k-127})$$

alakban adjuk meg, és az alábbi módon ábrázoljuk:

előjel (S) (1 bit: 31.)	karakterisztika (k) (8 bit: 23-30)	mantissza (M) (23 bit: 0-22)
----------------------------	---------------------------------------	---------------------------------

Az előjel 1 bit hosszúságú. Negatív számok esetén értéke 1, pozitív számok esetén 0.

A karakterisztika 8 bit hosszúságú; ez jelöli ki a számban a ketteses pont helyét. A karakterisztikát eltolt nullpontú formában szokás tárolni. Ha a karakterisztika mező hossza k bit, akkor az eltolási érték $e = 2^{k-1} - 1$. Esetünkben $e = 127$.

A mantissza 23 bit, ami egy egészre normált törtszám, melynek első jegye mindig 1. Ezt a bitet a formátum nem tárolja, csak a törtrészt.

A tárolt számot az alábbi módon számolhatjuk ki:

$$\text{szám} = (-1)^S (1+M) \cdot 2^{k-127}$$

ahol S az előjelbit, M a mantissza, k a karakterisztika és e az eltolás.

A mantissza számára fenntartott bitek száma a számábrázolás pontosságát, míg a karakterisztika mérete az ábrázolható számok nagyságrendjét határozza meg.

Példa

Mely x számot ábrázoltuk 1 10000111 101000000000000000000000 módon, 32 biten?

Megoldás: Látható, hogy

$$s = 1; k = 10000111_2 = 135_{10}; M = 0.101_2 = 0,625_{10},$$

tehát

$$x = -1,625 \cdot 2^8 = -1,625 \cdot 256 = -416.$$

Feladatok

Mely számot ábrázoltuk 32 biten az alábbi módokon?

0 01111111 000000000000000000000000

0 01110101 010101000000000000000000

0 10110000 101010000000000000000000

1 00101010 111000000000000000000000

1 10000010 000101000000000000000000

Ábrázoljuk a következő számokat:

1; 300; -8,625.

Kódolt ábrázolás

Binárisan kódolt decimális (Binary Coded Decimal - BCD)

Ennél a módszernél a szám tízes számrendszerbeli számjegyeit ábrázolják számjegyenként 4 (pakolt) vagy 8 (pakolatlan) biten. Ez pazarló tárolási mód, de előnye, hogy nagyon könnyű a bitsorozatból a tízes számrendszerbeli alakot előállítani, hátránya viszont, hogy a műveletvégzés nagyon nehézkes.

Pakolt:

9613 —> 10010110 00010011 (2 byte)

Pakolatlan (1 karakter = 1 byte):

9613 —> 00001001 00000110
00000001 00000011 (4 byte)

Nem-numerikus karakterek, kódtáblázatok

Egy karaktert (számot, betűt, egyéb írásjelet) többnyire egy byte-on tárolnak. Ebből következően 256 lehetséges állapot van, ami elég a kis- és nagybetűk, számjegyek, írásjelek tárolására. Azt, hogy milyen byte-értékek milyen karaktert jelentenek, kódtáblázat tartalmazza. A használt kódtáblázat megállapodás kérdése, azonban a számítógépek közötti adatcsere miatt fontos, hogy az egyik gép ugyanolyan karakterként értelmezze a byte-okat, mint a másik. Ezért alakultak ki a szabványos kódtáblák, a legismertebb az **ASCII** (American Standard Code for Information Interchange) kódrendszer.

Eredetileg az ASCII kódrendszer 7 bites volt, 128 karaktert tartalmazott. (Ma ez a szabványos része a kódtáblának.) Ezek közül is a 0-tól 31-ig terjedő értékek az úgynevezett vezérlő karakterek, és a 32-től 127-ig terjedő értékek jelentenek megjeleníthető karaktereket. A 8 bites byte-ok használata óta az ASCII táblázat második, 128-tól 255-ig terjedő értékeket tartalmazó része alkalmazásfüggő, nem szabványos. Itt tárolhatók például az ASCII szabványban nem szereplő magyar ékezetes karakterek.

A szabványos ASCII kódtábla:

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040	 	Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	\$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	((72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051))	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[[123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174	|	
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135]]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177		DEL

Példa egy kiterjesztésre:

128	Ç	144	É	161	í	177	⌚	193	⊥	209	⌞	225	β	241	±
129	ü	145	æ	162	ó	178	⌛	194	⌞	210	⌞	226	Γ	242	≥
130	é	146	Æ	163	ú	179		195	⌞	211	⌞	227	π	243	≤
131	â	147	ô	164	ñ	180	⌞	196	—	212	⌞	228	Σ	244	∫
132	ä	148	ö	165	Ñ	181	⌞	197	+	213	⌞	229	σ	245	∫
133	à	149	ò	166	²	182	⌞	198	⌞	214	⌞	230	μ	246	+
134	â	150	û	167	°	183	⌞	199	⌞	215	⌞	231	τ	247	≈
135	ç	151	ù	168	¿	184	⌞	200	⌞	216	⌞	232	Φ	248	°
136	ê	152	—	169	—	185	⌞	201	⌞	217	⌞	233	⊙	249	·
137	ë	153	Ö	170	¬	186	⌞	202	⌞	218	⌞	234	Ω	250	·
138	è	154	Ü	171	½	187	⌞	203	⌞	219	■	235	δ	251	√
139	ï	156	£	172	¾	188	⌞	204	⌞	220	■	236	∞	252	—
140	î	157	¥	173	¡	189	⌞	205	=	221	■	237	φ	253	²
141	ì	158	—	174	«	190	⌞	206	⌞	222	■	238	ε	254	■
142	Ä	159	f	175	»	191	⌞	207	⌞	223	■	239	∩	255	
143	Å	160	á	176	⌚	192	⌞	208	⌞	224	α	240	≡		

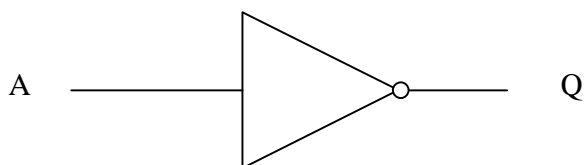
Műveletek a számítógépen

A digitális számítógépeket a kettes számrendszer alapján építik, mivel egy számjegy egy kétállapotú egységgel megvalósítható. A digitális logikai szintet a kapuáramkörök alkotják, amik analóg alkatrészekből épülnek fel, de működésükkel a bináris rendszer alapját képezik. A két állapot megkülönböztetésére két jelszintet alkalmaznak: az alacsony – a hamis vagy 0 – értéket a 0/1 V jelenti, a magas – az igaz vagy 1 – értéket pedig 2/5 V. Az e két intervallumon kívül eső feszültségek nem megengedettek.

Minden kapunak van egy vagy több digitális bemenete és egy kimenete. A kapuk kombinációjából felépített áramkörök leírására egy olyan algebrára van szükség, amiben a változók és a függvények csak 0 vagy 1 értéket vehetnek fel. A George Boole (1815-1864) által kitalált és róla elnevezett **Boole-algebra** ilyen. A Boole-algebrában összesen két szám van: a 0 és az 1. Itt már a szokásos összeadás műveletét sem definiálhatjuk, új műveletekre van szükség.

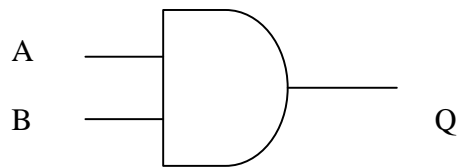
Egy n változós Boole-függvény bemeneti értékeinek 2^n lehetséges kombinációja van, ami egy 2^n soros táblázattal adható meg, amit **igazságtáblának** nevezünk.

Az egyváltozós művelet, ami megfordítja a bemenet értékét, azaz 0-ra 1-et, 1-re 0-t ad, a negáció, **NEM**, vagy angolul **NOT**, függvény és igazságtáblája az alábbi:



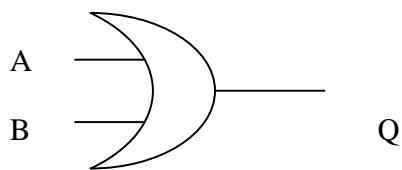
A	Q
0	1
1	0

A kétváltozós műveletek mindkét bemenete kétféle lehet. A bemenetek közötti műveletek az **ÉS**, **VAGY** és **KIZÁRÓ VAGY**. Az **ÉS**, vagy angolul **AND** művelet a konjunkció, ami csak akkor ad egyet, ha az egyik és a másik bemenete is egy.



A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

A másik művelet a diszjunkció, a **VAGY**, vagy angolul **OR**, ami akkor ad egyet, ha vagy az egyik vagy másik bemenete egy.



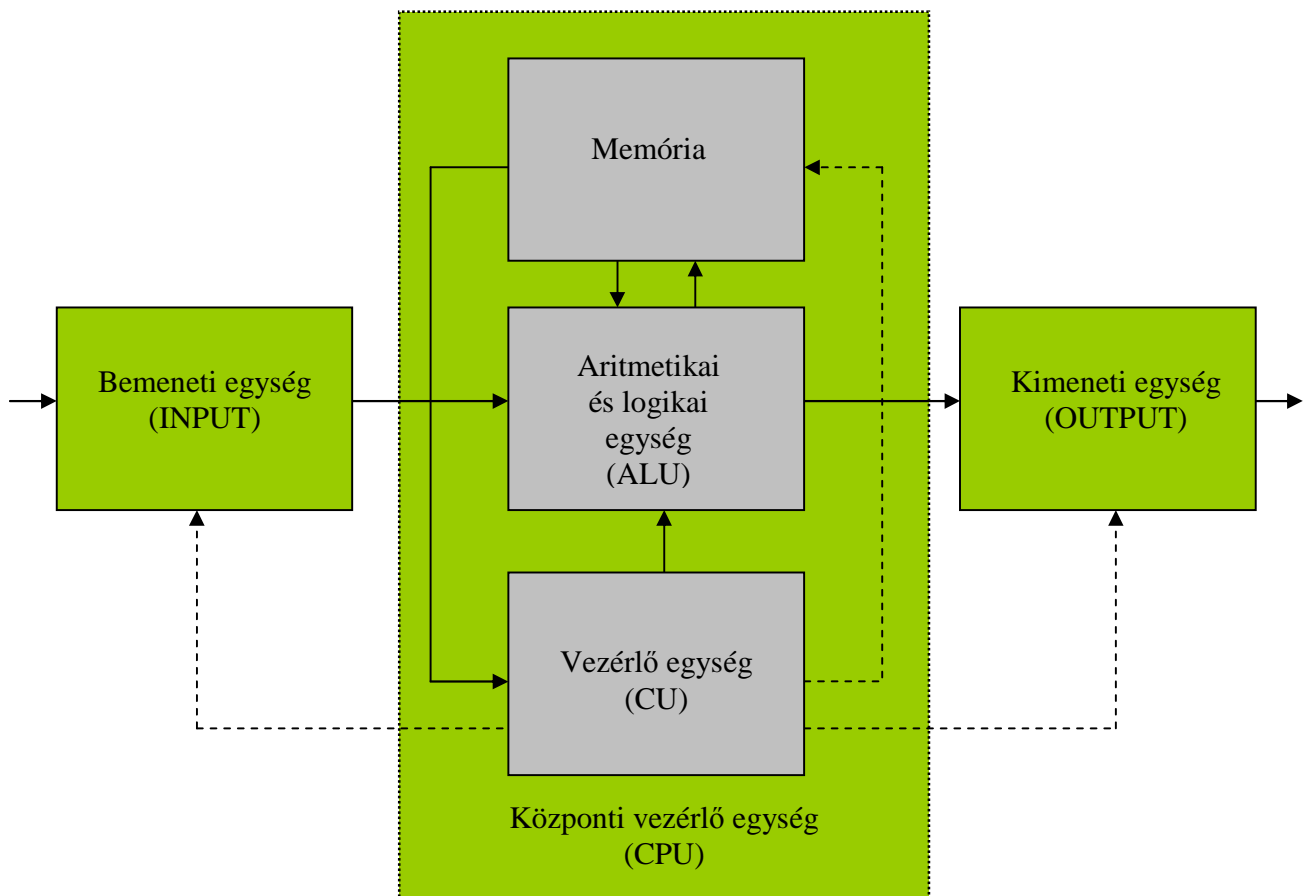
A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

Ezekkel az alapl műveletekkel már a többi kétváltozós művelet is megadható. Például az előző műveletekkel kifejezve a **KIZÁRÓ VAGY**, vagy **XOR** művelet akkor ad 1-et, ha a két bemenete különböző:

$$A \text{ XOR } B = (A \text{ AND NOT } B) \text{ OR } (\text{NOT } A \text{ AND } B).$$

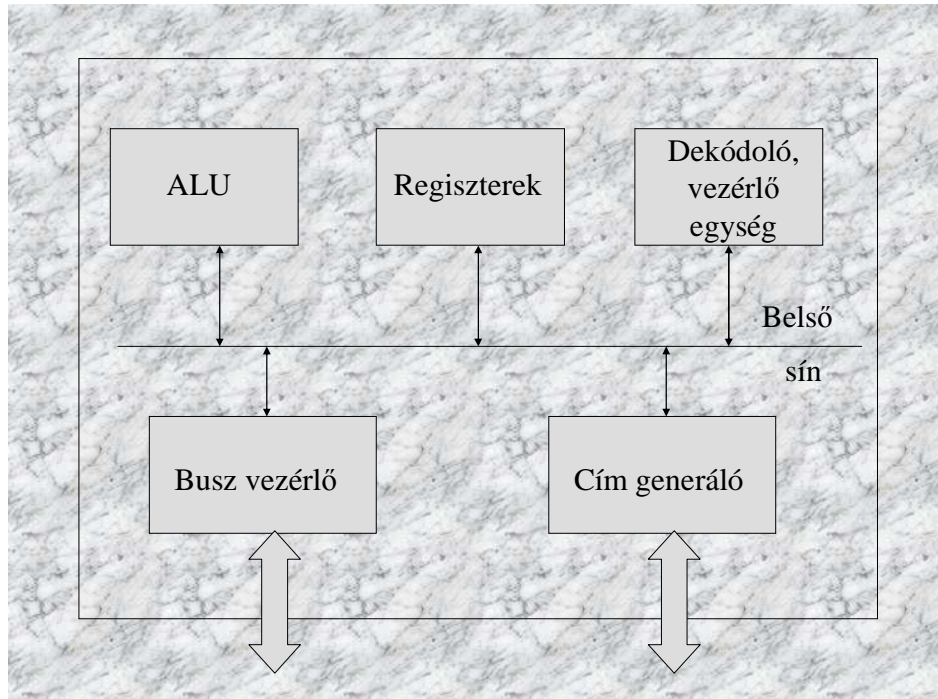
A számítógép felépítése

A számítógép működésének megértéséhez szükséges, hogy ismerjük a hardver felépítését, és tisztában legyünk a hardverelemek funkcióival. A következő ábra a számítógép funkcionális felépítését, a Neumann-modellt mutatja:



A be- és kimeneti (I/O) egység feladata értelemszerűen a kommunikáció biztosítása a számítógép felé, illetve felől. A CPU (Central Processing Unit) feladata az operatív tárban (memóriában) elhelyezkedő program feldolgozása és végrehajtása. Minden, ami a rendszerben történik, innen származik, mint parancs, vagy ide fut be, mint jelzés. A műveletvégrehajtó egység az ALU (Arithmetic and Logical Unit).

A korszerű számítógépekben a központi feldolgozó egység a **processzor**. Olyan elektronikai alkatrész, nagy bonyolultságú félvezető eszköz, mely ma már egyetlen, nagy integráltságú lapkán tárolófelület, vezérlő-, illetve input-output funkciókat ellátó elemeket tartalmaz. Dekódolja az utasításokat, vezérli a műveletek elvégzéséhez szükséges belső adatforgalmat és a csatlakozó perifériális berendezések tevékenységét.



A processzor teljesítménye alatt azt az időt értik, amelyre a processzornak szüksége van egy bizonyos feladat végrehajtásához. A processzornak két lényeges jellemzője, amelyek utalnak a teljesítményre: a **szóhossz** (bitszám vagy bitszélesség) és az **órajelfrekvencia**.

A szó hosszát, amellyel a processzor dolgozik, belső szóhossznak nevezzük. Emellett fontos még a buszrendszer szóhossza is: az adatbusz és a címbusz bitszélessége. Az adatbusz szélessége azt jelenti, hogy a processzor hány bitet tud egyidejűleg a hozzá kapcsolt perifériákra küldeni. A címbusz közvetíti azokat a jeleket, amelyek a tárolóhelyek eléréséhez szükségesek. A címbusz szélessége határozza meg a közvetlenül megcímezhető címtartomány nagyságát.

Az órajelfrekvenciát a vezérlőkvarc (órajeladó) hozza létre, amely vagy közvetlenül integrálva van a processzorba, vagy azon kívül helyezkedik el. A rendszeróra folyamatosan, periódikusan jeleket szolgáltat. Két ilyen jel ad ki egy processzorciklust. Az egyszerű utasításokat kevesebb, míg a bonyolultabbakat több processzorciklus alatt hajtja végre a processzor. Két

processzorciklus alkot egy buszciklust, melyek során a processzor a memóriához fordul. Az első ciklus során a memória címzése történik meg, a második ciklus alatt a processzor az utasítást közli.

Az órajelet megahertzben (MHz) mérik. Egy Hertz az a frekvencia, amely 1 másodperc alatt egy rezgést végez. A 8 MHz tehát azt jelenti, hogy a kvarc másodpercenként 8 milliószor rezeg. Ez a rezgés határozza meg az utasítások végrehajtásának gyorsaságát. Általában azt lehet mondani, hogy minél magasabb az órajel, annál gyorsabban tud a számítógép dolgozni. Ha a rendszeróra frekvenciáját növeljük, akkor a processzor gyorsabban fogja végrehajtani az utasításokat. A processzor sebességét a MIPS (Million Instruction Per Second) mutatja meg, azaz, hogy mennyi utasítást képes a processzor elvégezni másodpercenként.

Egy processzor utasításkészlete gépi kódú (elemi) utasítások összessége, melyek végrehajtására a processzor hardver szinten alkalmas. A számítástechnika fejlődése során a processzorok tervezésében két irányvonal alakult ki.

Kezdetben a **CISC** (Complex Instruction Set Computer = bonyolult utasításkészletű számítógép) architektúrájú gépek voltak többségben. Ezek főbb jellemzői:

- sok utasítás, akár néhány száz, közöttük több összetett;
- bonyolult címzési módok lehetségesek, így viszont változó hosszúságúak az utasítások, ami nehezen optimalizálható;
- a gépi utasítások változó vagy több ciklusidőt igényelnek;
- az assembly programozás egyszerűbb, mert a bonyolult utasítások bonyolult feladatokat oldanak meg;
- csak a szükséges néhány regiszterrel rendelkezik;
- ismertebb CISC processzorok (Intel 286/386/486, Pentium; Motorola 68000; DEC VAX).

A **RISC** (Reduced Instruction Set Computer = csökkentett utasításkészletű számítógép) architektúrájú gépek főbb jellemzői:

- csak a legalapvetőbb utasítások léteznek gépi szinten;
- sok regiszter van, ezért kevesebb a tárművelet, több a regiszterművelet, ezért gyors;
- fix a kódhosszúság, ezért egyszerűek a címzési módok;
- egyszerű és gyors a kódolás, így a ciklusok száma kicsi;

- az egy feladatra eső utasítások száma kevés, mert az operációs rendszerhez, illetve a compiler-ekhez tervezik;
- az egyszerű utasítások egyforma hosszúságúak, azonos ciklusidejűek;
- a bonyolult feladatok programozása bonyolult, hosszú;
- ismertebb RISC processzorok (DEC Alpha; HP PA-RISC; SUN SPARC; IBM PowerPC és RISC6000).

Egy új megoldás az **EPIC** (Explicitly Parallel Instruction Computing = teljes párhuzamosságú utasításokon alapuló számítógép) technológia, amely támogatja a nagy párhuzamosságot: 20 művelet végrehajtását teszi lehetővé órajelenként. Az EPIC architektúra számos olyan új jellemzőt is magába foglal, amelyek tovább növelik a processzor teljesítményét. Ezek olyan megoldások, amelyek csökkentik a processzor megállásait, illetve folyamatosabbá teszik működését. Az alkalmazások képesek előtölteni az adatok jelentős részét a virtuális memóriába, így lehetővé téve a processzor villámgyors elérését. Ez csökkenti az adatok virtuális memóriába töltésének idejét, valamint a keresést, olvasást, írást a tárolóeszközre, így téve lehetővé az alkalmazásoknak, hogy gyorsabban és hatásosabban fussanak. Ilyen EPIC processzor az Intel Itanium (IA-64). Jellemzői:

- Új utasításkészlet.
- 128-bites utasításcsomag:
- 3 db 41-bites utasítás (=123 bit);
- a maradék 5 bit határozza meg az utasítások típusát a csomagban.
- 128 db 64-bites általános használatú regiszter.
- 128 db 82-bites lebegőpontos regiszter.
- Intel X86-os utasítások végrehajtása.
- Az utasítások párhuzamos végrehajthatósága.

A memóriák

A számítógép memóriájának legkisebb címezhető egysége a **byte**, így a **memóriakapacitás** mértékegysége is a byte, illetve annak többszörösei:

1 **kbyte** (kilobyte) = 1024 byte (2^{10} byte);

1 **Mbyte** (megabyte) = 1024 kbyte (1024×1024 byte = 1 048 576 byte);

1 **Gbyte** (gigabyte) = 1024 Mbyte ($1024 \times 1024 \times 1024$ byte = 1 073 741 824 byte);

1 **Tbyte** (terabyte) = 1024 Gbyte ($1024 \times 1024 \times 1024 \times 1024$ byte = 1 078 036 791 296 byte).

Írhatóság szerint a memóriákat két csoportba osztjuk. A **RAM** (Random Access Memory – *Megjegyzés:* Az elnevezés helytelen, mert ma már minden memória véletlen elérésű, de annyira elterjedt ez az elnevezés, hogy zavart okozna a megváltoztatása.) írható és olvasható memória, mely az áram kikapcsolásával teljes tartalmát elveszti. Tartalma tetszőlegesen módosítható, akárhányszor alkalommal. Feladata, hogy munka közben a változó adatokat tartalmazza.

A RAM memóriák között felépítés szerint megkülönböztetünk dinamikus (Dynamic RAM vagy DRAM) és statikus (Static RAM vagy SRAM) memóriát. A DRAM kondenzátorokból áll, melyek töltésüket idővel elvesztik, ezért a DRAM-ot meghatározott időközönként frissíteni kell. Az SRAM integrált áramköri tranzisztorokból épül fel, nem kell frissíteni, gyorsabb a DRAM-nál.

A **ROM** (Read Only Memory) csak olvasható memória, mely kikapcsoláskor sem „felejt”. Feladata, hogy tartalmazza azokat az adatokat és programokat, melyekre már a gép bekapcsolásakor szükség van. A ROM-ot többnyire a számítógéppel együtt szállítják.

Eredetileg a ROM-ba gyárilag „égetik be” a programot. A technológia fejlődésével létrehoztak olyan ROM-okat, melyeket egy egyszerű eszköz segítségével a felhasználó maga programozhat (**PROM**, Programmable ROM). Az **EPROM** (Erasable PROM) UV fény segítségével törölhető, majd újraégethető. Az **EEPROM** (Electronically Erasable PROM) elektromos úton, a processzor utasításai alapján törölhető és programozható át, tehát nem kell a gépből kiszerezni az átprogramozáshoz. Az EEPROM egy speciális típusa a **Flash** memória, melynek törlése és újraprogramozása nem byte-onként, hanem blokkonként történik.

A processzoron belül is vannak memóriatípusú elemek. A **regiszter** a processzorba beépített nagyon gyors elérésű, kisméretű memória. A regiszterek ideiglenesen tárolják az információkat, utasításokat addig, amíg a processzor dolgozik velük. A regiszterek között nem csak adattároló elemek vannak, hanem a processzor működéséhez elengedhetetlenül szükséges számlálók és állapotjelzők is.

A modern processzorok fontos része a **cache** (gyorsítótár). A cache a processzorba vagy a processzor környezetébe integrált memória, ami az **operatív tár** (RAM, ROM) viszonylag lassú elérését hivatott kiváltani azoknak a programrészeknek és adatoknak előzetes beolvasásával, amikre a végrehajtásnak közvetlenül szüksége lehet. A gyorsítótár mérete ma már Mbyte-os nagyságrendű.

A RAM, a ROM, a cache és a regiszter az úgynevezett fő vagy **elsődleges memória** típusai. A **másodlagos memóriák** a nagy tárolókapacitással rendelkező háttértárolók. (Egyes szakirodalmak csak a gyors mágneses háttértárolókat sorolják ide, és **harmadlagos** tárolóknak nevezik az optikai és a mágnesszalagos eszközöket, mint tipikusan archiválásra használt berendezéseket.)

Perifériák

A számítógéphez nagyon sokféle eszköz, úgynevezett periféria csatlakoztatható. Vannak beviteli célú eszközök, más eszközök pedig kivitelre valók. A háttértárak mind a két célt szolgálják. Az átvitel vagy az adatoknak az operatív memóriából a periféria felé való kiküldését, vagy a periféria felől érkező adatok memóriába való tárolását jelenti. A háttértárolók esetén ez a folyamat kétirányú: az adatokat tárolni is és visszaolvasni is tudjuk.

I/O vezérlő

A perifériák és a központi egység közötti adatforgalom vezérlésére, a sebességkülönbség áthidalására **perifériavezérlő** (I/O vezérlő) **alrendszer** alkalmaznak, ami a processzor feltartása nélkül bonyolítja le az adatátvitelt. A processzor csak elindítja a folyamatot, aminek a befejezésről a perifériavezérlő megszakítással értesíti azt. Egy ilyen alrendszer általában egyidejűleg több periféria kiszolgálására is képes.

Párhuzamos és soros adatátvitel

A bitek továbbítása alapvetően két különböző módon történhet. A legegyszerűbb eset, amikor a biteket sorban egymás után egy csatornán elküldjük a vevőnek. Ezt az átviteli módot nevezik **soros adatátvitelnek**.

A másik lehetőség, hogy az adó és a vevő között annyi vonalat alakítunk ki, amennyi bitet egyszerre át szeretnénk vinni. Ebben az esetben tehát bitcsoportok átviteléről van szó. Ezt az adatátviteli módot **párhuzamos adatátvitelnek** nevezik.

Mindkét módszernek van előnye és hátránya. A soros átvitel kialakítása olcsó, mivel kevés számú kapcsolódásra van szükség, de ezzel együtt az átvitel sebessége a párhuzamos átvitelhez képest lényegesen kisebb. A soros kapcsolattal nagyobb távolság hidalható át, mint a párhuzamossal. Azt, hogy melyik módszert alkalmazzák, egyértelműen a feladat dönti el. Általában mikroszámítógépek belső áramköreinek az összekapcsolására párhuzamos módot választanak a kis távolságok és a nagy átviteli sebesség miatt. A külső eszközök összekapcsolása a számítógépekkel már mindkét módszer szerint történhet (például az egér soros, a nyomtató viszont párhuzamos átvitelt használ).

FireWire

A FireWire egy nagysebességű, soros adattovábbító technológia a különböző kiegészítő eszközöknek a számítógéphez való illesztésére. Az Apple által kifejlesztett rendszer ma már ipari szabvány, melyet az IEEE 1394 néven jegyeztek be. A FireWire-t úgy tervezték, hogy könnyedén továbbítsa a különböző, magas átviteli képességet igénylő multimédia adatokat az eszközök között (kamerák, szintetizátorok stb. és/vagy merevlemezek). Kiemelkedő teljesítményével, üzemi közbeni illeszthetőségével, az illesztett eszközök önműködő beállításával kategóriájában egyedülálló illesztési szabvány.



USB

Az **USB** (Universal Serial Bus) olyan csatlakozási szabvány perifériák számára, amely függetlenül attól, hogy milyen berendezésről, operációs rendszerről vagy platformról van szó, lehetővé teszi a kapcsolódást. Átviteli sebessége kellően nagy, hogy szinte bármilyen kis vagy közepes adatforgalmú külső egységet használhassunk vele, előállítása pedig kellően olcsó, hogy megérje gyártani.

Az USB eszközök valóban megvalósítják a Plug&Play elvét, azaz amint csatlakoztatjuk az eszközt, már működik is. A régebbi rendszereken a Plug&Play azt jelentette, hogy a legközelebbi újraindításnál ismeri fel az operációs rendszer az új hardver elemet.

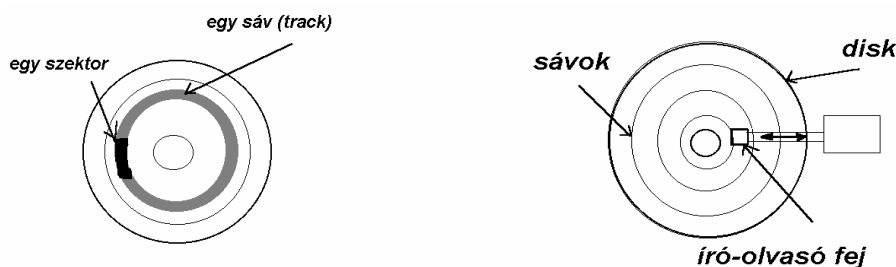


Háttértárolók

A mágneses háttértárolók

A mágneses háttértárolás esetén egy nem mágnesezhető felületre (alumínium, műanyag) vékony rétegben felhordott ferromágneses anyagot meghatározott módon, a tárolandó információnak megfelelően változó irányú mágneses térrel átmágneseznek úgy, hogy a kialakult maradandó (remanens) mágnesesség elegendő erejű legyen ahhoz, hogy a tárolt információ kiolvasásakor a felület felett elhaladó olvasófejben változó irányú áramot indukáljon. A kódolást, dekódolást speciális vezérlő áramkörök végzik.

Az adatokat **közvetlen** (direkt) **elérhető** módon tárolják a **mágneslemezek**en. A lemezek alapanyaga szerint megkülönböztetünk **hajlékony** (floppy) vagy **merev** (winchester) mágneslemezeket. A rögzítés elve mindkét esetben ugyanaz – az adatokat a lemez felszínén koncentrikus körök (sávok = track-ek) mentén rögzítik -, de az elérhető adatsűrűség a merevlemezen nagyságrendekkel nagyobb, mint a hajlékony lemezen. Az adatokat a sávokon belül **szektorokban** tárolják. A tárolás logikai egysége a **klaszter**, ami rendszerektől függően eltérő számú szektorból állhat. Gyárilag rögzített szektorok esetén **hardszektoros**, egyébként **szoftszektoros** lemezekről beszélhetünk. A legjellemzőbb szektorméret 512 byte.



A **mágnesszalagos** tárolók **soros elérésűek**, mivel egy bizonyos adat megkereséséhez az összes előtte levő adaton végig kell haladni. A mágnesszalagokon az adatokat **blokkokban** tárolják, amiket egy üres, adatot nem tartalmazó rész, úgynevezett **gap** választ el egymástól. Az írássűrűséget a szalag egységnyi hosszúságán elhelyezhető bitek számával jellemezzük.

Az optikai háttértárolók

Az optikai elven működő háttértárolók esetén egy polírozott üveglemezre fotoérzékeny agyagot visznek fel, az adatokat lézersugár segítségével írják, illetve olvassák. Fajtái:

- **CD-ROM**: gyárilag rögzített tartalommal rendelkezik, csak olvasható. A lézerfény visszaverődési és kioltási (interferencia) tulajdonságait használják a lemezen tárolt adatok olvasásához.
- **CD-MO** (Magneto-Optical): írható, törölhető és újraírható. A fény mágneses térben való viselkedését használják ki az adattárolás érdekében.
- **CD-R** (Recordable) vagy **WORM** (Write Once Read Many): egyszer írható, sokszor olvasható.
- **CD-RW** (ReWritable): írható, törölhető és újraírható.
- **DVD, DVD-RAM, DVD-RW**: működése lényegében megegyezik a CD-ével, a sávok azonban sokkal sűrűbben helyezkednek el.

RAID

A processzorok teljesítménye évről évre megsokszorozódik, míg a háttértárolók kapacitása csak kisebb mértékben növekszik. A felvetődő problémák megoldására különböző megoldásokat dolgoztak ki. A **RAID** (Redundant Array of Inexpensive Disks vagy Redundant Array of Independent Disks) egy olyan technika, melyet a háttértárolók nagyobb megbízhatósága és a tárolókapacitás növelése érdekében fejlesztettek ki. A lemezekre írt adatokhoz redundáns információkat is társítva lehetővé teszi azoknak helyreállítását bizonyos mennyiségű adat megsérülése esetén. A RAID technológia lényege a nevében is benne van: több független merevlemez összekapcsolásával egy nagyobb méretű és megbízhatóságú logikai lemezt hozunk létre.

Pendrive

A félvezető technika fejlődésének köszönhetően ma már egyre kisebb méretű, de egyre nagyobb tárolási kapacitású adathordozókat gyártanak. Ilyen például a **pendrive**, ami egy USB csatlakozóval egybeépített flash memória.



Bemeneti perifériák

Az **input egységek** (beviteli eszközök) segítségével visszük be a számítógépbe mindazokat az információkat, amelyekre a feldolgozáshoz szükség van, tehát a feldolgozandó adatokat és programokat. Ezek az eszközök nem csak az adatmozgatást végzik, hanem az adatokat az ember által értelmezhető formáról átalakítják a gép által értelmezhető formára.

Billentyűzet

A **billentyűzet** (keyboard, klaviatúra, konzol) az elsődleges bemeneti periféria. A billentyűzet több részre tagolódik. Az **alfanumerikus** rész az írógépekre hasonlít, amely a karakteres billentyűket tartalmazza.

A **váltóbillentyűk** csoportjába sorolható az **Alt** jelentésmódosító (funkcióváltó) gomb, ami csak valamilyen más billentyűvel együtt lenyomva hatásos. (Az **Alt Gr** a Windows-os klaviatúrák jelentésmódosító gombja. Így a Shift-tel és az Alt Gr-rel egy-egy gombhoz három különböző jelet is hozzárendeltek.) A **Ctrl** (vezérlőváltó) gomb szintén jelentésmódosításra szolgál, és más billentyűvel együtt lenyomva van hatása. (Érdemes megjegyezni, hogy az Alt és a Del gombok lenyomásával egy időben használva a számítógép újraindul; Windows rendszerben a Windows Feladatkezelő ablak aktiválását váltja ki ez a művelet.) A **Shift**. speciális váltógomb, amelyre azért van szükség, hogy a billentyűzeten minél több karakter helyet kapjon. Így bizonyos gombokat megosztottak, azaz két különböző jel megjelenítésére is képessé tették. A Shift-et

lenyomva a billentyűk felső részére festett jelet aktivizálhatjuk, míg betűk esetén a nagy- és kisbetű közötti váltást eredményezi.

A **kapcsolóbillentyűk** csoportjába tartozik a **Caps Lock**, amelyet kikapcsolva kisbetűket, bekapcsolva nagybetűket írhatunk. A **Num Lock** bekapcsolásával számbillentyűzetként, kikapcsolásakor kurzorblokként használható a külön blokkban elhelyezett numerikus billentyűzet. A **Scroll Lock** ritkán használt billentyű, amelyet a képernyőn történő szöveggörgetés módosítására (ki- és bekapcsolására) terveztek.

A **szerkesztőbillentyűk** csoportjába tartozó **Ins** vagy **Insert** billentyű a beszúrás/felülírás váltására szolgál. A **Del** vagy **Delete** az aktuális pozícióban levő karakter, a **Back Space** vagy **←** pedig az aktuális pozíció előtti karakter törlésére való.

A **numerikus billentyűket** a gyorsabb adatbevitel érdekében hozták létre a billentyűzet jobb oldalán. (Megfigyelhető, hogy az úgynevezett origógomb az 5-ös felirattal kézzel tapintható jelzéssel van ellátva azok számára, akik „vakon” szeretnék a numerikus padot kezelni).

A **kurzormozgató** billentyűk (**↑**, **↓**, **←**, **→**, **Page Up**, **Page Down**, **Home**, **End**) értelemszerűen a kurzor mozgatását szolgálják.

A **funkcióbillentyűk** (F1 - F12) olyan vezérlőgombok, melyekhez a futó program rendelhet értelmet, így a programok kezelése is egyszerűbbé válik segítségükkel.

Az **Esc** az aktuális feladat törlésére, abból való kilépésre szolgál.

Az **Enter** billentyű a bevitelt, az utasítás értelmezését, a végrehajtást kezdeményező billentyű. Bizonyos alkalmazásokban az új sor jelzésére szolgál.

A **pillanatstop** gomb a **Pause** vagy **Break**, amivel egy éppen futó feladat felfüggesztését kezdeményezhetjük.

A **Print Screen** gomb a képernyő teljes tartalmát a vágólapra teszi, ahonnan az kinyomtatható.

Egér

A bemeneti **perifériák** közül a második legfontosabb az **egér** (mouse) egy úgynevezett egérkurzort használ, amely a képernyőn pontosan követi az elmozdulás irányát. Ennek segítségével lehet rámutatni a megfelelő objektumra (például ikon, menüpont, nyomógomb) majd az egér gombjával aktivizálni. (Ezt a műveletet a számítástechnikai szlengben „klikkelésnek”

hívják, ugyanis a műveletet egy kattanó hang jelzi, amely az egér gombjának lenyomásával keletkezik). Nyomógombokból, alapértelmezés szerint kettő található. Sok modell azonban hárommal rendelkezik, de a középsőt csak speciális programozással lehet használhatóvá tenni. A manapság használatos egerek középső gombja azonban görgető funkcióval is rendelkezik, ami jelentősen meggyorsítja például a dokumentumon belüli mozgást.

A **mechanikus** típusú egér esetén egy gumival bevont fémgolyó mozgását követi két érzékelő korong (az egyik a függőleges-, míg a másik a vízszintes elmozdulás állapotát figyeli). Az ilyen egerekhez speciális alátétet (úgynevezett egérpadot) árusítanak, amely csúszásmentes felületet biztosít az egér golyójának.

Az **optikai** típusú egér alján egy kis optikai érzékelő figyeli az elmozdulás irányát és sebességét.

Egyéb beviteli eszközök

A **lapolvasó** (scanner) képek, illetve írógéppel írt vagy nyomtatott szövegek bevitelére alkalmas.

A **vonalkódolvasókat** főként a kereskedelemben használják, de már terjedőben van olyan helyeken (például raktárakban, könyvtárakban), ahol sok különböző tárgyat kell gyorsan, egyszerűen azonosítani.

A műszaki gyakorlatban használják a **digitalizáló táblát**. A készülékkel vonalas rajzok (térképek, tervrajzok) számítógépbe vitelét valósíthatjuk meg könnyen.

A **videokamera és mikrofon** a multimédiás kommunikáció (pl. videokonferencia) lebonyolításának elengedhetetlen hardver eszközei.

A **digitális fényképezőgép** a látható világ képpontokká történő leképezését végzi. Az eredmény annál inkább közelíti a valóságot, minél több képpontból (pixel) áll össze a keletkezett kép.

Kimeneti perifériák

Az **output egységek** (kiviteli eszközök) a gép által végrehajtott feladatok eredményeinek megjelenítésére valók, vagyis a géptől a felhasználó felé közvetítenek információkat.

Monitor, videokártya

A **monitor** (megjelenítő, képernyő, display) a számítógépek elsődleges kimeneti perifériája, az információk megjelenítésére szolgál. Alaphelyzetben minden szöveg, ábra és egyéb

megjeleníthető információ a képernyőre kerül. A gép a memóriájából viszi át az adatokat a monitorra, tehát itt is egyirányú, de a billentyűzettel ellentétes adatáramlásról van szó. Az adatfeldolgozás eredményei, a gép üzenetei, a billentyűzeten begépeltek szöveg is kikerül a képernyőre, és ezen láthatjuk minden egérrel végzett műveletünk folyamatát és eredményét is.

A **videokártya** tartalmazza azt az elektronikát, amely a monitort illeszti számítógépünkhöz. A kártya paraméterei (típusa) határozzák meg azt a monitortípust, melyet használnunk kell, ha a kártyánk képességeit ki akarjuk használni. A PC-k hőskorában csak monokróm adapterek léteztek. Ezek egyik legelterjedtebb típusa az **MDA** (Monochrome Display Adapter) kártya volt. Ez a kártya csak szöveg kiírására volt alkalmas, grafikus ábrát nem tudott megjeleníteni. Később a Hercules Corporation kifejlesztette a népszerű **Hercules** videokártyát (HGC, Hercules Graphics Controller). Ez a kártya már képes volt monokróm grafikus ábrákat is megjeleníteni. A színes szövegek és képek előállításához kifejlesztették a **CGA** (Color Graphics Adapter) színes grafikus videokártyákat. Ez a kártya 640 x 200 képpontos (**pixel**) felbontással csak két színt, 320 x 200-as felbontással a létező 16 színből egyszerre már négy színt tett láthatóvá. Az **EGA** (Enhanced Graphics Array) videokártya 640 x 350-es felbontással és 64 színnel dolgozott, de ebből csak 16 színt tudott megjeleníteni egyidejűleg a képernyőn. Ezek a kártyák még digitális videojelet szolgáltatnak a monitorok számára. Ezeken kívül még számos videokártya jelent meg a piacon. 1987-től kezdték gyártani az első **VGA** (Video Graphics Array) adaptereket, amelyek már analóg videojelet szolgáltatnak. A felbontásuk 640 x 480 képpont. Ennek a kártyatípusnak a továbbfejlesztésével a fejlesztők eljutottak napjaink legelterjedtebb **SVGA** (Super Video Graphics Array) videokártyájához, melynek felbontása rugalmasan változtatható a monitor és a felhasználó igényei szerint. A jelenleg használt SVGA kártya felbontása 640 x 480; 800 x 600; 1024 x 768, 1152 x 864 vagy 1280 x 1024 képpont.

A videokártya felbontása a képernyőn megjelenő **pixelek** számát jelenti. Ha nagyobb a kártya felbontása, nagyobb a pixelek száma is, így élesebb a képernyőn megjelenő kép. Az ideális videokártyának nagy felbontása van, és ezzel a felbontással képes sok színt megjelenítésére. Az, hogy egy kártya hány színt tud megjeleníteni a különböző képfelbontások esetén, a kártyán elhelyezett, úgynevezett videó memória nagyságától függ.

Működési elvük szerint a képernyők lehetnek:

- katódsugárcsőes (CRT);

- plazmakijelzős (LED);
- folyadékkristályos(LCD);
- vékonyfilm tranzistoros(TFT).

Fontos jellemzője a képernyőknek a **képátló** mérete, amelynek mértékegysége az inch (coll), jele a " (1 inch = 2,54 cm).

Nyomtatók

Amióta számítógépek léteznek, azóta használnak hozzájuk **nyomtatókat** (printer) is. A nyomtatók feladata az információ papírra rögzítése az ember által olvasható formában. A nyomtatókat különböző szempontok szerint csoportosíthatjuk.

A nyomtatási technikák szerint lehetnek:

- impact (érintéses vagy ütő) nyomtatók, melynél a rögzítés az érintés hatásán alapul (mátrix, gömbfejes, margarétakerekes, íróhengeres, íróláncos, írórudas nyomtatók);
- non-impact (érintés nélküli) nyomtatók (hő-, tintasugaras, elektrosztatikus, mágneses nyomtatók).

A kinyomtatott karakterek megjelenítési módja szerint:

- teljes karaktert író;
- pontokból összeállított karaktert író.

Az egyszerre kinyomtatott karakterek száma alapján:

- karakternyomtatók;
- sornyomtatók;
- lapnyomtatók.
- Az írásminőség alapján:
- levélminőségű (LQ = Letter Quality);
- közel levélminőségű (NLQ = Near Letter Quality);
- piszkozati minőségű (Draft).

A napjainkban leggyakrabban használt nyomtatók a mátrix-, a tintasugaras és a lézernyomtatók.

A **mátrixnyomtató** működésének alapja egy apró tűket tartalmazó írófej, amelyen függőlegesen 9 (7, 12, 18, 24) tű helyezkedik el egymás alatt (a 24 tűs modellek esetében kettő ilyen 12

darabos oszlop található). A nyomtató vezérlőelektronikája minden egyes kinyomtatandó karakter képét pontokból állítja össze, és ennek megfelelően nyomja rá a tűket a festékszalagra. A mátrixnyomtatóknál az úgynevezett traktor alkalmassá teszi a berendezést arra, hogy leporellós papírlapokra nyomtathassunk. További fontos jellemzőjük az eltérő nagyságú (A3, A4), és a kettő-, illetve hárompéldányos, önindigós papír használata. A mátrixnyomtatóknál megjelentek a színes szalagok. Ezek olyan osztott két- vagy háromszínű szalagok, amelyeknél az egyik szín a fekete, míg a másik a piros vagy a zöld (esetleg mindkettő).

A **tintasugaras** nyomtató a nyomtatófejben található fúvókán (kis átmérőjű lyukon) keresztül finom tintacseppeket juttat a papírra a tintapatronból. A patronban található egy kisebb kamra, amelyben a nyomtatáshoz szükséges festékmennyiség található. (A teljes festékmennyiséget egy szivacszerű anyag hordozza, ezáltal elérhető a szivárgásmentesség.) A kamra hátulján levő piezoelektromos kristály a rá adott feszültség hatására megváltoztatja méretét, aminek hatására egy csepp tinta lökődik a fúvókán keresztül a papírra. A feszültség megszüntetése után a piezoelektromos elem alakváltozása szívó hatást fejt ki, amelynek eredményeképpen újabb adag tintát szív a kamrába.

A **lézernyomtató** a mai legmodernebb nyomtató. Működésének alapja egy gumihenger, amely vegyi anyaggal, szelénnel van bevonva. Egy lézerberendezés is található a készülékben, amely folyamatosan sugarakat bocsát ki. Egy lézersugár egy tükörrendszer segítségével a hengerre pontokat rajzol, a pontok helyén feszültség keletkezik, és így a szintén töltéssel rendelkező szilárd festékpó (toner) a megfelelő helyekről lelökődik. A hengeren csak a nyomtatandó helyen marad festék, mely egy ellentétes tér hatására a hengerről a papírra tapad. A festék papírra égetését a felfűtött hengerek közötti átvezetéssel érik el.

Tintasugaras- és lézernyomtatók nyomtatási minőségét **DPI**-ben (Dot Per Inch) mérik. Ez a kinyomtatott információ finomságát jelenti, azaz azt, hogy milyen közel vannak egymáshoz azon pontok, amelyekből a kívánt adat összeáll.

Egyéb kiviteli eszközök

Plotter

A **plotter** (rajzgép) a nyomtatók mintájára készült berendezés, amely teljes mértékben műszaki rajzok készítésére alkalmas. A CAD (Computer Aided Design) rendszerek előszeretettel használják.

Hangszóró

A hangszóró hangkártyára csatlakozik. A hangszórókat sok esetben egyes monitorokba be is építik.

Számítógéphálózatok

A számítógéphálózatokat kommunikációs csatornákkal összekötött, egymással kommunikálni tudó számítástechnikai eszközök vagy csomópontok alkotják. A csomópontok számítógépek, terminálok, munkaállomások vagy különböző kommunikációs eszközök lehetnek a térben tetszőlegesen elosztva. A kommunikáció különböző átviteli közegeken keresztül történik a csomópontok között.

A hálózat célja a feladatok ésszerű megosztása a tagok között, ami nyilvánvaló kölcsönös előnyökkel jár: kommunikáció; távoli hozzáférés; rendszerfelügyelet; közös adatok, állományok elérése; költséges perifériák használata; nagyobb megbízhatóság.

A hálózatoktól elvárt legfontosabb tulajdonságok:

- összeköthetőség (a különböző hardver- és szoftverelemek kompatibilitása);
- egyszerűség (a felhasznált hardver és szoftver elemek könnyen installálhatók és működtethetők);
- modularitás (a különböző gyártóktól származó elemekből mint építőkockákból felépíthető az igényeknek megfelelő hálózat);
- megbízhatóság (hibamentes adatátvitel biztosítása);
- hajlékonyság (lehetőség a továbbfejlesztésre az igények bővülésekor);
- sokoldalúság (sokféle szolgáltatás egyszerű hozzáféréssel).

A hálózatok fontos jellemzője a **topológia**, ami a csomópontok geometriai elrendezését és összeköttetését jelenti. Az összeköttetés lehet teljes vagy részleges.

Sín vagy busz topológia

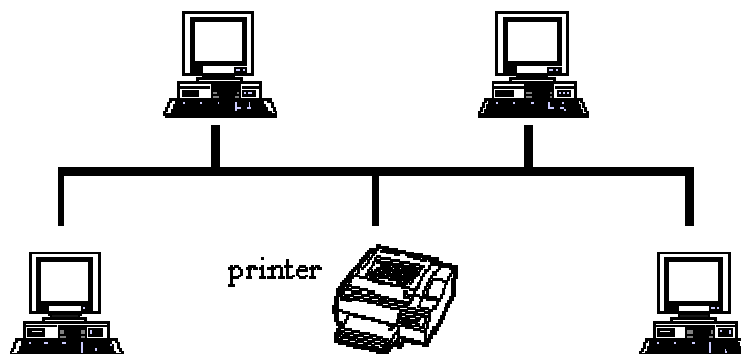
A legegyszerűbb és legolcsóbb hálózati elrendezés a sín vagy busz topológia. Ez az elrendezés egyetlen, busznak nevezett átviteli közeget használ. A buszon lévő mindegyik számítógépnek egyedi címe van, ez azonosítja a hálózaton.

Egy busz topológiájú hálózat esetén a számítógépeket az esetek többségében koaxiális kábellel csatlakoztatják egymáshoz. Nem egyetlen hosszú kábel, hanem sok rövid szakaszból áll, amelyeket T-csatlakozók segítségével kötnek össze. Ezen kívül a T-csatlakozók lehetővé teszik a kábel leágazását, hogy más számítógépek is csatlakozhassanak a hálózathoz. Egy speciális

hardverelemet kell használni a kábel mindkét végének lezárásához, hogy ne verődjön vissza a buszon végighaladó jel, azaz ne jelenjen meg ismételt adatként. Ahogy az adat végighalad a buszon, mindegyik számítógép megvizsgálja, hogy eldöntse, melyik számítógépnek szól az üzenet. Az adat vizsgálata után a számítógép vagy fogadja az adatot, vagy figyelmen kívül hagyja, ha az nem neki szól.

A busz topológiával az a probléma, hogy ha a buszkábel bárhol megszakad, akkor a szakadás egyik oldalán lévő számítógépek nem csak az összeköttetést veszítik el a másik oldalon lévőekkel, hanem a szakadás következtében mindkét oldalon megszűnik a lezárás. A lezárás megszűnésének hatására a jel visszaverődik és meghamisítja a buszon lévő adatokat.

A busz topológiájú hálózatot kialakításakor korlátozott a buszhoz köthető gépek száma. Ez azért van, mert ahogy a jel a kábelben halad, egyre inkább gyengébb lesz. Ha több számítógépet csatlakoztatunk a hálózathoz, akkor használnunk kell egy jelerősítőnek (repeater) nevezett speciális hálózati eszközt, amely a busz mentén meghatározott helyeken felerősíti a jeleket. Előnye az egyszerűsége és olcsósága, hátránya viszont, hogy érzékeny a kábelhibákra. Az alábbi ábra példa egy busz topológiájú hálózatra:



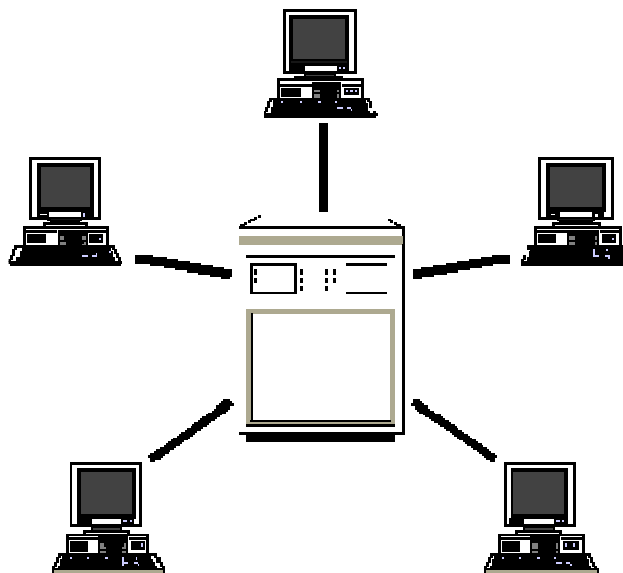
Csillag topológia

A legelső topológiák közé a csillag topológia tartozik, mivel segítségével könnyen megoldható volt már korábban is a központosított vezérlés. A csillag topológia esetén a munkaállomások közvetlenül tartanak kapcsolatot a csillag középpontjában levő szerverrel, így a központi erőforrások gyorsan és egyszerűen elérhetők. Ha nincs szükség folyamatos adatátvitelre, akkor a csomagkapcsolt eljárást alkalmazzák, különben pedig a klasszikus vonalkapcsolást. Ha az egyik számítógép kapcsolatba akar lépni a hálózat egy másik számítógépével, akkor a központi vezérlő (HUB) létrehozza az összeköttetést, vagy legalábbis kijelöli a másik berendezés elérési útvonalát,

s miután ez megtörtént, elkezdődhet a kommunikáció. Az összeköttetést követően az információcsere úgy bonyolódik le, mintha közvetlen kapcsolatban állna egymással a két számítógép. Ekkor a központi vezérőnek már nincs feladata, tehát mintegy közvetítőként működik.

A csillag topológia esetén az adatcsomagok az egyes csatlakozási pontoktól a központi HUB felé haladnak. A központi HUB az adatcsomagokat rendeltetési helyük felé továbbítja. Egy HUB-ot használó rendszerben nincs közvetlen összeköttetés a számítógépek között, hanem az összes számítógép a HUB-on keresztül kapcsolódik egymáshoz. Mindegyik gép külön kábelen csatlakozik a HUB-hoz, ezért meglehetősen sok hálózati kábelre van szükség, ami adott esetben drágává teheti a telepítést.

A csillag topológia legfőbb előnye, hogy ha megszakad a kapcsolat a HUB és bármelyik számítógép között, az nem befolyásolja a hálózat többi csomópontját, mert mindnek megvan a saját összeköttetése a HUB-bal. A topológia hátránya, hogy a központ meghibásodásával az egész hálózat működésképtelenné válik. Másik hátránya, hogy ha az egyik gép üzen a másiknak, előbb a központi gép kapja meg a csomagot, majd azt a célállomásnak továbbítja. Emiatt a központi gép gyakran túlterhelt. Az alábbi ábra példa egy csillag topológiájú hálózatra



Gyűrű topológia

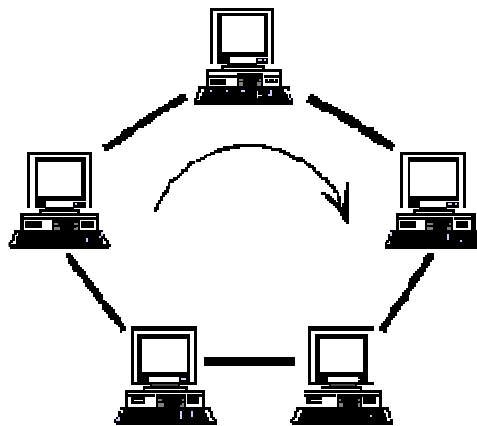
Minden állomás, beleértve a szervert is, két szomszédos állomással áll közvetlen kapcsolatban. Az összeköttetés körkörös, folyamatos gyűrű (megszakítás nélküli, de szükségszerűen kört

képező), ebből következően a hálózatnak nincs végcsatlakozása. Bármely pontról elindulva végül visszatérünk a kiindulóponthoz, hiszen az adat csak egy irányban halad.

Az üzeneteket a gépek mindig a szomszédjuknak adják át, s ha az nem a szomszédnak szól, akkor az is továbbítja. Addig vándorol az üzenet gépről gépre, amíg el nem érkezik a címzetthez. Mindegyik csomópont veszi az adatjelet, elemzi az adatokat, és ha az üzenet másik gép részére szól, akkor az adatokat a gyűrű mentén a következő géphez továbbítja. Az adatfeldolgozás cím alapján történik, azaz csak a címzett dolgozza fel az adatot, a többiek csak továbbítják.

A csillag topológiától eltérően a gyűrű topológia folyamatos útvonalat igényel a hálózat összes számítógépe között. A gyűrű bármely részén fellépő meghibásodás hatására a teljes adatátvitel leáll. A hálózattervezők a meghibásodások ellen néha tartalék útvonalak kialakításával védekeznek. Ezen kívül hátránya még az is, hogy az adat a hálózat minden számítógépén keresztülhalad, és a felhasználók illetéktelenül is hozzájuthatnak az adatokhoz.

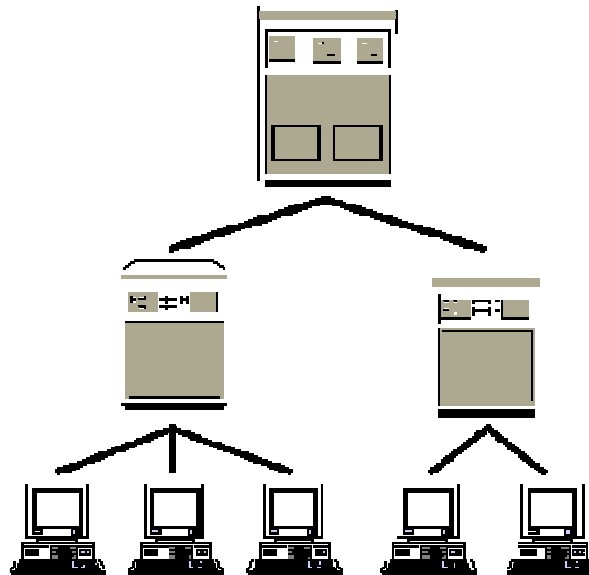
A gyűrű alakú topológia esetén a hálózati kommunikáció lehet csomagkapcsolt és vezérjel (Token Ring) elve alapján működő. Ez utóbbi esetben egy vezérjel kering körbe a vonalon, és csak az a gép küldhet üzenetet, amelynél éppen a vezérjel van. A küldő gép csak az üzenetküldés után továbbítja a vezérjelet. Az alábbi ábra példa egy gyűrű topológiájú hálózatra



Fa topológia

A fa topológiájú hálózat jellemzője a központi, kiemelt szerepkört betöltő számítógép. A központi gép úgynevezett közvetítő gépekkel vagy munkaállomásokkal van összekötve. Van egy gyökér, amelyre rákapcsolódnak a kisebb központok, a kisebb központokra kapcsolódnak a kliens gépek vagy még kisebb szerverek. Tehát a munkaállomások hierarchikus rendben kapcsolódnak

egy vagy több másik munkaállomáshoz. Egy-egy ilyen ágat alhálózatnak is nevezünk. Minden összekötött gép között csak egyetlen út van. Előnye a kis kábelezési költség, valamint, hogy nagyobb hálózatok is kialakíthatók. Hátránya viszont, hogy egy kábel kiesése egy egész alhálózatot tönkreteszhet. Az alábbi ábra példa egy fa topológiájú hálózatra



A hálózatokat csoportosíthatjuk **térbeli kiterjedésük szerint**. Ilyen értelemben megkülönböztetünk

- helyi hálózatokat (LAN - Local Area Network), a néhány méteres kiterjedéstől az általában egy épületen belüli méretekig;
- városi hálózatokat (MAN - Metropolitan Area Network), amelyek több LAN-t köt(het)nek össze;
- nagyterjedésű hálózatokat (WAN - Wide Area Network), melyek mérete az 50 km-t meghaladja és a vezetékes összeköttetést felvált(hat)ja a rádiós vagy mikrohullámú adatátvitel.

Az adatok továbbítása a fizikai hálózaton **protokollok** szerint történik. Az **ISO** (International Standards Organization) - **OSI** (Open System Interconnection) modell logikailag hét rétegre bontja a hálózatot.

1. A **fizikai réteg** (physical layer) a bitek kommunikációs csatornára való kibocsátásáért felelős. Biztosítani kell, hogy az adó által küldött jeleket a vevő is azonosként értelmezze.
2. Az **adatkapcsolati réteg** (data link layer) alapvető feladata a hibamentes átvitel biztosítása a szomszéd gépek között, vagyis a hibás, zavart, tetszőlegesen kezdetleges átviteli vonalat hibamentessé transzformálja az összeköttetés fennállása alatt. Az adatokat adatkeretekké (data frame) tördeli, továbbítja, a nyugtát fogadja, hibajavítást és forgalomszabályozást végez.
3. A **hálózati réteg** (network layer) a kommunikációs alhálózatok működését vezérli, feladata az útvonalválasztás a forrás és célállomás között. Ha az útvonalban eltérő hálózatok vannak, akkor fragmentálást, protokoll átalakítást is végez. Az utolsó olyan réteg, amely ismeri a hálózat topológiáját.
4. A **szállítási réteg** (transport layer) feladata a végpontok közötti hibamentes adatátvitel biztosítása. Már nem ismeri a topológiát, csak a két végpontban van rá szükség. Feladata az összeköttetések felépítése, bontása, csomagok sorrendbe állítása.
5. A **viszonyréteg** (session layer) lehetővé teszi, hogy két számítógép felhasználói kapcsolatot létesítsenek egymással. Jellegzetes feladata a logikai kapcsolat felépítése és bontása, a párbeszéd szervezése. Ellenőrzési pontok beépítésével szinkronizációs feladatokat is ellát.
6. A **megjelenítési réteg** (presentation layer) az egyetlen olyan réteg, amely megváltoztathatja az üzenet tartalmát. Tömörít, rejtjelez (adattvédelem és adatbiztonság miatt), kódcserét (pl.: ASCII - EBCDIC) végez el.
7. Az **alkalmazási réteg** (application layer) a felhasználóknak nyújt közvetlen szolgáltatásokat. LAN-ok esetén ezt a munkaállomásokra és a szerverre telepített hálózati alkalmazások biztosítják.

Vezetékes adatátviteli közegek

Csavart érpár

A **csavart vagy sodrott érpár** (Unshielded Twisted Pair - UTP) két szigetelt, egymásra spirálisan felcsavart rézvezeték. Ha ezt kívülről egy fémszövet burokkal is körbe vesszük, árnyékolt csavart érpárról (Shielded Twisted Pair - STP) beszélünk. Régi és ma is kedvelt módszer. A sodrott érpár két szigetelt, általában 1 mm vastag rézhuzalból áll. A két eret spirálisan összetekerik, hogy az egymás mellett lévő erek villamos kölcsönhatását kiküszöböljék. Minél sűrűbb a csavarás, annál nagyobb az átviteli sebesség. Általában több csavart érpárt fognak össze közös védőburkolatban. A sodrott érpár alkalmas analóg és digitális jelek átvitelére egyaránt. Jelerősítés nélkül is több kilométer távolságra használhatók, nagyobb távolságok áthidalására ismétlődőket vesznek igénybe. A sávszélesség függ a távolságtól, a csavarástól és a huzal vastagságától, 10 Mbit/s-tól 100Mbit/s-ig terjedhet.

Koaxiális kábelek

Széles körben két fajtáját használják: az egyik az alapsávú, melyet digitális jelátvitelre, a másik a szélessávú **koaxiális kábel**, melyet az analóg jelek átvitelére használnak. A kábel közepe tömör rézhuzalmag, amely körül szigetelő van. A szigetelőt sűrűn szőtt, hengeres vezető veszi körül. Végül az egészet egy műanyag védőburkolat öleli körül. A kábel szerkezete nagy sávszélességet, kitűnő zajvédelmet eredményez. A sávszélesség a kábel hosszától függ. Egy kilométer távolságon 10 Mbit/s, kisebb távolságokon nagyobb átviteli sebesség valósítható meg.

Üvegszálas kábel

Az optikai kutatások eredményeképpen lehetővé vált az adatok fényimpulzusokkal történő átvitele üvegszálas kábelrel. A fényimpulzus a logikai 1, az impulzus hiánya a logikai 0. A látható fény frekvenciája közel 108 MHz, így egy optikai adatátviteli rendszer sávszélessége potenciálisan óriási. A rendszer három összetevőből áll: fényforrás, átviteli közeg, fényérzékelő. Az átvitel közeg hajszálvékony, üvegből vagy szilikátból készült szál. A fényforrás vagy LED (Light Emitting Diode - fényemittáló dióda) vagy lézerdíóda, mely villamos áram hatására fényimpulzusokat bocsát ki magából. A fényérzékelő egy fotodíóda, amely fény hatására villamos jeleket állít elő. Az optikai szál egyik végére LED-et vagy lézerdíódát, a másik végére fotodíódát téve adatátviteli rendszerhez jutunk. A jelenleg kapható optikai szálak 1 km-es

távolságon 1000 Mbit/s átviteli sebességet érnek el. A nagyobb teljesítményű lézerek 100 km-es távolságot képesek áthidalni erősítés nélkül.

Vezeték nélküli adatátviteli közegek

Bár a legtöbb kommunikációs rendszer rézhuzalt vagy optikai szálát használ átviteli közegként, vannak olyanok is, melyek vezeték nélküliek, melyek a levegőt használják. Alkalmazásuk különböző helyzetekben történik. Adódhat olyan helyzet, amikor kábelek, optikai szálak elhelyezése csak utcák feltörésével, árkok ásásával volna lehetséges. Ez nemcsak költséges, egyes esetekben lehetetlen is. Másfelől a vezeték nélküli hálózatok teszik lehetővé, hogy a mozgó munkaállomások is kapcsolatba léphessenek a hálózattal.

Infravörös, lézer átvitel

A lézer és infravörös fényt alkalmazó adó-vevő párok könnyen telepíthetők magas épületek tetejére. Teljesen digitális kommunikációt tesz lehetővé, nagyobb távolságokon is lehetséges energiakoncentráció védetté teszi a külső lehallgatás ellen. Sajnos a légköri zavarok (eső, köd, por) az átvitelt zavarják.

Rádióhullám

Nagyobb távolságok áthidalására gyakran használják a mikrohullámú átvitelt. A kiemelkedő adó- és vevőantennák egymásnak sugárnyalábokat küldenek, melyek több száz kilométert is átfoghatnak. A jelismétlést reléző állomásokkal oldják meg, a vett jelet egy más frekvencián sugározzák a következő állomás felé. A rossz légköri viszonyok itt is problémaként jelentkeznek.

Szórt spektrumú sugárzás

Lokális hálózatoknál, 1 km távolságig használható megoldás. Más vevők az adást fehér zajnak (azonos amplitúdó minden frekvencián) észlelik, a vevő viszont felismeri és érti az adást.

Műhold

A távközlési műholdakat nagy, világűrben lévő mikrohullámú ismétlőknek foghatjuk fel. A frekvencia spektrumnak csak egy részét figyelik, felerősítik a vett jeleket, és a beérkező hullámokkal való interferencia elkerülése érdekében más frekvencián adják azokat újra. Hogy a földön lévő antennákat ne kelljen mozgatni, geostacionárius pályára állított műholdakat használnak. Az egyenlítő felett 36000 km magasan keringő műholdak sebessége megegyezik a

Föld forgási sebességével, így állónak látszanak. Átlagos sávszélesség: 12 db 50 MB/s-os transzponder, vagy 800 db 64 kbit/s-os hangcsatorna.

Adatátviteli vezérlő egységek

Repeater

A **repeater** (jelismétlő) helyi hálózat egy szegmenséről kapott jeleket újrarendezi és felerősíti, majd továbbítja a következő szegmensnek.

Bridge

A **bridge** (híd) két azonos típusú hálózat összekapcsolására szolgál adatkapcsolati szinten. A felhasználó nem érzékeli a jelenlétét. Egy kommunikációs számítógép, mely egymásba alakítja az eltérő keretformátumokat.

Router

A **router** (forgalomirányító vagy útvonalválasztó) kapcsolat hálózati szinten jön létre. Az összekötendő hálózatok különböző hálózati, de azonos szállítási réteggel rendelkeznek.

Gateway

A **gateway** (átjáró) különböző szoftver és hardver elemű összekapcsolódó hálózatok közötti kapcsolatot hozza létre felhasználói szinten, jelenléte a felhasználó számára is érzékelhető lehet.

Az internet

Az internet sok millió számítógép világméretű hálózata. Gerincét az a több százezer nagy teljesítményű számítógép, szerver alkotja, amelyeket műhold, üvegszál, mikrohullám és egyéb adatátviteli technikák kapcsolnak össze egymással. Ezekhez a gépekhez közvetlenül vagy internetszolgáltatókon keresztül csatlakoznak az egyes felhasználók, a kliensek. A hálózati technológiának köszönhetően bármely kliens - a szervereket összekötő gerinchálózaton keresztül - el tudja érni a hálózat bármely másik pontját függetlenül attól, hogy az egymással kapcsolatba került számítógépek fizikailag hol helyezkednek el. A gépek és a rajtuk tárolt anyagok hihetetlenül sokfélék.

Az internet története

A történet az Egyesült Államokban kezdődött az 1960-as évek elején, a hidegháborús kutatások keretén belül. A RAND Corporation foglalkozott azzal a stratégiai problémával, hogyan lehetne létrehozni egy olyan információs struktúrát, amelynek segítségével az amerikai állam és hadvezetés központjai és alközpontjai egy esetleges atomtámadás esetén is fenn tudják tartani egymással a kapcsolatot, vagyis Amerika szervezett és irányítható maradjon.

Abból indultak ki, hogy egy országos információs és irányító hálózat egyetlen központja elsődleges célpontja lenne a támadásnak, tehát azonnal megsemmisülne. A megoldás a decentralizáció. Olyan rendszert kell tehát létrehozni, amelynek nincs egyetlen kitüntetett központja, hanem eleve kis alegységek formájában működik. Fontos követelmény, hogy a keletkező struktúra szabadon konfigurálható legyen abban az értelemben, hogy új csomópontok felvétele, illetve eltávolítása egyszerűen elvégezhető műveletek legyenek, de akár néhány csomópont megsemmisülése se legyen katasztrofális a rendszer egésze szempontjából.

A csomópontok az alapegységek, melyek tökéletesen egyenrangúak. Szabadon küldhetnek, fogadhatnak és továbbíthatnak adatokat az összes többi felé. Az üzeneteket a küldő csomópont kicsi, megcímzett csomagokra bontja, melyeket a fogadó állít újra össze. Ezek a csomagok nem feltétlenül egyazon útvonalon közlekednek, csupán a végcéljuk azonos.

A 60-as évek második felében a RAND Corporation, a Massachusetts Institute of Technology (MIT) és a University of California at Los Angeles (UCLA) kísérletezett a csomagokra bontott

információ átviteli módszereinek a kifejlesztésével. Az első próbahálózatot a National Physical Laboratory brit intézet hozta létre 1968-ban. Majd a Pentagon hatáskörébe tartozó Advanced Research Project Agency (ARPA) is bekapcsolódott a kísérletekbe. Az ARPA kutatói olyan rendszert képzeltek el, amelynek csomópontjait nagyteljesítményű szuperszámítógépek alkotják. A tervezés és kivitelezés során gondoltak arra, hogy ez a hálózat békeidőben is kitűnő lehetőséget teremthet egymástól távol eső erőforrások elérésére.

- Az első hálózatot 1969 őszén építették ki az UCLA-n négy csomópontból, melyet ARPANET-nek neveztek el.
- 1971-ig 15-re nőtt a bekapcsolt helyek száma. A kutatók szélesítették a felhasználás körét: ekkor kezdett megjelenni az elektronikus levelezés.
- 1973-ban fejlesztették ki a hálózati protolloknak nevezett kommunikációs szabványokat, melyek lehetővé tették a bővítést, újabb gépek bekapcsolását. A kezdetben néhány gépet összekötő zárt rendszerből a bővítés lehetőségét magában hordozó nyílt rendszer lett.
- A földrajzi terjeszkedés innentől kezdve egyre gyorsult. Az 1980-as évekre az USA minden egyeteme rácsatlakoztatta helyi számítógépeit az immár országos méretű hálózatra.
- A 80-as évek második felében Nyugat-Európában indult meg a bekapcsolódott gépek számának növekedése, a 90-es évekre ez a hullám elérte Kelet-Európát, köztük Magyarországot is.

Csomagkapcsolt átvitel

A csomagkapcsolt átvitel lényege, hogy az adatok csomagokra bontva jutnak el egyik gépről a másikra. A csomagok a továbbítandó információ feldarabolásával keletkeznek, amit fejléccel látunk el. A fejléc az útvonalinformációt, a prioritást, a csomag sorszámát, a hibajavítás információit és egyéb járulékos információt tartalmaz. A vevőoldalon a fejléct leválasztjuk, majd a csomagokból visszaállítjuk az eredeti információt.

RFC

Az RFC a Request For Comment rövidítése. Az RFC dokumentumok az internet protollok és alkalmazások szabványgyűjteménye. Minden egyes RFC-nek van egy száma, amely az RFC-t azonosítja.

TCP/IP

A TCP/IP egy protokollkészlet, amelyet arra dolgoztak ki, hogy hálózatba kapcsolt számítógépek megoszthassák egymás között az erőforrásaikat. Mivel a protokollok közül a TCP (Transmission Control Protocol - átvitelvezőlő protokoll) és az IP (Internet Protocol - hálózatok közötti protokoll) a legismertebb, ezért az egész családra a TCP/IP kifejezést használják. A TCP végzi az üzenetek szétbontását, összeállítását, az elvesztett részek újraadását, a helyes sorrend visszaállítását. Az IP az üzenetsomagok továbbítására szolgál.

Fontosabb szolgáltatások

Elektronikus levelezés

Az **elektronikus leveleket** az angol „electronic mail” kifejezésből **e-mail**-nek szokás nevezni. Az e-mail a legrégebbi, a legalapvetőbb az internet szolgáltatásai közül. Levelezőprogramokkal üzenetek küldhetők a világ más tájaira vagy akár a szomszéd szobába, ahol a címzett néhány másodperc múlva olvashatja a levelet.

Az e-mail lényegét tekintve olyan, mint a hagyományos levelezés: mindenki, akinek internet előfizetése van, saját e-mail címmel és elektronikus postafiókkal rendelkezik. Az e-mail cím név@hol alakú. Itt a ”név” a már említett felhasználói név, a ”hol” pedig annak a számítógépnek az internet-azonosítója, amelyre az üzenetet küldjük.

A levelezést két fő protokoll irányítja: az **SMTP** (Simple Mail Transfer Protocol) a feladó oldalon, míg a **POP** (Post Office Protocol) a fogadó oldalon.

Távoli gépre bejelentkezés

A számítógéphálózatok kialakulásának jelentős előnye, hogy saját számítógépünkről távoli gépeket ugyanúgy elérhetünk, mintha annak egyik terminálja előtt ülnénk. Az interneten a távoli bejelentkezésre a **TELNET** program szolgál, amely egyben annak az alkalmazási protokollnak a neve is, amely a helyi és a távoli gép közötti párbeszédet megteremti.

A TELNET azonban elavult, ma már annak minden funkcióját megvalósító, de adattitkosítást és tömörítést is tartalmazó, fejlettebb **SSH**-t (Secure Shell) használják erre a célra. Előfordulhat például, hogy közbülső gépek lehallgathatják az adatforgalmat, beleértve a jelszavakat is, amit a TELNET nem titkosít, de sok egyéb módszer is létezik a jogosulatlan hozzáférés

megszerzéséhez. Az SSH megakadályozza ezt úgy, hogy titkosít minden adatforgalmat a terminál és a szerver között.

Adatállományok átvitele

Az állománytovábbítási protokoll az **FTP** (File Transfer Protocol), aminek szolgáltatása az állományok mozgatása egyik számítógépről a másikra, függetlenül a számítógép típusától, földrajzi elhelyezkedésétől.

Az FTP helyekhez kétféle módon lehet hozzájutni: teljes hozzáférési joggal vagy korlátozott, úgynevezett anonymous hozzáférési joggal. Ha van azonosítónk olyan számítógépen, amelyen az FTP szerver fut, akkor a teljes hozzáférési jogú FTP-t használhatjuk. Az anonymous FTP az állományok nyilvános elérését korlátozott hozzáférési joggal teszi lehetővé. Bejelentkezési névként hagyományosan az anonymous-t használjuk, s jelszóként az e-mail, azaz az elektromos levelezési cím vált elterjedté.

Az interneten folyamatosan hatalmas mennyiségű információ halmozódott fel, és a rendelkezésre álló állományok nevééről és méretéről nem történt semmilyen központi nyilvántartás. A felhasználók munkájának megkönnyítésére fejlesztették az **Archie** nevű indexelő programot, ami az FTP helyeken található állományokból indexelt adatbázist hoz létre. Az így létrejött adatbázis az állomány nevét, méretét, típusát, egyéb állományinformációt tartalmaz. Ennek segítségével lényegesen gyorsabban megtalálható a keresett állomány az FTP szervereken.

A TELNET esetében ismertetett okokból az FTP helyett ma már az **SFTP**-t (Secure FTP) használják.

World Wide Web

A **World Wide Web** (WWW vagy Web) az elektronikus levelezés után az internet legfontosabb alkalmazása, amely az egész világot behálózó információkezelő rendszer. A Web egymással kapcsolatban álló dokumentumok millióinak gyűjteménye, melyeket a világ legkülönbözőbb helyein lévő számítógépek tárolnak.

A Web 1989 márciusában született meg. Megalkotója Tim Berners-Lee, aki a genfi székhelyű Európai Részecskegyorsító Intézetben (CERN) dolgozott, információkat akart megosztani a kutatásban résztvevő, földrajzilag egymástól távoli kutatók között. A CERN támogatta a Web létrehozását, és világméretűvé bővítette. Első nyilvános használatára 1992 januárjában került sor.

A kommunikáció a **hypertext** technológia alkalmazására épül. Az információkat a böngésző programok segítségével jeleníthetjük meg. A böngészők a webserverekkel **HTTP** protokollon keresztül kommunikálnak. A HTTP segítségével a böngészők adatokat küldhetnek a szervereknek, valamint weblapokat tölthetnek le róluk.

A lapokat a böngésző az **URL** segítségével találja meg, mely a lap címét jelöli. Az URL a címhez tartozó protokollal kezdődik, például a http: a HTTP protokoll jelölése. Sok böngésző több más protokollt is támogat, mint például az ftp: az FTP.

A weblaphoz tartozó fájl formátuma többnyire **HTML** (HyperText Markup Language). A HTML a böngészőkkel együtt fejlődött, a „hivatalos” HTML-változatokat a W3C (World Wide Web Consortium, mely nyílt szoftver szabványokat alkot a világhálóra) hagyta jóvá, illetve készítette el. A böngészők sokfélesége és a cégek saját HTML módosításai kompatibilitási problémákhoz vezettek.

A szoftver

Az előző fejezetekben már tárgyaltuk a számítógép hardverét, de magát a fogalmat még nem határoztuk meg. A **hardver** (hardware) a számítógépet alkotó mechanikus és elektronikus alkatrészek összessége.

A számítógéphez tartozó másik összefoglaló fogalom a **szoftver** (software), ami a hardver elemeinek működtetését végző programok, a gép használatához szükséges szellemi termékek összessége. A szoftver teszi használhatóvá a számítógépet, mert a hardver önmagában nem működőképes. Szükség van programokra, amelyek biztosítják a számítógép egységeinek összehangolt működését és a felhasználó igényeinek lehető leghatékonyabb kielégítését.

A szoftverek csoportosítása

A szoftvertermékeket attól függően, hogy milyen feladatokat látnak el, illetve milyen számítógépes rendszerekhez, környezethez készülnek, többféle szempont szerint lehet csoportosítani. Az osztályozásra egységes elvet nehéz meghatározni, de a legtöbb szakirodalomban az alábbi csoportosítást találjuk.

Rendszer- és rendszerközeli szoftverek

A **rendszer- és rendszerközeli szoftvereket** jellemzően a számítógépet gyártó cégtől vagy szoftverfejlesztőtől vásároljuk meg. Ezek a szoftverek biztosítják a számítógép összehangolt vezérlését, megkönnyítik az operációs rendszerek használatát, illetve segítik a programfejlesztést.

BIOS

A **BIOS** (Basic Input/Output System) lényegében egy rendszerprogram, amelynek a segítségével a programok szabványos módon tudnak kommunikálni a ki- és bemeneti eszközökön keresztül. A BIOS a számítógép ROM típusú memóriájában található.

A BIOS feladata ma már két pontban foglalható össze. Az egyik a számítógép indításához kapcsolódó beállítások és ellenőrzések elvégzése. A BIOS ekkor vizsgálja meg, hogy milyen eszközeink, milyen típusú processzorunk és mennyi memóriánk van, illetve beállítja ezek alap működési módját, majd mindegyik hardverelemen elvégez egy rövid tesztet. Ezt a folyamatot egységesen POST-nak (Power On Self Test) hívjuk.

A BIOS másik fontos feladata az operációs rendszer behúzójának, a merevlemez Master Boot Recordjának betöltése a memóriába, majd a vezérlés átadása erre a kódrészletre. A merevlemezről behúzott kód, az operációs rendszer betöltője, hamarosan teljesen átveszi az irányítást, és saját meghajtó programjaira váltva már a BIOS nélkül fut tovább.

Operációs rendszer

Az **operációs rendszer**, a hozzátartozó segédprogramokkal és könyvtárakkal, olyan szoftver, amely a számítógép működtetéséhez szükséges parancsokat értelmezi, és végre is hajtja.

Programfejlesztő rendszerek

A **programfejlesztő rendszerek** olyan rendszerközeli szoftverek, amelyek lehetővé teszik a felhasználók igényeinek megfelelő programok készítését, fordítását a gép által közvetlenül végrehajtható utasításokra, valamint e programoknak vagy részeinek szerkesztését és ellenőrzését.

Felhasználói vagy alkalmazói szoftverek

A **felhasználói vagy alkalmazói szoftvereket** a számítógépet üzemben tartó külön vesz meg, fejleszt ki, vagy dolgoztat ki saját feladatainak megvalósításához.

Standard felhasználói szoftverek

A **standard felhasználói szoftvereket** általános, sok helyen előforduló feladatok megoldásához készítene (szövegszerkesztők, táblázatkezelők, adatbáziskezelők, tudományos szubrutinyűjtemények, termelésirányítási rendszerek, stb.).

Egyedi felhasználói szoftvereket

Az **egyedi felhasználói programok** egy vállalat, intézmény vagy magánszemély számára készült szoftvertermékek, ahol jobban érvényesíthetők az egyéni igények, elvárások. Az ilyen programokat legtöbbször kisebb szoftvercégek vagy önálló szakemberek fejlesztik. Nagyelőnyük a módosíthatóság, igények szerinti átalakíthatóság.

Az egyedi szoftverfejlesztés lépései:

- A megoldandó probléma meghatározása, elemzése: az igények meghatározása.
- Tervezés, modellezés: specifikáció.
- Kivitelezés: programozás, folyamatos dokumentálás.

- Ellenőrzés, tesztelés, szükség szerinti módosítás.
- Installáció, átadás: felhasználó kézikönyv elkészítése.

Szoftverek osztályozása kereskedelmi szempontból

Vásárolt szoftver

Valamilyen adathordozón példányonként, többnyire egyedi vagy személyes felhasználásra megvett szoftver.

Szoftvercsomag

Több felhasználót foglalkoztató vállalatok, intézmények számára kedvező áron beszerezhető programok összessége. Az ár rendszerint függ a felhasználók számától, illetve összetételétől. (Például a különböző szoftvergyártó cégek az oktatási intézmények számára rendszeresen állítanak össze kedvezményes árú programcsomagokat.)

Shareware szoftver

Egy adott időintervallumban ingyenesen kipróbálható szoftver. A meghatározott idő letelte után a felhasználó vagy megvásárolja a programot, vagy nem tudja tovább használni.

Freeware szoftver

Többnyire az internetről ingyenesen letölthető szoftver, amelyet nincs jogunk kereskedelmi céllal tovább adni, más szoftverbe beépíteni vagy módosítani.

Public-domain szoftver

Teljesen ingyenes, szabadon másolható és felhasználható szoftverek. Általában a fejlesztésének korai fázisaiban minden szoftver ilyen, hiszen a kezdeti verziókat érdemes ingyenesen hozzáférhetővé tenni. Egy szoftver public domain volta nem jár együtt a forráskódjának nyilvánossá tételével, és a szerzők bármikor úgy dönthetnek, hogy egy adott verziótól kezdve a szoftver már ne legyen ingyenes.

Az operációs rendszer

Kezdetben a számítógépek megépítése, a hardverelemek működésének biztosítása volt az elsődleges cél. Miután a számítógépek biztonságosan működtek, egyre nagyobb volt az igény a hatékonyság növelésére. Olyan, egymással jól együttműködő programokat (rendszereszoftver) fejlesztettek ki, amelyek felügyelik az egyes hardveregységeket, összehangolják működésüket, biztosítják a számítógép erőforrásainak hatékony kihasználását, és segítik a programok végrehajtását. Ezeket **operációs rendszereknek** nevezték.

Az operációs rendszer feladata, hogy felhasználóbarát módon elégítse ki a felhasználó és a számítógép közötti kapcsolatot, lássa el a felhasználói programok kezelését, futtatását, vezérlését, illetve gondoskodik a számítógép erőforrásainak a különböző programok közötti hatékony elosztásáról.

Az operációs rendszerek sok tekintetben különbözhetnek egymástól, és számos szempont szerint csoportosíthatók. Egy általánosan elfogadott osztályozás különbséget tesz **gyártóspecifikus** és **nyílt** operációs rendszerek között. Kezdetben egy adott számítógépcsaldra hoztak létre speciális operációs rendszereket, ma már a különböző hardverekre telepíthető operációs rendszerek a jellemzőek.

Az osztályozás további lehetséges szempontjai lehetnek:

- a hardver mérete (nagy-, kis- és mikrogépes);
- a felhasználók száma (egy- vagy többfelhasználós);
- a multiprogramozás foka (egy- vagy többprogramozható);
- az elérés módja (köteget, interaktív és valós idejű);
- a rendszer struktúrája (centralizált, elosztott vagy hálózati);
- a kommunikáció módja (utasításvezérelt, menüvezérelt, grafikus).

A számítógépes rendszer hatékonyságának biztosítására az operációs rendszerek különböző technikákat alkalmaznak.

A megszakítás (interrupt) kezelése

A megszakítások a számítógép munkájának összehangolásában játszanak fontos szerepet. A megszakítási rendszer a folyamatok közben keletkező események feldolgozására szolgál. Ezen

események lehetnek szinkron jellegűek, melyek keletkezése egy program futása közben meghatározható helyen és időpontban várható (például túlcsordulás), aszinkron események, melyek várhatóak, de időpontjuk előre nem ismert (például adatbeolvasás), valamint váratlan aszinkron események, amelyek keletkezése nem várható (ilyen lehet egy hardverhiba). A megszakítási kérelem jelzi a processzornak valamely esemény bekövetkeztét, amely egy kiszolgáló folyamatot indít el egy későbbi időpontban (megszakítás időpontja). A megszakítás tulajdonképpen a futó folyamat felfüggesztése annak kiszolgálása céljából.

Különbséget kell tenni a külső eredetű megszakítások (interrupt) és az utasítások végrehajtását megállító kivételek (exception) között. Míg az elsőnél a processzor a végrehajtás alatt levő utasítást befejezve kiszolgálja a megszakítást, majd folytatja a feldolgozást a következő utasítással, addig kivétel esetén a kiváltó esemény kiszolgálása után a processzor megkísérli a megszakított utasítást újra végrehajtani.

Megszakítások kiszolgálásánál a rendszernek meg kell állapítania a keletkezés helyét, szabályoznia a megszakítási lehetőségeket (egyes utasítások megszakításkérelmi lehetőségének maszkolása). Szükség van a prioritás szabályozására több, egy időben bekövetkező kérelem kiszolgálási sorrendjének meghatározására, valamint a kiszolgáló folyamat közben beérkező kérelmek kezelése is.

A kérelem keletkezési helyének megállapítása történhet szoftver, illetve hardver segítségével is. A szoftveres módszert lekérdezési eljárásnak nevezik. Egy program, ami általában az operációs rendszer része, meghatározott időközönként megvizsgálja az eszközök megszakítási kérelemre vonatkozó állapotjelzőjét, és ahol megszakítási igényt detektál, elindítja a kiszolgáló rutint. Ezt a módszert csak egyszerűbb esetekben alkalmazzák.

A hardveres módszer esetén egy megszakításvezérlő áramkör szabályozza a megszakítások kiszolgálását. A kérelem elfogadását visszaigazolás követi. Egy megszakítási vonal esetén a hely meghatározása úgy történik, hogy a visszaigazoló jel a kiszolgálást kérő eszköztől már nem halad tovább, és ez elindítja a kiszolgáló rutint. Több megszakítási vonal esetén pedig a kérelem helye egyértelműen megállapítható (minden eszköznek saját vezetéke van). A legáltalánosabb hardveres módszer a vektoros, ahol a kérelmező eszköz a kiszolgáló rutin címét határozza meg a vezérlő és a processzor számára. Lehetőségek:

- az eszköz a kiszolgáló rutint elindító hívó utasítást átadja a processzornak;

- az eszköz a kiszolgáló rutint elindító hívó utasítás tárolóhelybeli címét adja át a processzornak;
- az eszköz a kiszolgáló rutinnak a kezdőcímét adja át a processzornak;
- a leggyakrabban alkalmazott módszer, amikor az eszköz az őt kiszolgáló rutin sorszámát adja át, amely alapján a processzor a megszakítási vektortáblából kikeresi a kiszolgáló rutin kezdőcímét. (Ennek létezik egy úgynevezett autóvektoros változata, ahol a vektortáblázatot a processzor a belső táblázatában tárolja.)

A kiszolgálási eljárást a processzor indítja el, amely az alábbi lépésekből áll:

- Az eszközvezérlő megszakítást kér a processzortól.
- Az aktuális gépi ciklus befejezésekor a processzor nyugtázza a kérést.
- A nyugtázás után az eszközvezérlő kiadja a saját megszakítási vektorát.
- A processzor fogadja azt és elmenti.
- A processzor elmenti a programszámlálót és a legfontosabb regisztereket a verembe.
- A processzor megkeresi a megszakítási vektorhoz tartozó kiszolgáló rutint.
- A processzor lefuttatja a megszakítási rutint, melynek megszakítását csak magasabb prioritású esemény számára engedélyezi.
- A megszakítási rutin végrehajtása után a processzor visszaállítja a használt regisztereket és végrehajtja a „visszatérés a megszakításból” utasítást.
- A processzor visszaolvassa a veremből a mentett regisztereket a programszámlálóval együtt, és a program a megszakítást megelőző állapotba kerül.

A megszakítási kérelmeket általában prioritási elv felhasználásával szolgálják ki. Többszintű megszakítási rendszerek esetében a kiszolgáló rutin is megszakítható. Ekkor a kiszolgáló rutin:

- a vele egyező vagy nála alacsonyabb prioritású kérelmeket letiltja;
- ideiglenesen alacsonyabb prioritású szintre lép;
- a kiszolgálás idejére új prioritási szinteket rendel az egyes eszközökhöz.

Spooling technika

A spooling technika a lassú perifériák (például nyomtatók) esetén úgy küszöböli ki a központi egység tétlenségét, hogy a kivitel először egy gyorsabb háttértárra történik viszonylag rövid idő alatt, és maga a nyomtatás más feladatokkal párhuzamosan, a központi egység "hulladék

idejében" hajtódik végre. (A SPOOL egy betűszó, amely az IBM-től ered: a "Simultaneous Peripheral Operation On-Line" rövidítése.)

Perifériák ütemezése

A perifériák hatékony kihasználására a **dedikált hozzárendelés** valósítható meg, ami azt jelenti, hogy a perifériát viszonylag hosszabb időre adott programhoz rendeli az operációs rendszer.

Tárkezelési problémák

Több felhasználót kiszolgáló, illetve több feladatot egyidejűleg kezelő rendszerek esetén természetesnek tűnik, hogy a futtatandó **programoknak áthelyezhetőeknek** kell lenniük. Ezt a logikai és a fizikai címtartományok bevezetésével lehet megoldani. A kulcskérdés a kettő közötti leképezés.

Megoldható a multiprogramozás úgy is, hogy egyidejűleg csak egy feladatot tartunk a tárban, ami így az operációs rendszer által szabadon hagyott teljes területet uralhatja. Az ilyen esetben használt technikát **swapping**-nak (cserebere) nevezik. A swapping lényege, hogy feladatváltáskor a felfüggesztett feladathoz tartozó teljes tárterületet a háttértárra másolják, és onnan behozzák a következő feladatot.

Amikor a swapping-nál bevezették a több puffer használatát, eljutottak a **többpartíciós** multiprogramozáshoz. A tárat több, különböző méretű, egybefüggő részre (partíciókra) osztották, melyek mindegyikében egy végrehajtásra váró feladat foglalt helyet. Attól függően, hogy a felosztás rögzített vagy változtatható, beszélünk fix vagy változó particionálásról. Bármilyen ügyes algoritmusokat is dolgoztak ki a particionált tár kezelésére, a tár elaprózódása ellen nem lehetett hatásosan védekezni. Egy bizonyos idő elmúltával, a tárban sok, össze nem függő, apró terület alakult ki, melyeket bár együttes méretük számottevő volt, mégsem lehetett hasznosítani.

A megoldást a **virtuális tárkezelés** megvalósításával bevezetett **lapkezelés** jelentette. A tárat azonos méretű lapokra osztják. Minden feladathoz annyi -- nem szükségképpen folytonos fizikai címeken elhelyezkedő -- lapot rendelünk, amennyit igényel. A lapkezelő alrendszer fő feladata a felhasználó logikai tárigényét leképezni a fizikai lapokra.

A lapkezelés általánosítása a **szegmentálás**, ami tulajdonképpen változó méretű lapokkal folytatott gazdálkodást jelent. Bevezetésének másik oka az, hogy a lapozás néha az algoritmusokat kellemetlen ponton vágja ketté, így a programrésznek állandóan két lap között

kell ugrálnia. Egy új szegmens használatával biztosítható, hogy a kritikus rutin egy lapra kerüljön.

Processzor időbeosztás

A **processzor ütemezése** a számítógép legfontosabb erőforrásával való gazdálkodást jelent. Az alapelv az **időszeletelés** (time slicing), ami a következőt jelenti:

- a processzor idejét azonos hosszúságú időszeletekre bontják;
- egy időszelet alatt egy programon dolgozik a processzor;
- az időszelet lejártát követően az operációs rendszer dönti el, hogy melyik programhoz rendelje a processzort.

Az egyes megvalósítások abban különböznek, hogy milyen elv szerint történik a processzor hozzárendelése a következő programhoz. Néhány ezek közül:

- **egyenlő részesedés** (equal share): minden program sorra megkapja az időszeletet;
- **processzoridő-igény szerinti prioritás**: a kevesebb processzoridőt igénylő kapja a nagyobb prioritást (shortest job first), vagy a sok processzoridőt igénylő kapja meg gyakrabban a processzort (longest job first);
- **perifériaigény szerinti prioritások**: I/O igényes vagy CPU igényes feladatok;
- **fizetett**, illetve **irányított prioritások** (felhasználói account alapján).

Személyi számítógépek operációs rendszerei

A személyi számítógépek operációs rendszerei a nagyszámítógépektől elvárt funkciókkal és képességekkel rendelkeznek, sőt bizonyos értelemben a felhasználói felületek és a felhasználóbarát alkalmazói szoftverek vonatkozásában túl is mutatnak a mainframe-es rendszereken. A fejlődés iránya a klasszikus operációs rendszerek hagyományos szemléletétől egyre inkább a felhasználóbarát felületek biztosítása irányába tart, és így az operációs rendszer magja, működése valójában a felhasználó előtt ma már szinte teljesen rejtve marad.

Parancsvezérelt (DOS alapú) operációs rendszerek

A Seattle Computer Products 1979-ben fejlesztette ki az MS-DOS-t, majd a Microsoft megvásárolta a szoftver terjesztésének és továbbfejlesztésének jogát. Az IBM 1981-ben jelent meg a PC-DOS operációs rendszerrel, amely az MS-DOS egy változata. Később a Microsoft kapta meg a jogot, hogy a különböző gyártók által készített személyi számítógépek MS-DOS operációs rendszerének hivatalos szállítója legyen.

Az MS-DOS operációs rendszer egyfelhasználós, egyfeladatos üzemmódú szoftver. Az operációs rendszer programjait két csoportba sorolhatjuk: az egyik a rezidens programok, amelynek rutinjai állandóan az operatív memóriában vannak; a másik csoportba azok a rendszerprogramok tartoznak, amelyek igény szerint töltődnek be a memóriába.

Grafikus felületű operációs rendszerek

A grafikus felhasználói felületek kifejlesztésére először a 80-as évek elején az első lépéseket az Apple tette meg, és szállította a számítógépeit ilyen operációs rendszerrel. Az Apple után a Microsoft is hamarosan nyilvánosságra hozta a Windows-nak nevezett grafikus felület-szoftvert. Az első verzióknak nem volt túl nagy sikere, de az 1992-ben bejelentett 3.1-es verzió üzembiztosabb és gyorsabb működésével jobban megnyerte a felhasználók tetszését. Szigorúan véve a Windows 3.1 valójában nem egy önálló operációs rendszer, hanem egy olyan – DOS-ra épülő – grafikus felület, amely megkönnyíti a rendszer és az alkalmazások használatát.

Az IBM-mel közös OS/2-ből kiindulva 1992-ben a Microsoft egy új, saját fejlesztésű operációs rendszert Windows NT néven hozott forgalomba.

A rendszerek számtalan lehetőséget kínálnak, amelyekkel a felhasználó a számítógéppel interaktív módon, egyszerűen, könnyen kezelhetően végezheti munkáját. Az ablaktechnika lehetővé teszi, hogy az egyes funkciókra vonatkozó lehetőségeket láthassuk, egyidejűleg akár többet is. Menü funkciókat kínál fel kis szöveges ablakokban vagy gombokkal. A valós világot szimbolizáló objektumok, ikonok egyszerűsítik a kezelést, a képecskék (nyitott könyv, grafikon szimbólum, stb.) a végzendő funkcióra utalva. Ha pedig a felhasználó elakad, akkor súgók (helpek) állnak rendelkezésre és segítenek a probléma megoldásában, a hiba elhárításában.

A Microsoft a Windows 3.1 grafikus felület, valamint a Windows NT sikeréből kiindulva azt a célt tűzték ki, hogy olyan átfogó operációs rendszert fejlesszenek ki, amely egyszerűbbé teszi a személyi számítógépek használatát, segíti a hálózatok elérését és biztosítja a kompatibilitást a korábbi operációs rendszerekkel. Így születtek meg a Windows különböző verziói (95, 98, 2000, XP).

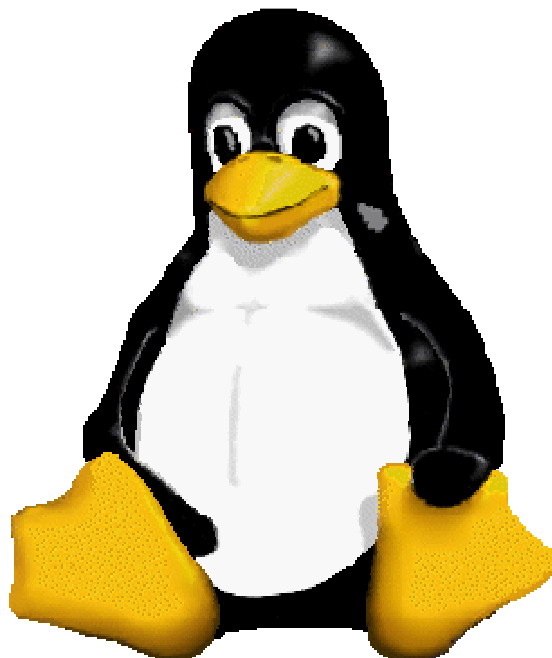
UNIX rendszerek

A számítógépek operációs rendszerei között az egyik legjelentősebb, központi szerepet betöltő operációs rendszer a UNIX. Kedvelt, sok változatban ingyenes változata a Linux. A UNIX-ot eredetileg nagyszámítógépes, hálózatos környezetre szánták, de manapság rohamosan terjed a személyi számítógépek világában is. Fő előnye a hatékony hardverkihasználás biztosítása és a különböző hardverkomponensek egységes rendszerbe illesztésének lehetősége. A UNIX volt az első olyan operációs rendszer, amelynek fejlesztése során figyelembe vették a nyílt rendszerek felépítésének alapelveit. A UNIX belső felépítésére jellemző a rétegszerkezet. Az alapfunkciókat az operációs rendszer állandóan futó magja, a kernel valósítja meg. A felhasználói interfészt a független shell biztosítja. Több párhuzamosan fejlesztett shell létezik, közös jellemzőjük a hatékony felhasználói környezet megteremtése. Multiprogramozott operációs rendszer, amely támogatja az alkalmazások párhuzamos futását is.

Személyi számítógépekre fejlesztett első UNIX-szerű operációs rendszert a MINIX volt, amit egy holland professzor, Andrew S. Tanenbaum (1944 -) fejlesztett.

A Linux fejlesztését a finn Linus Torvalds kezdte 1991-ben, aki akkor másodéves hallgató volt a Helsinki Egyetemen. A 80386 processzor védett módú (protected mode), feladat-váltó (task-switching) lehetőségeivel szeretett volna megismerkedni, és ehhez kezdett programot (rendszermagot, más néven kernelt) írni MINIX alatt, eleinte assembly-ben, majd C-ben. Az

internet révén később többen bekapcsolódtak a Linux fejlesztésébe, ami a kilencvenes évek végére egyértelműen bebizonyította a létjogosultságát szabad szoftverek kategóriájában. A képen a Linux emblémája látható:



Nagyon sok Linux disztribúció létezik, köztük több magyar fejlesztésű. Néhány az ismertebbek közül: Debian, Red Hat, SuSe.

Algoritmusok

Az **algoritmus** szó **Abu Abdalláh Muhammad ibn Musa al-Khwarizmi** taskenti bölcse (780?-850?) latin nyelvre lefordított, *Algoritmi de numerus indorum* (magyarul: al-Khwarizmi az indusok számjegyeiről) könyvének címében szereplő *algoritmi* szóból (Khwarizm városából való), nevének elferdítéséből ered.

A modern algoritmuselmélet atyjának **Alan Mathison Turing** (1912-1954) angol matematikust tekintjük, aki az 1930-as években alkotta meg és tette közzé az algoritmus matematikailag pontos fogalmát. Elkészített egy absztrakt (elméleti) gépet, a Turing-gépet, aminek segítségével algoritmuselméleti problémák szimbolikusan kezelhetők, ebben a témakörben tételek mondhatók ki és bizonyíthatók.

Az algoritmus fogalom meghatározására sok „konyhanyelvi” leírás található a különböző szótárakban, lexikonokban, irodalmakban:

- Az algoritmus legáltalánosabb értelemben nem más, mint tervszerűség. Ha egy elvégzendő cselekvéssorozatot lépésről lépésre előre átgondolunk, megtervezünk, úgy is mondhatjuk, hogy algoritmust adunk egy adott cél elérésére.
- Az algoritmus egy olyan előírás, amely alapján egy adott feladat véges számú lépésben megoldható.
- Véges számú, egymást meghatározott sorrendben követő lépésekkel a probléma megoldásához vezető módszer, eljárás.
- Műveletek tartalmát és sorrendjét meghatározó egyértelmű utasításrendszer, amely a megfelelő kiinduló adatokból a kívánt eredményre vezet.
- Egy adott feladat megoldásán azt a műveletsorozatot értjük, amellyel a kiindulási adatokból véges számú lépésben eljutunk a keresett eredményadatokhoz.

Minden meghatározásból leszűrhetők az algoritmus **tulajdonságai**:

- **egyértelmű** (az algoritmust alkotó utasítások egyértelmű, meghatározott sorrendben követik egymást);
- **determinisztikus** (ugyanazon kiindulási adatokra tetszőleges számú végrehajtás esetén ugyanazt az eredményt szolgáltatja);

- **véges** (véges számú lépés után befejeződik, és eredményre vezet);
- **redundáns lépéseket** (ismétlődéseket, felesleges utasításokat) **nem tartalmaz**;
- **teljes** (nemcsak egy konkrét esetre alkalmazható, hanem az összes azonos jellegű feladatra).

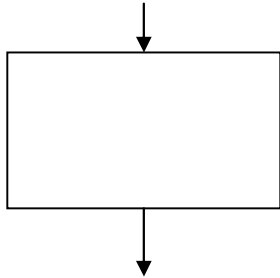
Egy algoritmus **elemi** (azonnal végrehajtható) és **összetett** (elemi tevékenységekből felépülő) műveletek sorozata. Az algoritmus struktúráját **szekvenciák**, **szelekciók**, illetve **iterációk** adják, amelyek tetszőleges mélységben egymásba skatulyázhatók.

A **szekvencia** egymás után végrehajtandó elemi tevékenységek sorozata. Lehet, hogy a megoldás bizonyos pontokon nem látható előre, és feltételektől függően más és más utat kell választanunk (szelektálunk). A **szelekció** esetén egy feltétel igaz vagy hamis voltától függ, hogy bizonyos utasítások végrehajtnak-e vagy sem. Előfordulhat, hogy a megoldás érdekében valamely tevékenységet többször is végre kell hajtani, vagy ismételni (iterálni) kell. Az **iteráció** lehetővé teszi meghatározott utasítások tetszőleges számú ismételt végrehajtását. Lehet, hogy az iterációk számát előre tudjuk, lehet, hogy az ismételt végrehajtásnak feltétele van.

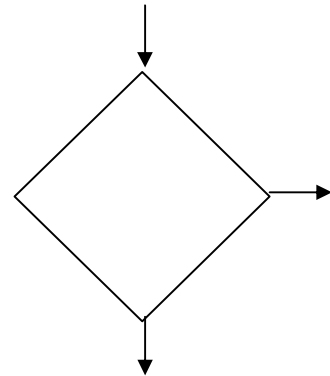
Az algoritmusok tervezésére, szemléltetésére, megadására, áttekinthető formában történő leírására sokféle eszköz létezik. A leggyakrabban használt a **flowchart**, amelynek alapelveit Neumann és Goldstine a programozás kapcsán dolgozta ki. (A flowchart szabványban meghatározott szimbólumok rendezett sorozata, amelyben az egyes elemek a probléma meghatározásának, elemzésének vagy megoldásának a résztevékenységeit - lépéseit - jelentik.) Az egyes szerkezeti elemek között **nyilakkal** jelöljük a végrehajtási irányt, sorrendet. A flowchart alapalakzatai:



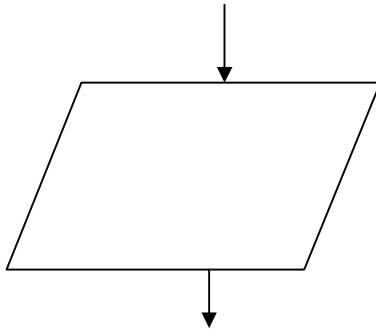
Az algoritmus kezdetét és végét jelölik.



Feltétel nélkül végrehajtható utasítást jelöl.



Feltételes utasítás jelöl.



Input/output műveletet jelöl.



A folyamatábra megszakítását és folytatását jelölik.

Egy másik eszköz a **leíró nyelv** (pszeudokód – „ál nyelv”, „mondatszerű leírás”), melynek segítségével megfogalmazott algoritmus könnyen átírható általános célú programozási nyelvre. A beszélt nyelvhez hasonló, de annál tömörebb leírási mód. Annyiban tér el a folyamatos írástól, hogy bizonyos szabályokat be kell tartanunk, a struktúrák képzésére megállapodás szerinti formákat és szavakat használunk.

Példák:

Beolvasás és kiíratás:

read (input file) változók listája;
write (output file) változók listája.

Szekvencia (pl. értékadás):

a := 5; b := 15;
 c := a+b.

Szelekció:

```
if [ha] feltétel then [akkor]
    utasítás[ok]
endif [elágazás vége]
```

```
if feltétel then
    utasítás[ok]1
else [egyébként]
    utasítás[ok]2
endif
```

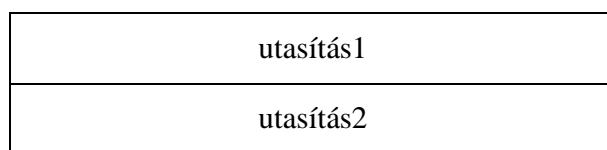
Iteráció:

```
i := 1
repeat [ismétlés]
    i := i+1
until [amíg] i > 100.
```

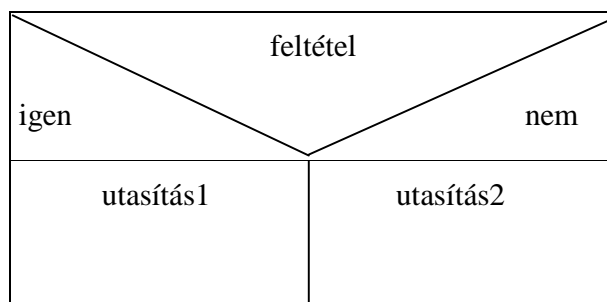
A harmadik eszköz a **struktogram** (NSD, **Nassi-Shneiderman diagram**), amelyet Ike Nassi és Ben Shneiderman vezetett be a hetvenes évek elején, a strukturált programozás algoritmusleíró eszköze. Az egyes szerkezeti elemeket téglalapba foglalható ábrákkal jelöljük. A szerkezetek egymásba ágyazhatók, de vonalaik nem keresztezhetik egymást. Az ábrát felülről lefelé haladva kell olvasni.

Példák:

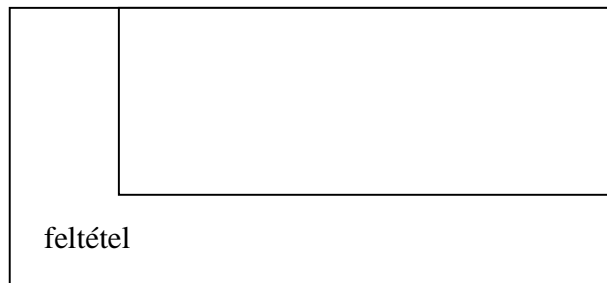
Szekvencia:



Szelekció:



Iteráció (repeat-until):



Nevezetes algoritmusok

Gyakori programozási feladat egy tömb elemeinek bizonyos szempont(ok) szerinti rendezése (esetleg eleve rendezett tömb létrehozása), illetve bizonyos feltétel(ek)nek eleget tevő tömbelem keresése, kiválasztása. Az alábbiakban néhány ilyen alapalgoritmust ismertetünk.

Kereső algoritmusok

Keresést rendezetlen és rendezett tömbökön egyaránt végezhetünk. Rendezetlen tömb esetén a keresés nehezebb, lassúbb, de az egyéb tömbkezelő műveletek (pl. új elem beszúrása) egyszerűbbek. Rendezett tömb esetén gyorsabb a keresés, viszont a kapcsolódó tömbműveletek bonyolultabbak, hiszen pl. egy új elem beszúrása után is meg kell tartani a tömb rendezettségét.

Lineáris keresés

A legegyszerűbb keresési algoritmus, amely rendezetlen tömbön dolgozik. A tömb első elemétől kezdve összehasonlítjuk a tömbelemeket a keresendő elemmel. A keresési ciklus akkor ér véget, ha valamelyik tömbelem megegyezik a keresettel, vagy, ha a tömb végére érünk. Az utóbbi esetben a keresett elem nincs a tömbben. Az algoritmus onnan kapta nevét, hogy a keresések száma, és így a futási idő, lineáris függvénye a tömb elemszámának.

Logaritmikus (bináris) keresés

A logaritmikus vagy bináris keresési algoritmus rendezett tömbön működik, így az előző módszernél lényegesen gyorsabb keresést tesz lehetővé. A keresendő elemet először a tömb középső eleméhez hasonlítjuk. Ha a két elem egyezik, megtaláltuk a tömbelemet, ha nem, a

keresést abban a tömbfélben folytatjuk, amelyet a középső és a keresett elem viszonya kijelöl. Ha a tömb növekvő sorrendbe rendezett és a keresendő elem nagyobb, mint a középső elem, akkor a második tömbrészben, egyébként pedig az elsőben folytatjuk a keresést. A keresést akkor fejezzük be, ha megtaláltuk a keresett tömbelemet, vagy a tömbrész elfogyott. Az összehasonlítások száma, s így a futási idő, az elemszám kettes alapú logaritmusával arányos, ami nagy elemszámok esetén lényegesen kisebb lehet, mint a lineáris keresés esetén.

Rendező algoritmusok

Beszűrés

Van egy n elemű rendezett tömbünk. Be akarunk szűrni egy $n+1$ -edik elemet a rendezettség megtartásával. Egyik lehetőség, hogy végigszaladunk a halmazon, és betesszük a beszűrandó elemet az elé az elem elé, ami már nem előzi meg. Általában $n/2$, de a legrosszabb esetben n összehasonlításra van szükség. Másik lehetőség, hogy a kérdéses elemet a középső elemmel hasonlítjuk össze. Ha az új elem megelőzi a középsőt, akkor a továbbiakban a halmaz első felében keressük a helyét, stb. A lépésszám $\log_2 n$, vagy az erre következő első egész szám, azaz $\lceil \log_2 n \rceil$, ha n nem 2 hatványa.

Beszűrésos rendezés

A rendezés során sorrendbeli hibát keresünk egy tömbben. Ennek során használhatunk lineáris, illetve bináris keresést. A kialakult sorrendtől eltérő helyen levő elemet kiemeljük, majd megkeressük a sorrendnek megfelelő helyét. Amikor a helyét megtaláltuk, akkor a közbeeső elemeket eltoljuk, majd az imént kiemelt elemet a felszabaduló helyre beszűrjük. Ez a rendezés túl sok mozgatót igényel.

Shell rendezés

A beszűrésos módszer lépésenként finomított változata. A tömbnek csak minden d -edik elemét vizsgáljuk, azaz d lépésközzel végezzük el a beszűrésos rendezést. Ha a rendezettség nem alakul ki, akkor csökkentjük a lépésközt, és úgy végezzük el a rendezést. A folyamat addig tart, amíg a rendezettség ki nem alakul, vagy a lépésköz 1 nem lesz.

Összefésülés

Van két n elemű rendezett tömbünk: (a_1, \dots, a_n) és (b_1, \dots, b_n) . Ezeket kell egyesíteni. Hasonlítsuk össze az b_1 elemet rendre az a_1, a_2, \dots elemekkel. Beszűrjük b_1 -et az elé az a_i elé, amelyet

közvetlenül megelőző. Ettől az a_i -től folytatjuk tovább b_2 -re, és így tovább. Legfeljebb $2n-1$ összehasonlításra van szükség.

Minimum (maximum) elven alapuló rendezés

Egy adott résztömbben (kezdetben a teljes tömb) megkeressük a legkisebb (legnagyobb) elemet, és ezt tesszük az első helyre. A következő lépésben a második elemtől kezdve vizsgáljuk és ismét a legkisebb (legnagyobb) elemet visszük előre, és így tovább. A mindenkor legkisebb elem keresése lineáris kereséssel történik.

Rendezés közvetlen kiválasztással és cserével

A közvetlen kiválasztásos rendezés során megkeressük a tömb legkisebb elemét és felcseréljük a tömb első elemével. Majd a maradék tömb legkisebb elemét keressük meg és felcseréljük a tömb második elemével. Az algoritmust addig folytatjuk, míg a tömb végére nem érünk.

Buborékrendezés

Az egyik leggyakrabban használt rendező algoritmus. A rendezés "alulról" és "felülről" is indulhat. Az alulról induló rendezésnél mindig a legkisebb elem kerül biztosan az adott rendezési ciklusban a helyére, mert mindegyiknél könnyebb. A felülről induló rendezésnél pedig a legnagyobb elem fog biztosan a helyére kerülni, hiszen mindegyiknél nehezebb. A többi elem azonban még (valószínűleg) nem került a végleges helyére. Így a rendezést folytatni kell mindaddig, míg a teljes tömb rendezett nem lesz. Előfordulhat, hogy a tömb már menet közben, a rendezési ciklus vége előtt, rendezetté válik, esetleg eleve rendezett volt. Ezt onnan vehetjük észre, hogy az adott rendezési ciklusban nem történik csere.

Gyorsrendezés (quick-sort)

Az eddig ismert rendező algoritmusoktól eltérő, rekurzív elven működik. Kiválasztjuk a rendezetlen tömb egy tetszőleges (általában középső) elemét. Ez lesz az úgynevezett alapelem, a többi elemet ehhez fogjuk viszonyítani. A tömböt úgy rendezzük át, hogy a tömb egyik felében csak az alapelemnél kisebb, vagy egyenlő, a másik felében pedig csak az alapelemnél nagyobb, vagy egyenlő elemek helyezkedjenek el. Az átrendezést úgy hajtjuk végre, hogy az alapelemtől balra eső tömbrészen megkeressük az első, az alapelemnél nagyobb elemet, az alapelemtől jobbra eső tömbrészen pedig megkeressük az első, az alapelemnél kisebb elemet majd a két

elemet felcseréljük. A kereséseket és a cseréket addig folytatjuk, míg valahol össze nem ér, illetve át nem lapolódik a balról, illetve jobbról indított tömbindex.

A programozás alapjai

A számítógép számára a feladat meghatározását **programozásnak** nevezzük. A programozás valamilyen a **programozási nyelv** segítségével történik. A programozási nyelv a számítástechnikában használt olyan, az ember által is olvasható és értelmezhető utasítások sorozata, amivel közvetlenül, vagy közvetve (például gépi kódra fordítás után) közölhetjük a számítógéppel egy adott feladat elvégzésének módját.

A programozás egy összetett folyamat, melynek célja és eredménye a programok előállítása. A programozás lépései:

- a megoldandó feladat megfogalmazása, definiálása;
- a megfelelő algoritmus kiválasztása, megtervezése;
- az algoritmus logikai szerkezetének megtervezése;
- az algoritmus szerkezeti lebontása;
- az algoritmus kódolása;
- a kész program futtatása, tesztelése;
- a dokumentáció elkészítése;
- a program karbantartása, felügyelete, nyomon követése.

A programozási nyelveket történeti fejlődésük alapján sokféleképpen lehet osztályozni, csoportosítani. A programnyelvek változásának egy-egy fontosabb állomását a programozási nyelvek generációjának nevezzük.

A programozási nyelvek **első generációja**, a legalacsonyabb szintű programnyelv, a **gépi (bináris) kód**. A gépi kódot tulajdonképpen nem is lehet igazi programnyelvnek tekinteni, hiszen erre a klasszikus értelemben vett programnyelvi elemek (utasítások, vezérlőszervezetek, kifejezések) nem jellemzőek.

A számítógéppel való kapcsolat megteremtésének a legközvetlenebb módja viszont a gépi nyelv használata. A gépi nyelvek a számítógépek speciális műszaki tulajdonságait használják ki. Ez az a nyelv, amelyet a processzor közvetlenül megért. A gépi kódú programok az adott számítógép típusokhoz kötődnek, azok más felépítésű számítógépen nem futtathatók. A gépi utasításkészlet

bináris számokból álló egyszerű utasítások csoportja, az utasítások pedig műveleti kódból és címrészből tevődnek össze.

A gépi kódban való programozás azonban rendkívül nehézkes, hisz az egyes műveletek kódjának és az utasítások szerkezetének az ismerete is szükséges hozzá. Ezért a gépi kódú programozást főként a számítógépek kezdeti időszakában használták (más lehetőség ekkor még nem is állt rendelkezésre).

A gépi kódú programozás kényelmetlenségeit kiküszöbölendő, az egyes utasításoknak néhány karakteres szimbólumokat, úgynevezett **mnemonikokat** feleltetnek meg, melyeket programmal gépi kódra fordítanak. A programnyelvek **második generációját** képviselik az ilyen alacsony szintű (gépközel) nyelvek. Ezen nyelveknél megmarad a számítógép erőforrásainak (hardver, operációs rendszer) maradéktalan kihasználhatósága, a memóriacímek, a regiszterek, a verem és a megszakítások közvetlen elérhetősége, programozhatósága.

Ilyen második generációs nyelv az **assembly**, melynek fordítóprogramja az **assembler**. Az alacsony szintű nyelveknél minden gépi utasításnak megfelel egy nyelvi utasítás (egy nyelvi utasítás azonban állhat több gépi utasításból is). Megjelennek az adatok és az adatok helyfoglalását, a programkód memóriába helyezését definiáló utasítások. A memóriacímek azonosítóval való ellátására is van lehetőség. Elnevezhetjük az adatok kezdőcímét, ebből alakul ki később a változó fogalma, az utasítások címeinek elnevezésből pedig a címke fogalma.

Egy minden szempontból megfelelő assembly szintű program létrehozásához nem csak a processzor végrehajtható utasításait helyettesítő szimbólumok, hanem a program szerkesztését, javítását és dokumentálását megkönnyítő lehetőségek, úgynevezett direktívák is szükségesek. A direktívák csak az assembler számára hordoznak információt.

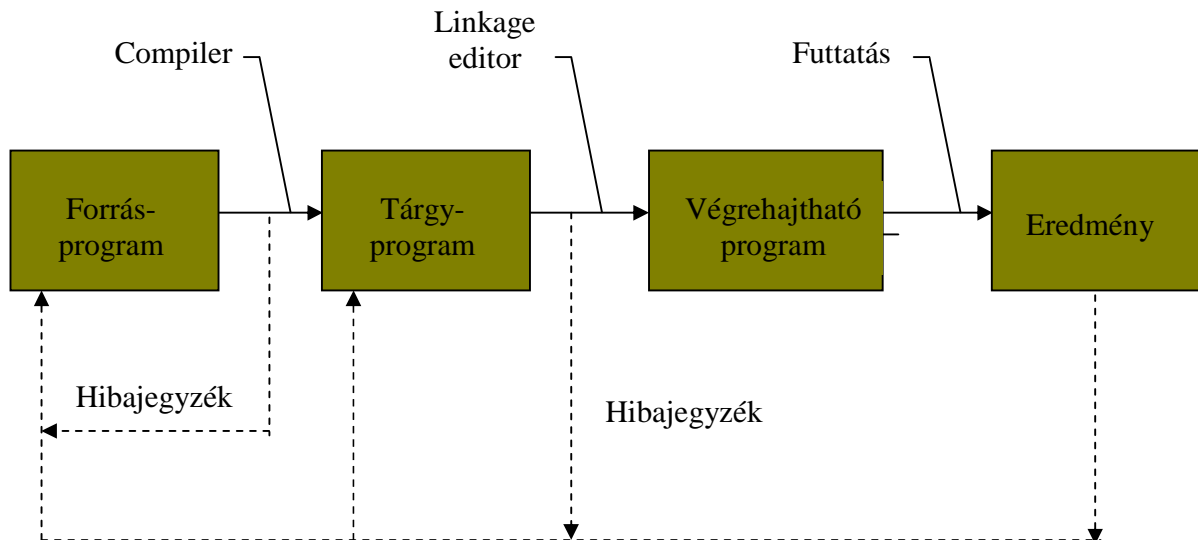
Azonban még ez a programozási mód is túl nehézkes, és nagyobb feladatok megoldására alkalmatlannak bizonyult. Ezért fejlesztették ki a **magas szintű programozási nyelveket**, melyek segítségével a megoldandó feladatot könnyebben és tömörebben lehet megfogalmazni. A programnyelvek **harmadik generációját** nevezzük magas szintű programozási nyelveknek, amelyeket gyakran a 3GL jelöléssel szokás ellátni (3rd Generation Language). A programnyelvek e generációjánál a változófogalom pontosabb megjelenése az egyik legfontosabb tulajdonság. A változó azonosítója nem memória-kezdőcímet, hanem egy egész memóriatartományt jelöl. A változófogalom mellett kialakul a típusfogalom, és lehetőség van arra, hogy a programozó saját

típusokat is létrehozzon. Kialakulnak a strukturált programnyelvek, megjelennek az elágazások, a ciklusok, az eljárások és a függvények. Az eljárásoknak lehetnek név, illetve cím szerint átadott paraméterei is. A harmadik generációs programnyelveket feladatorientált vagy **eljárásorientált** nyelveknek is nevezzük, mivel felépítésükből fakadóan a programfejlesztő számára egy program adott részfeladatainak megoldására szolgáló struktúrákat (eljárásokat) kínálnak. Nagy előnye a magas szintű programnyelveknek, hogy a velük készült programok függetlenek a konkrét számítógéptől, hiszen az egyes gépek specialitásait a fordítóprogramnak kell figyelembe vennie a fordítás során.

Egy-egy magas szintű programnyelvhez minden géptípusra más-más fordítóprogram tartozik. A **fordítóprogram (compiler)** segítségével lehet a magas szintű programnyelv utasításait az adott számítógép gépi kódú utasításaira lefordítani. Az interaktív gépek megjelenésével kialakult a programozási nyelvek egy olyan csoportja, melyeknél a fordítási menet nem különül el a végrehajtástól, hanem a forrásprogram utasításait rögtön értelmezi és végre is hajtja a fordítóprogram. Az ilyen típusú fordítóprogram az értelmező (**interpreter**).

Ma már nagyon sok különböző nyelv létezik. Ezek kifejlesztése általában valamilyen típusú problémák, feladatok megoldására irányul - célorientált, problémaorientált. Az egyes programozási nyelvek önálló utasításkészlettel, szabályrendszerekkel rendelkeznek. Ezeket a szerkezeti, formai követelményeket a nyelv **szintaxisának** nevezzük. Az utasítások szerkezete a gépi kódú utasításokhoz viszonyítva sokkal összetettebb, képzési módjuk sokkal inkább követi az emberi gondolkodásmódot. A magas szintű programozási nyelvek utasításai több gépi utasítást, tehát egy teljes utasítássorozatot takarnak, nem veszik figyelembe a számítógép felépítését. Ennek megfelelően a magas szintű fordítóprogramok is bonyolultabbak. A magas szintű programozási nyelvek gépfüggetlenek, tehát átvihetők a forrásprogramok az egyik gépről a másikra. Ennek csupán egyetlen feltétele van, az, hogy az adott gépen az adott operációs rendszer alatt legyen ugyanolyan fordító. Minden programozási nyelvnek létezik egy úgynevezett hivatkozási nyelve (például Standard Pascal), amelyet – ideális esetben - a nyelv implementációi (vagyis a konkrét fordítóprogramok illetve értelmezők) megvalósítanak. Ha csak a hivatkozási nyelv által definiált eszközöket használjuk, akkor a program - forrás szinten - változtatás nélkül átvihető az egyik típusú gépről a másikra, egyébként, ha kihasználjuk az implementációnak a hivatkozási nyelvtől eltérő (esetleg azzal ütköző) lehetőségeit, akkor a programot módosítani kell egy másik implementációra való átvitelhez.

Számítógépes program írásakor az adott nyelv szintaktikája alapján készítjük el a programot. A program kódolásától a végrehajtásáig vezető leggyakoribb utat a következő ábrán szemléltetjük:



A fordítást a programozó indítja el. A **forrásprogramot** a fordítóprogram szintaktikailag ellenőrzi. Ha hibátlan, előáll a **tárgyprogram**, ellenkező esetben hibajegyzéket kapunk. A hibajelzések alapján a programozó javít, és újraindítja a fordítást. Sok esetben többszöri próbálkozás után áll elő a tárgyprogram.

A tárgyprogram önmagában még nem végrehajtható. Ezt a szerkesztőnek más néven kapcsolatszerkesztőnek (linkage editor) össze kell szerkesztenie egyéb, a nyelv, illetve a rendszerszoftver által biztosított rutinokkal. Az esetleges további (strukturális) hibák kiküszöbölése után áll elő a végrehajtható program.

Ha a végrehajtható programot futtatjuk, elkezdjük a program tesztelését, azaz a **szemantikai** (logikai, tartalmi) **hibák** kiküszöbölését. A program szemantikailag hibás, ha elindul ugyan, de nem azt csinálja, amit kell, esetleg menet közben le is áll. Az ilyen hiba sok gondot okozhat a programozónak, mivel az poloska módjára rejtőzik a programban. Ezért a hiba neve a poloska angol megfelelője után **bug**, megtalálása és kiirtása a **debuggolás**, az erre alkalmas segédprogram a **debugger**.

A programfejlesztés során ezeket a feladatokat egyre inkább úgynevezett **integrált fejlesztői környezetben (IDE - Integrated Development Environment)** hajtják végre. Az IDE a programozást jelentősen megkönnyítő, részben automatizáló programok összessége. Tartalmaznak egy szövegszerkesztőt a program forráskódjának szerkesztésére, egy fordítóprogramot vagy értelmezőt, fordításautomatizálási eszközöket, valamint nyomkövetési, és gyakran grafikusfelület-szerkesztési és verziókezelési lehetőségeket sok egyéb mellett. A komolyabbakhoz kiegészítők tömege érhető el, amelyek a rendszerfejlesztés egyéb fázisaiban, például a dokumentálásban, projektmenedzsmentben stb. nyújtanak nagy segítséget.

Ma már nagyon sok magas szintű programozási nyelvet ismerünk. Ezek között van általános és speciális célú. Az alábbiakban néhány, az elsők közül való és valamilyen szempontból nevezetes, harmadik generációs programozási nyelvet ismertetünk.

A **BASIC** (Beginner's All-purpose Symbolic Instruction Code) programozási nyelv a magas szintűek közül az egyik legegyszerűbb. Első verzióját is oktatási célból készítette **Kemény János** (1926-1992) magyar-amerikai tudós. A BASIC-nek azóta nagyon sok különböző változata született meg.

1954-ben az IBM-nél egy kutatócsoport hozta létre a **FORTRAN** (FORmula TRANslation - formulafordítás) nyelvet, ami egyike a legrégebben használatos magas szintű programozási nyelveknek. Ennek megfelelően elég sok változata létezik, amelyek túlnyomórészt a korszerűsítések eredményei. Elsősorban matematikai számítások, fizikai kutatóintézetekben szükséges számítások elvégzésére fejlesztették ki.

Szintén a legkorábban kifejlesztett programozási nyelvek közé tartozik az **ALGOL60** és az **ALGOL68** (ALGOrithmic Language). Elsősorban algoritmikus problémák megoldására fejlesztették ki. Az ALGOL megjelenése annyiban volt hatással a programozási nyelvekre, hogy még ma is van sok ún. "ALGOL"-szerűnek nevezett nyelv.

A 60-as években fejlesztették ki az USA Hadügyminisztériumának megbízásából a **COBOL** (Common Business Oriented Language – általános üzletorientált nyelv) programnyelvet. A 60-as, 70-es években a magas szintű nyelven írt programok többségét COBOL-ban írták. Elsősorban adatfeldolgozásra készült, ezen a területen még ma is használják. Hasonlít a normál beszédhez, utasításai állítások.

Az IBM a 60-as évek közepén új nyelvet tervezett, ami a **PL/1** nevet kapta. Ebben a nyelvben a korábbi programozási nyelvek előnyeit próbálták egyesíteni. A nyelv ezért eléggé univerzális, s ebből következően célszerű használatához nagyon nagy gépre volt szükség. Egyik legfontosabb erénye az adatállományok igen jó, rugalmas, magas logikai szintű definiálása és kezelése, illetve hibakezelése. Ma már kevésbé használatos nyelv, az ESzR gépek korszakában volt népszerű (1969-85).

Ken Thompson (1943-) 1970-ben dolgozta ki a **B** nyelvet, aminek segítségével az első UNIX operációs rendszer készült. Mivel a B nyelv nem volt elég hatékony, ezért 1971-ben Dennis Ritchie (1941-) kifejlesztette a **C** nyelvet, amiből 1983-ban létrejött az első szabványos C, majd 1989-ben megszületett az ANSI C szabvány végleges változata. *(Megjegyzés: Ezt C89-ként, illetve C90-ként is szokták emlegetni, ugyanis az ISO egy évvel később fogadta el. Van C99 is azóta.)* C nyelven írják a UNIX operációs rendszert, és ehhez általában hozzátartozik a C fordító is. A C nyelv elterjedését segítette, hogy nagyon hatékony programozási nyelv és a C forrásprogramok elég jól hordozhatóak a különböző platformok között. Ha egy C programban csak szabványos elemeket használunk, akkor valószínűleg mindenhol futni fog, minimális átalakítással. (Manapság a C nyelv szerepét a C++ veszi át, ami a C nyelv objektumorientált kiterjesztése. Általában a C programok gond nélkül használhatóak C++ rendszerben is, és a fordítók többsége ismeri mindkettőt.) Szokás a C nyelvet alacsony szintű programozási nyelvként is említeni. Ez azonban csak azt jelenti, hogy általában a számítógép által támogatott adattípusokkal dolgozik. Támogat olyan alacsony szintű műveleteket is, amelyet más magas szintű programozási nyelvek nem támogatnak, illetve sajátossága, hogy támogatja a bitstruktúrák kezelését is. (Ez utóbbi tulajdonságai miatt sokszor kiváltja az assembly nyelvet.) A C nyelv azonban nem tartalmaz I/O utasításokat. E műveletek elvégzésére függvénykönyvtárakat hoztak létre. A C nyelv nem tartalmaz sztringek (karakterláncok), halmazok és listák kezelésére szolgáló műveleteket sem. A függvénykönyvtárak segítségével azonban több hasznos eszközhöz juthatunk, mint más programozási nyelvekben.

A programnyelvek következő generációját nevezték el **negyedik generációs** (4GL = 4th Generation Language) nyelveknek, mivel az ezekkel az eszközökkel történő programfejlesztés merőben más technikákat és módszereket kívánt a fejlesztőktől, mint a magas szintű programozási nyelvek.

A 4GL eszközök magas szintű programozási nyelvre épülő komplex, objektumorientált programfejlesztői rendszerek. A 4GL rendszerek a hagyományos programozási nyelvektől eltérően nem igénylik az összes elemi - különösképp a felhasználói felületekre vonatkozó - tevékenység implementációját. A programozó készen kap olyan elemeket, objektumokat, amelyekkel ezeket a gyakran időrabló feladatokat gyorsan és hatékonyan képes megoldani. Így például nem szükséges a felhasználói felületet a részletek szintjén utasításonként programozni, elegendő csak a képernyőn megjelenő elemeket megadni, és a fejlesztőrendszer ez alapján már automatikusan generálni tudja az űrlap (ablak) egy alapértelmezés szerinti felépítését.

Ezzel a módszerrel a 4GL eszközökkel történő fejlesztés során a programozó magasabb szintű absztrakcióval készítheti a programjait. Erre az úgynevezett komponens alapú fejlesztési módszer teremti meg a lehetőséget. A komponensek olyan előre gyártott építőelemek, melyeket a fejlesztőkörnyezet tartalmaz. A programozó a vizuális tervezőfelületen ezekből összeállíthatja a programja vázát, felhasználói felületét. Ezen a szinten a programkód is automatikusan generálódik, tehát a fejlesztő idejének nagyobb hányadát fordíthatja a speciális algoritmusok elkészítésére. Manapság a nagyobb rendszereket szinte kizárólag ilyen 4GL környezetben fejlesztik, és olyan kisebb alkalmazásoknál is egyre jobban teret hódít ezek használata, mint például az adatbáziskezelő rendszerek.

A következő fejlődési szakasz a programozási nyelvek tekintetében az **ötödik generációs** nyelvek, vagy „intelligens nyelvek”, amelyek kutatása, fejlesztése még folyamatban van. Igen nagy számítógépkapacitást igényel: elvileg az emberi gondolkodás leírása történne meg, és a természetes, emberi nyelvet transzformálnák a számítógép által értelmezhető formára.

A programozási nyelvek számának növekedésével egyidejűleg változtak a programozási módszerek is, amely a nyelvek egy másik csoportosítását teszi lehetővé.

A kezdeti programozási nyelveknél alkalmazott, az építőkockák elvén alapuló programfejlesztési módszer a **moduláris programozás**, amikor a számítógéppel megoldandó feladatot egymástól független részfeladatokra osztják fel. Az egyes a modulokat külön-külön lehet megírni, átvenni már megírt programokból (rutinyűjteményből), és a tesztelés is modulonként történik. Második lépésként a kész, jól működő modulokat egy egységbe illesztjük, ellenőrizzük a modulok közötti információcserét, valamint a vezérlésátadások helyességét. A modulokat **szubrutinnak** szokás nevezni. A szubrutin nem más, mint az utasítások olyan sorozata, amely az adott program

keretein belül önálló logikai és szerkezeti egységet alkot (azaz részfeladatot lát el), és a program futása során többször is meghívható. A program a szubrutinokat hívja meg a megfelelő sorrendben, és biztosítja a megfelelő paraméterátadást.

A **strukturált programozás** esetén a legelső dolgunk, hogy a feladatot (a problémát), amire programot szeretnénk írni, kielemezzük. A nagy feladatot mindig kisebb részekre bontjuk, azokat pedig még kisebbekre, egészen addig, amíg el nem érünk valamilyen, tovább már nem bontható részfeladatig. Az ilyen, tovább már nem bontható részfeladatokhoz már kidolgozhatjuk a megfelelő algoritmust. Az algoritmusban csak meghatározott vezérlési struktúrákat használhatunk, és megírjuk hozzá a megfelelő programrészteket. Az így kapott program könnyen javítható, hiszen hamar kideríthető, hogy a hiba melyik programrészben fordult elő, és a program fejlesztése is viszonylag egyszerű. Az egyik legismertebb magas szintű programozási nyelv, amellyel a strukturált programozás elve megvalósítható, a **Pascal**, amit eredetileg nem a gyakorlatban használatos programnyelvnek szántak, hanem a strukturált programozási elv oktatására kidolgozott eszköznek.

Az **objektumorientált programozás** (OOP) sok tekintetben a moduláris programozás továbbfejlesztett változata, mely háttérbe szorítja a strukturált programozást. Az egyes elemeket itt nem moduloknak, hanem osztályoknak nevezzük. Az egyes osztályok a modellezett világ azonos fajta tulajdonságokkal és azonos viselkedéssel rendelkező objektumait írják le. Az objektumorientált programozás jobban közelíti, utánozza a valóságot, mint elődei, mert jobban igazodik a tárgyhöz. Az osztály létrehozásakor definiáljuk a szerkezetét, megadva objektumainak tulajdonságait, és a szubrutinokat, melyek leírják objektumainak a viselkedését, így a későbbiekben mindent együtt kezelhetünk, ami az adott objektumhoz tartozik. Az így létrejött nyelv sokkal strukturáltabb, sokkal modulárisabb, ami megmutatkozik az adatok szerkezetében, valamint a fölösleges hivatkozások elhagyásában is.

Főbb tulajdonságai:

- Egy rekord összekapcsolása eljárásokkal és függvényekkel. Ez jellemzi az új adattípust, az osztályt.
- Az osztályok a korábban definiált osztályoktól adatokat és módszereket vehetnek át (örökölhetnek). Egy adott típusú objektum kompatibilis a szülő típusok objektumaival.

Az **aspektusorientált programozás** (AOP) is a moduláris programozás továbbfejlesztett változata. Bár a legtöbb aspektusorientált nyelv egyben objektumorientált is, az AOP elvei függetlenek az objektumorientáltságtól. Lényege, hogy a program modulokra bontásakor egyidejűleg több szempontrendszert is érvényesíthetünk. Ezzel a módszerrel el tudunk különíteni (külön modulba tudunk kiemelni) olyan programrészeket is, amelyek a moduláris programozás hagyományos eszközeit használva több modulban szétszóródva jelennének meg. A hagyományos eszközökkel ugyanis a programot csak egyetlen, kitüntetett szempontrendszer szerint bonthatjuk modulokra, viszont emiatt azok a programrészek, amelyek egy másik szempontrendszer szerint logikailag összetartoznának (a program egy vonatkozását hordozzák), így több modulban szétszóródva jelennek meg. Ezzel egyidejűleg jellemző probléma, hogy a modulokban emiatt összekeverednek a program különböző vonatkozásait hordozó elemek.

Az AOP úgy küszöböli ki a vonatkozások szétszóródását és keveredését, hogy a szétszóródó elemek is kiemelhetők egy modulba. Ilyenkor a modulban le kell írni, hogy az így kiemelt elemek a program mely pontjaira illesztendők be. Az ilyen modulokat rendszerint aspektusnak nevezik, a beillesztést pedig szövésnek. Ez történhet már a program fordításakor, vagy csak a futtatásakor is.

Alkalmazások

A felhasználó a számítógép és az operációs rendszer szolgáltatásait legtöbbször nem közvetlenül veszi igénybe, hanem hozzá közelebb álló, egyszerűbben kezelhető úgynevezett felhasználói programokon keresztül. Ma már minden operációs rendszer lehetővé teszi mind a hálózati, mind a helyi alkalmazások elérését.

A **hálózati alkalmazások** elérését az alkalmazási réteg teszi lehetővé. Az összes olyan funkciót biztosítja, amelyet az alsóbb rétegek a rendszerek közötti kommunikációnál használnak. Ezekről már volt szó a számítógéphálózatokról szóló fejezetben.

A **helyi alkalmazásokat** azok a standard felhasználói programok teszik lehetővé, amelyek több felhasználói terület, felhasználói réteg vagy szakmacsoport által használt szoftvereket foglalják magukba. Ezeket a nagy vásárlói kör miatt általában nagy nemzetközi szoftvervállalatok készítik és forgalmazzák, és árulják olcsóbban, mint az egyedi fejlesztésű programokat.

Néhány fontosabb alkalmazási terület:

- szövegszerkesztés;
- táblázatkezelés;
- bemutatókészítés;
- adatbáziskezelés;
- képkezelés;
- hangkezelés.

A helyi alkalmazásokat biztosító szoftverek közötti adatátvitelt az operációs rendszer az úgynevezett **vágólap** (clipboard) segítségével biztosítja. A vágólap valójában egy átmeneti tároló, ami vagy a memóriában, vagy a rendszert tartalmazó lemezen tárolódik. A legtöbb operációs rendszerben a tartalma törlődik a számítógép kikapcsolása után.

Az adatátvitel nagyon egyszerű, mert az alkalmazói programokban csak néhány utasítást lehet használni a vágólap kezelésére. A vágólapra adatot helyezni a **Kivágás** (Cut) és a **Másolás** (Copy) utasításokkal, onnan visszaszedni a **Beillesztés** (Paste) utasítással lehet.

Szövegszerkesztés

Az alkalmazások közül külön figyelmet érdemel a szövegszerkesztés, mint a leggyakrabban használt eszköz. Szinte minden egyéb alkalmazás, programfejlesztés, levélírás, kiadvány készítéséhez, stb. nélkülözhetetlen eszköze.

A szövegszerkesztő programmal könnyen, gyorsan szép, esztétikus kiadványokat tudunk készíteni, amit bármikor módosíthatunk, reprodukálhatunk, kinyomtathatunk. Az **egyszerű (ASCII) szövegszerkesztők** használatakor a beírt szöveget nem lehet formázni. A **formázást megengedő szövegszerkesztők** segítségével formai beállítások is végezhetők (betűformázás, képbeillesztés, stb.).

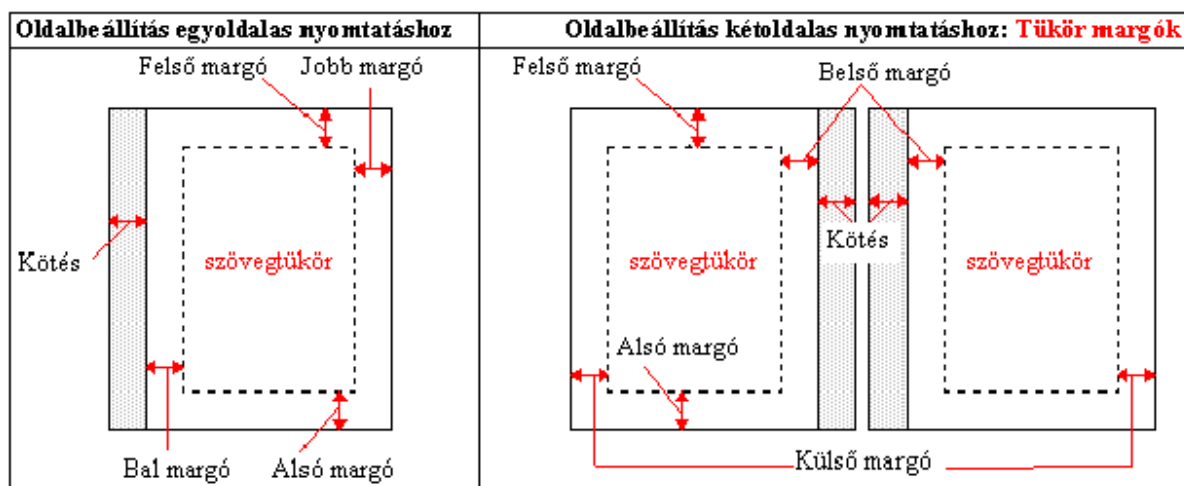
Régebben a szövegszerkesztők nem tudták megmutatni a nyomtatási képet, azaz, hogy kinyomtatva milyen lesz a dokumentum. A szerkesztés közben a szövegben különféle kódok utaltak a formázásokra. A jelenleg használatos szövegszerkesztők szerkesztés közben azt a képet mutatják, amit nyomtatáskor kapunk. Ezek a **WYSIWYG** (What You See Is What You Get = amit látsz, azt kapod) szövegszerkesztők.

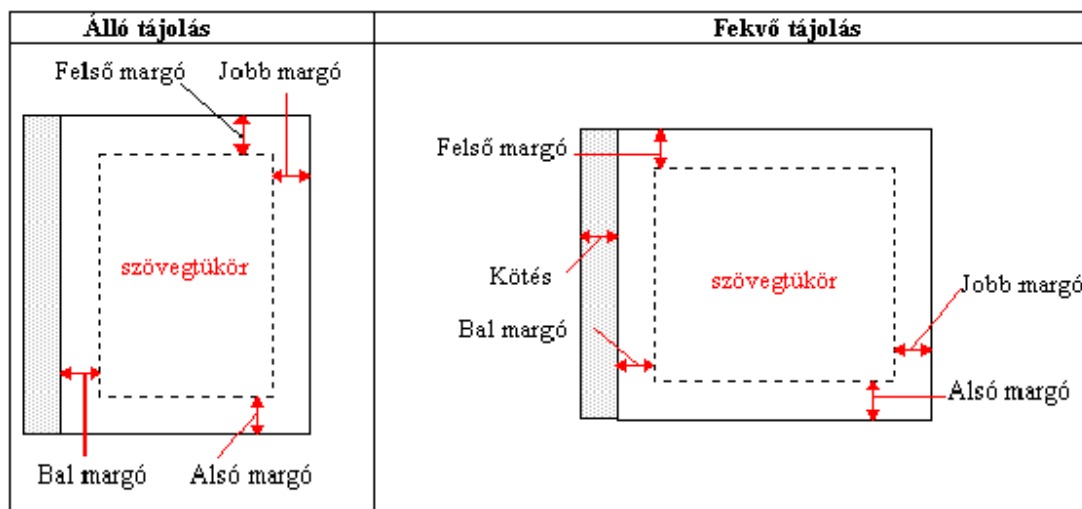
A szövegszerkesztés lépései

1. megnyitás (új, régi);
2. a szöveg begépelése;
3. formázás;
4. mentés (mentés másként);
5. nyomtatás.

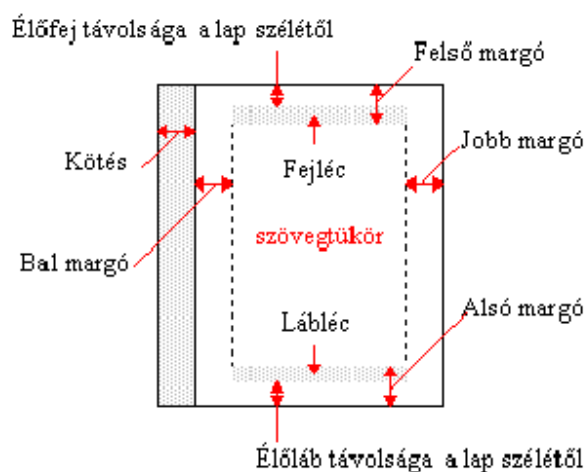
Tipográfiai ismeretek

- a papír beállításai (oldalbeállítások - papírméret, margók, befűzés, tájolás);

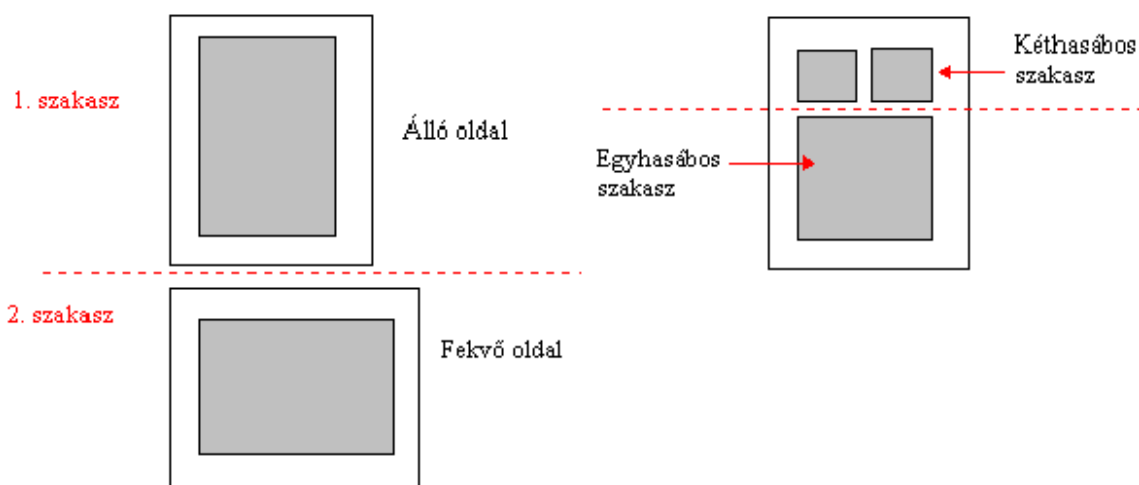




- **szöveghelyek** (szövegtükör, fejléc, lábléc, oldalszámzás);



- **tipográfiai egységek** (karakter, szó, sor, bekezdés, szakasz, dokumentum);






- **a dokumentumban elhelyezhető objektumok** (képek, rajzok, táblázatok).

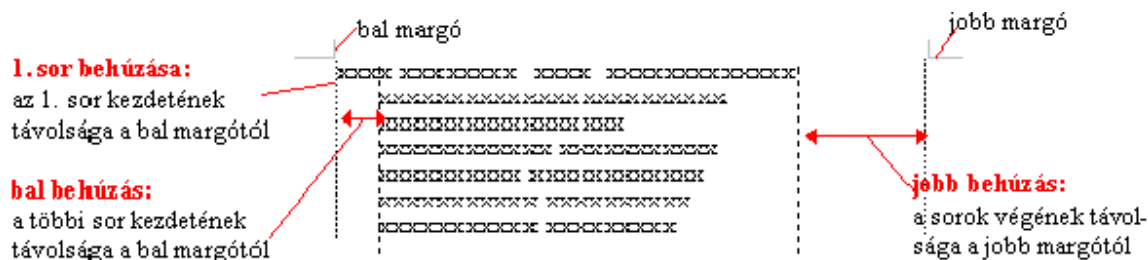
Jellemző formázási lehetőségek

Betűtípus: a betű jellegét (alapvető alakját) határozza meg. A betűtípusokat karakter tábláknak is nevezik. <ul style="list-style-type: none"> Több száz féle betűtípust (font készletet) lehet kapni lemezen (floppyn vagy CD-n). A CE (Central Europa=Közép-Európa) jelölésű betűtípusok tartalmaznak valódi ékezetes betűket. Az alábbi három betűtípus általában minden Windows-os gépen megtalálható: 		
Times New Roman betűtípus: ABCDEFGHIJKLMNOPQRSTUVWXYZ YZ abcdefghijklmnopqrstuvwxyz 1234567890	Courier betűtípus ABCDEFGHIJKLMNOPQRSTUVWXYZ abcdefghijklmnopqrstuvwxyz 1234567890	Arial betűtípus ABCDEFGHIJKLMNOPQRSTUVWXYZ YZ abcdefghijklmnopqrstuvwxyz 1234567890
Hasonló a Times újság egyszerűen szép betűihez. <ul style="list-style-type: none"> Eltérő szélességű karakterek 	Hasonlít az írógép betűihez. <ul style="list-style-type: none"> Egyforma széles karakterek 	Egyszerű egyenes betűkből áll <ul style="list-style-type: none"> Eltérő szélességű karakterek

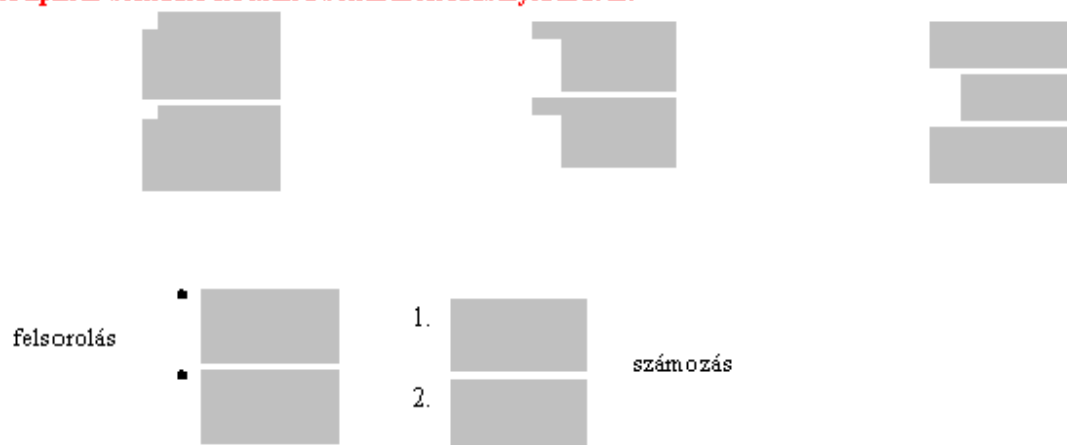
További formai jellemzők: Ezek a jellemzők minden betűtípuson beállíthatók.		
Jellemző	Minta	Megjegyzés
Betűközők: <ul style="list-style-type: none"> Ritkított Sűrített 	Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg	nagyobb betűtávolság kisebb betűtávolság
Pozíciók: <ul style="list-style-type: none"> Emelt alapvonalú Süllyesztett alapvonalú 	Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg	
Animált <ul style="list-style-type: none"> Fényreklám Konfetti Mene telő fekete hangyák Mene telő vöröshangyák Villódzás Villogó háttér 	Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg	

További formai jellemzők: Ezek a jellemzők minden betűtípuson beállíthatók.		
Jellemző	Minta	Megjegyzés
Betű méret	10, 12, 14, 16, 18...	Általában minden betűtípus méretezhető. A betűméret mértékegysége a pont. 1 pont=0,35mm
Betű stílusok: • Félkövér • Dőlt • Aláhúzott	Mintaszöveg <i>Mintaszöveg</i> <u>Mintaszöveg</u>	A betű stílusok szövegrészek kiemelésére használatos formai jellemzők.
Aláhúzások: • Szimpla • Dupla • Szóaláhúzás • Pontozott • Vastag • Szaggatott • Pont vonal • Pont pont vonal • Hullámos	<u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u> <u>Mintaszöveg</u>	további szövegrész kiemelési lehetőségek ← a szóközöket nem húzza alá
Szín	Normálszöveg → <i>Minta szöveg</i>	
Különlegességek: • Áthúzott • Kétszer áthúzott • Felső index • Alsó index • Rejtett • Kis kapitális • Árnyékolt • Körvonalas • Domború • Véssett • Nagybetűs	Normálszöveg → <i>Mintaszöveg</i> Normálszöveg → <i>Mintaszöveg</i> Normálszöveg → <i>Mintasöveg</i> Normálszöveg → <i>Mintsöveg</i> Normálszöveg → <i>Mintasöveg</i> Normálszöveg → MINTASZÖVEG Normálszöveg → <i>Mintasöveg</i> Normálszöveg → Mintasöveg Normálszöveg → <i>Mintasöveg</i> Normálszöveg → <i>Mintasöveg</i> Normálszöveg → MINTASZÖVEG	felfelé tolt alapvonalú (kisebb) lefelé tolt alapvonalú (nagyobb) nem látszik (de ott van) a kisbetűket kisebb nagybetűként jeleníti meg

Bekezdésre utaló formai jelek:		
Az új bekezdés új sorban kezdődik.	A bekezdések első sora vízszintesen el van tolva a többi sorhoz képest.	A bekezdéseket sokszor térközőkkel választják el.
 1. bekezdés <hr style="border-top: 1px dashed black;"/> 2. bekezdés	 1. bekezdés <hr style="border-top: 1px dashed black;"/> 2. bekezdés	 1. bekezdés <hr style="border-top: 1px dashed black;"/> 2. bekezdés



A tipikus bekezdés formák a behúzások szabályozásával:



Balra igazítás: a bekezdés sorai egy vonalban kezdődnek.	Középre igazítás: a bekezdés sorai a behúzások között középre igazodnak.	Jobbra igazítás: a bekezdés sorai egy vonalban végződnek.	Sorkizárt igazítás: a bekezdés sorainak mindkét vége egy vonalban van
 behúzások	 behúzások	 behúzások	 behúzások

Oldalkép nézet: Nézet→Oldalkép	Normál nézet: Nézet→Normál		
Minden objektum a helyén látható úgy, ahogy nyomtatáskor a papíron. (Néha van néhány apróbb eltérés a nyomtatáshoz képest.) Ha sok objektumot (rajzot, képet, táblázatot) használ a szövegben, szerkesztéskor lelassulhat a szövegrészek megjelenítése.	Egyszerűsített szövegbeépítési / megjelenítési nézet. Mivel a Word nem „szenvet” a dokumentum valósághű megjelenítésével, gyorsabban képes követni a szerkesztéseket.		
Látszik: <ul style="list-style-type: none"> minden szövegformázás beszárt kép rajzelem lapszélek, margók 	Látszik: <ul style="list-style-type: none"> minden egyszerű szöveg formázás (kivéve a hasábolás stb.) beszárt kép 	Nem látszanak: <ul style="list-style-type: none"> rajzelemek lapszélek, margók stb. (csak az oldaltükör látszik) 	

Tömörítés

A tömörítést elsősorban azért használjuk, hogy a tárolandó, az elküldendő adat kisebb helyen elférjen. Létezik veszteségmentes és veszteséges tömörítés. Az első esetben a kitömörítés után ismét az eredeti állományt kapjuk. A második lehetőség a kép-, hang- és videóállományok kevésbé fontos (érzékszerveink számára felfoghatatlan) részeinek kiszűrésén alapszik: ezek a tömörítések (pl. avi, jpeg, mpeg, mp3 stb.) veszteségesek, azaz a tömörített fájlok az eredetinél kevesebb információt tartalmaznak.

Vírusok

Olyan, rendszerint kisméretű programok vagy programrészek, amelyeket azzal a céllal hoztak létre, hogy észrevétlenül terjedjenek, és megfelelő körülmények között minél nagyobb kárt okozzanak. A felhasználó tudta nélkül működnek. Képesek saját maguk reprodukálására és terjesztésére, előbb-utóbb valamilyen kárt okoznak.

Ajánlott irodalom

1. Efraim Turban – R. Kelly Rainer, Jr. –Richard E. Potter:
Introduction to Information Technology
Wiley, 2001.
2. Csala Péter – Csetényi Arthur – Tarlós Béla:
Informatika alapjai
ComputerBooks, Budapest, 2001.
3. Alfred V. Aho – Jeffrey D. Ullman:
Foundation of Computer Science
Computer Science Press, New York, 1992.