Bevezetés az informatikába

Dr. Nyakóné dr. Juhász, Katalin

Dr. Terdik, György

Biró, Piroska

Dr. Kátai, Zoltán

Bevezetés az informatikába

Dr. Nyakóné dr. Juhász, Katalin
Dr. Terdik, György
Biró, Piroska
Dr. Kátai, Zoltán

Publication date 2011

A tananyag a TÁMOP-4.1.2-08/1/A-2009-0046 számú Kelet-magyarországi Informatika Tananyag Tárház projekt keretében készült.

A tananyagfejlesztés az Európai Unió támogatásával és az Európai Szociális Alap társfinanszírozásával valósult meg.

Tartalom

A táblázatok listája

Az egyenletek listája

1. fejezet - Bevezetés

Az informatikában használt alapfogalmak megfogalmazására, az alkalmazott eszközök bemutatására számtalan jegyzet, tankönyv található a könyvesboltokban, érhető el az interneten. Felmerülhet a kérdés, hogy miért van szükség egy újabb, sokadik bevezető informatika jegyzetre? A válasz nagyon egyszerű. „Ahány ház, annyi szokás! – tartja a közmondás, azaz minden oktatási intézmény a saját értékrendje, hagyományai alapján tanít. A Debreceni Egyetem Informatika Karán is kialakult egy tematika, ami szerint felkészítjük a hallgatókat azoknak az alapfogalmaknak az elsajátítására, melyek a további tanulmányaikhoz szükségesek.

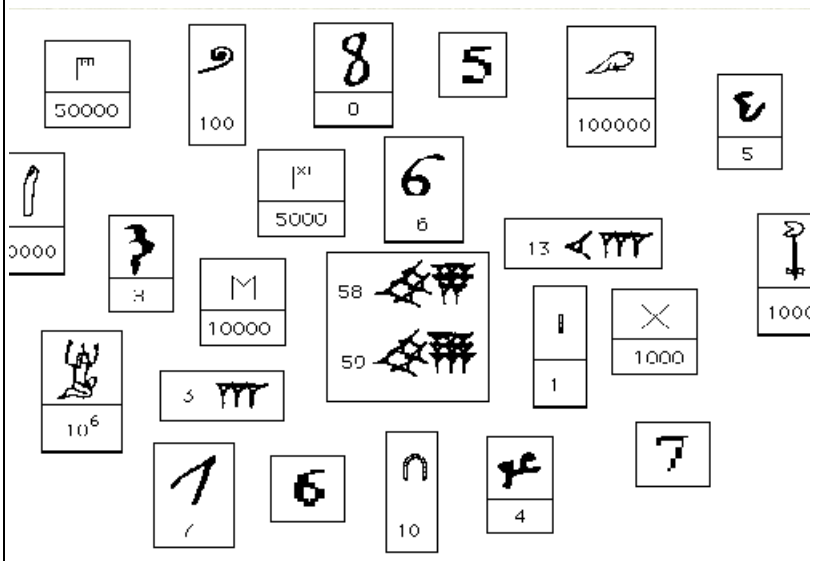
Ez a jegyzet a teljesség igénye nélkül a számrendszerektől, a számítógép felépítésén át az algoritmusok, programozási alapfogalmak, alkalmazások témakörökig fogja át azokat az ismereteket, amelyekre az elsőéves hallgatók tudnak építeni későbbi tanulmányaik során. Igyekszünk sok példával, mintafeladattal, feladattal segíteni a hallgatókat a tananyag alapos elsajátításában. Reméljük sikerrel!

A szerzők

2. fejezet - Számrendszerek

1. A számolás története, a számrendszerek kialakulása

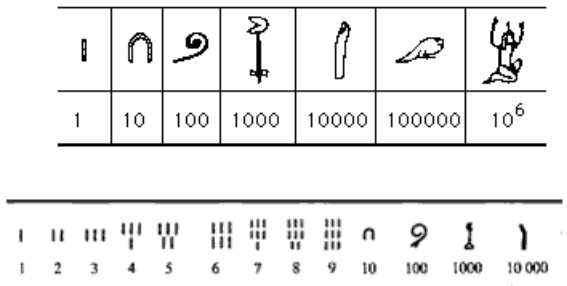
Ma Magyarországon arab számokat használunk, és tízes számrendszerben számolunk. De a történelem ennél többféle számrendszert és különböző írásmódokat ismer. Ezekből ad ízelítőt az alábbi ábra:



1. 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.)

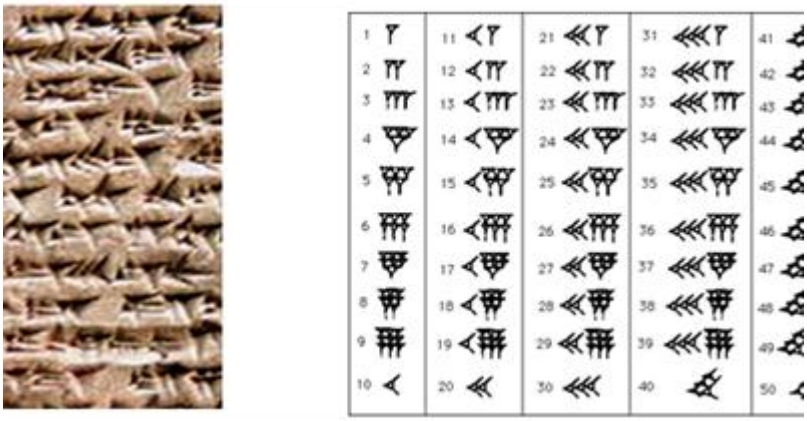
Egyiptomban az i.e. 2000 körüli időkben már jól kialakult tízes számrendszer volt, melynek elterjedését a mezőgazdaság és a csillagászat szükségletei mozdították elő. Minden magasabb tízes egységre külön jelet használtak, tehát a helyi érték fogalmát még nem ismerték. Egy pálcika: 1; két pálcika: 2; 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:



2. Ábra

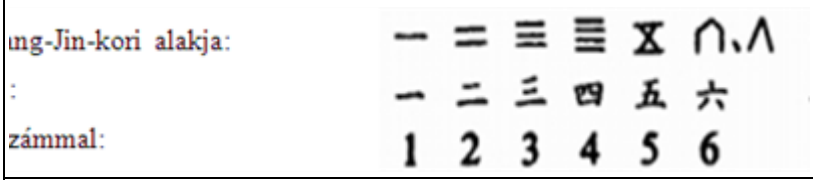
Mezopotámiában, a késői sumér korszakban, ahol a csatornázás és építkezés bonyolult számítást kívánt, fejlett helyi értékes hatvanas számrendszert találunk, amely még árulkodik az előző tízes számrendszer használatáról,

hiszen 1-től 60-ig a régebbi tízes számrendszer segítségével írták le a számjegyeket. Ékírásos jeleiket agyagba nyomták, az agyagtáblát tüzes kemencében kiégették, s ezzel olyan időtállóvá tették azt, hogy csak meg kell 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 ábrán:



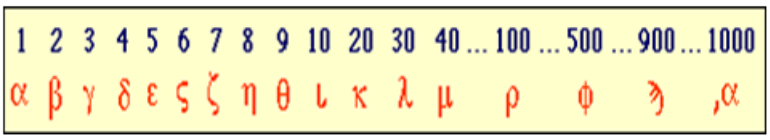
3. Ábra

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



4. Ábra

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:



5. Ábra

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:

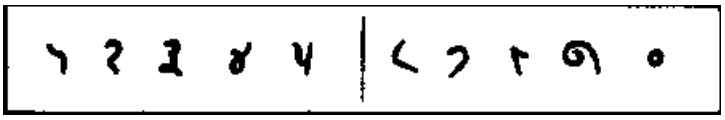
I = 1, V = 5, X = 10, L = 50, C = 100, D = 500, M = 1000,

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.

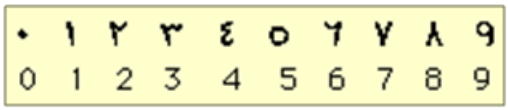
Mivel a rómaiak nem ismerték a számok helyi értékét, a nagy számok leírása kényelmetlen, és a számolás reménytelenül nehéz volt.

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.



6. Ábra

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 hétpecsétes 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 őseink. 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.



7. Ábra

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:

2.1. egyenlet - 1_egyenlet

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

ahol

2.2. egyenlet - 2_egyenlet

$$a_{-2} = 5, a_{-1} = 4, a_0 = 3, a_1 = 2, a_2 = 1$$

Általában egy

2.3. egyenlet - 3_egyenlet

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

alakú szám felírása polinom alakban, tízes számrendszerben

2.4. egyenlet - 4_egyenlet

$$\sum_{i=-m}^{n} a_i \cdot 10^i, \; ahol \; 0 \leq a_i \leq 10.$$

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

2.5. egyenlet - 5_egyenlet

$$0, 1, \ldots, p - 1$$

a helyi értékek pedig a *p* szám hatványai:

2.6. egyenlet - 6_egyenlet

$$\sum_{i=-m}^{n} a_i \cdot p^i, \; ahol \; 0 \leq a_i \leq p.$$

2. A számítástechnikában használatos számrendszerek

A számítástechnikában leggyakrabban a tízes (decimális, *p = 10*), a kettes (bináris, *p = 2*) és a tizenhatos (hexadecimális, *p = 16*) számrendszerrel dolgozunk. A kettes számrendszerben csak kétféle jelet használunk *(0,1)*, míg a tizenhatosban *16* különbözőt, ezért betűkre is szükség van *(0, 1, ..., 9, A, B, C, D, E, F)*. Az alábbi táblázat mutat néhány példát a különböző alapú számrendszerekre:

2.1. táblázat - Táblázat 1

Bináris p=2	Tetrális p=3	Kvantilis p=5	Oktális p=10	Decimális p=10	Duodecimális p=12	Hexadecimális p=16
0	1	0	0	1	0	0
1	1	1	1	1	1	1
10	2	2	2	2	2	2

Bináris p=2	Tetrális p=3	Kvantilis p=5	Oktális p=10	Decimális p=10	Duodecimális p=12	Hexadecimális p=16
11	10	3	3	3	3	3
100	11	4	4	4	4	4
101	12	10	5	5	5	5
110	20	11	6	6	6	6
111	21	12	7	7	7	7
1000	22	13	10	8	8	8
1001	100	14	11	9	9	9
1010	101	20	12	10	a	A
1011	102	21	13	11	b	B
1100	110	22	14	12	10	C
1101	111	23	15	13	11	D
1110	112	24	16	14	12	E
1111	120	25	17	15	13	F
10000	121	26	20	16	14	10

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 $2^4 = 16$.

2.2. táblázat - Táblázat 2

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 q alapú számrendszerből p alapúba történő átszámolásnál az egész részt p-vel való osztással, a tört részt p-vel való szorzással határozzuk meg.

Példák 1.

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:

2.3. táblázat - Táblázat 3

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

2.7. egyenlet - 7_egyenlet

$$123{,}45_{10} \rightarrow 1111011.0111001\ldots_2 \text{ és } 123{,}45_{10} \rightarrow 7B.733\ldots_{16}$$

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

2.8. egyenlet - 8_egyenlet

$$1111011.0111001\ldots_2 \rightarrow 2^6 + 2^5 + 2^4 + 2^3 + 2^1 + 2^0 + 2^{-2} + 2^{-3} + 2^{-4} + 2^{-7} = 64 + 32 + 16 + 8 + 2 + 1 + \frac{1}{4} + \frac{1}{8} + \frac{1}{16} = 123{,}4453125 \sim 123{,}45_{10}$$

2.9. egyenlet - 9_egyenlet

$$7 \cdot 16^1 + B \cdot 16^0 + 7 \cdot 16^{-1} + 3 \cdot 16^{-2} + 3 \cdot 16^{-3} = 112 + 11 + \frac{1}{16} + \frac{1}{256} + \frac{1}{4096} = 123{,}44995117185 \sim 123{,}45_{10}$$

2. Általában: Adott az

2.10. egyenlet - 10_egyenlet

$$a_n a_{n-1} \ldots a_1 a_0 a_1 a_{-2} \ldots a_{-m} \,|\, q$$

q ($q=10$) alapú számrendszerbeli szám és keressük azokat a b_i együtthatókat, amelyekre

2.11. egyenlet - 11_egyenlet

$$a_n \cdot q^n + a_{n-1} \cdot q^{n-1} + \ldots + a_0 + a_{-1} \cdot q^{-1} + a_{-2} \cdot q^{-2} + \ldots + a_{-m} \cdot q^{-m} + = b_N \cdot p^N + b_{N-1} \cdot p^{N-1} + \ldots + b_0 + b_{-1} \cdot p^{-1} + b_{-2} \cdot p^{-2} + \ldots + b_{-M} \cdot p^{-M},$$

Megoldás:

A feladatot két részre bontjuk. Először a

2.12. egyenlet - 12_egyenlet

$$b_N \cdot p^N + b_{N-1} \cdot p^{N-1} + \ldots + b_0$$

egész részt vizsgáljuk. A *p*-vel való osztás eredménye:

2.13. egyenlet - 13_egyenlet

$$b_N \cdot p^{N-1} + b_{N-1} \cdot p^{N-2} + \ldots + b_1$$

a maradék pedig b_0. A fenti kifejezést tovább osztjuk *p*-vel, a maradék b_1, és így tovább. Véges számú $(N+1)$ lépésben eljutunk a b_N maradékig. A maradékokat fordított sorrendben felírva megkapjuk a *p* alapú számrendszerbeli szám egész részét:

2.14. egyenlet - 14_egyenlet

$$b_N b_{N-1} \ldots b_1 b_0 .$$

A

2.15. egyenlet - 15_egyenlet

$$b_{-1} \cdot p^{-1} + b_{-2} \cdot p^{-2} + \ldots + b_{-M} \cdot p^{-M}$$

tört rész átalakítása során mindkét oldalt szorozzuk *p*-vel. Eredményül a

2.16. egyenlet - 16_egyenlet

$$b_{-1} + b_{-2} \cdot p^{-1} + \ldots + b_{-M} \cdot p^{-M+1}$$

kifejezést kapjuk. Mivel b_{-1} lehetséges értékei $0,1,\ldots,p-1$,

a

2.17. egyenlet - 17_egyenlet

$$b_{-2} \cdot p^{-1} + \ldots + b_{-M} \cdot p^{-M+1}$$

pedig kisebb, mint *1*, ezért a

2.18. egyenlet - 18_egyenlet

$$b_{-1} + b_{-2} \cdot p^{-1} + \ldots + b_{-M} \cdot p^{-M+1}$$

egész része b_{-1}, a tört része pedig

2.19. egyenlet - 19_egyenlet

$$b_{-2} \cdot p^{-1} + \ldots + b_{-M} \cdot p^{-M+1}$$

Ha ez utóbbit *p*-vel szorozzuk, az eredmény egész része b_{-2}. Ezt az eljárást addig folytatjuk, míg a tört rész nulla lesz vagy a b_{-i} együtthatók szakaszos ismétlődést mutatnak.

3. A konkrét példában az átírást egy véges tizedes törttel rendelkező számra végeztük, ami racionális, és tudjuk, hogy egy racionális szám nem lehet más, mint véges vagy végtelen szakaszos tört. A szakaszos ismétlődésre példa a $123,45_{10}$ konverziója hármas számrendszerbe:

$123,45_{10}=11120.1100,1100,\ldots_3$.

Az eredmény végtelen szakaszos harmados tört. Természetes követelmény, hogy a végtelen szakaszos előállításból is vissza tudjuk állítani a szám pontos értékét.

Megoldás:

Csak a tört résszel foglalkozunk, és használjuk a végtelen geometriai sor összegképletét. Nevezetesen, ha $p \leq 1$, akkor

2.20. egyenlet - 20_egyenlet

$$\sum_{i=0}^{\infty} p^i = \frac{1}{1-p}.$$

Az ismétlődő szakasz *1100*, tehát a periódusa *4*, ezért

2.21. egyenlet - 21_egyenlet

$$\sum_{i=0}^{\infty} 3^{-1-4i} + \sum_{i=0}^{\infty} 3^{-2-4i} = \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^{1+4i} + \sum_{i=0}^{\infty} \left(\frac{1}{3}\right)^{2+4i} = \frac{1}{3} \cdot \left(\frac{1}{1-\left(\frac{1}{3}\right)^4}\right) + \frac{1}{9} \cdot \left(\frac{1}{1-\left(\frac{1}{3}\right)^4}\right) =$$

2.22. egyenlet - 22_egyenlet

$$\frac{3^3}{3^4-1} + \frac{3^2}{3^4-1} = \frac{\left(3^3+3^2\right)}{\left(3^4-1\right)} = \frac{36}{80} = \frac{9}{20} = 0,45$$

Ebben a számolásban a p értéke $(1/3)^4$ volt. Nem véletlenül, hiszen a periódus *4*.

Megjegyzés: A tört rész nem kezdődik általában a periódussal. Például a *.01, 1100, 1100, …*$_2$ esetében a kettedes pont utáni *.01*-gyel külön kell foglalkozni.

4. A fenti gondolatmenetet megismételjük általánosan. Adott egy végtelen szakaszos q-ados tört előállítás:

$.b_{-1}b_{-2} \ldots b_{-u}, a_{-(u+1)}a_{-(u+2)} \ldots a_{-(u+K)}, a_{-(u+K+1)}a_{-(u+K+2)} \ldots a_{-(u+2*K)}, \ldots$

ahol az első u darab b_{-j} szám nem tarozik a szakaszhoz, és a K periódus szerint ismétlődő számokra teljesül, hogy $a_{-(u+K)} = a_{-(u+K)}$.

Általában $a_{-i} = a_{-j}$, ha i-u-nak és j-u-nak a K-val való osztási maradéka megegyezik. A .01, 1100, 1100, …$_2$ tört esetében például $u = 2$, $K = 4$, $b_{-1} = 0$, $b_{-2} = 1$ *és* $a_{-(u+1)} = 1$, $a_{-(u+2)} = 1$, $a_{-(u+3)} = 0$, $a_{-(u+K)} = 0$.

Ha a periódusból a második elemet tekintjük, akkor kapjuk, hogy

2.23. egyenlet - 23_egyenlet

$$a_{-(u+2)} \cdot q^{-(u+2)} + a_{-(u+K+2)} \cdot q^{-(u+K+2)} + a_{-(u+2K+2)} \cdot q^{-(u+2K+2)} + \ldots = a_{-(u+2)} \cdot q^{-(u+2)} + a_{-(u+2)} \cdot q^{-(u+K+2)} + a_{-(u+2)} \cdot q^{-(u+2K+2)} + \ldots =$$

2.24. egyenlet - 24_egyenlet

$$a_{-(u+2)} \cdot \sum_{j=0}^{\infty} q^{-(u+jK+2)} = a_{-(u+2)} \cdot q^{-(u+2)} \cdot \sum_{j=0}^{\infty} q^{-jK} = a_{-(u+2)} \cdot q^{-(u+2)} \cdot \sum_{j=0}^{\infty} \left(q^{-K}\right)^j = \frac{a_{-(u+2)} \cdot q^{-(u+2)}}{1-q^{-K}}.$$

Ezt megismételve a többi periódusban szereplő együttható esetére, azt kapjuk, hogy

2.25. egyenlet - 25_egyenlet

$$.b_{-1}b_{-2} \ldots b_{-u}, a_{-(u+1)}a_{-(u+2)} \ldots a_{-(u+K)}, a_{-(u+K+1)}a_{-(u+K+2)} \ldots a_{-(u+2K)}, \ldots = \sum_{j=1}^{u} b_{-j} \cdot q^{-j} + \frac{1}{1-q^{-K}} \cdot \sum_{k=1}^{K} a_{-(u+k)} \cdot q^{-(u+k)}$$

A *.01, 1100, 1100, …*$_2$ példára ezt a képletet alkalmazva kapjuk:

2.26. egyenlet - 26_egyenlet

$$.01,1100,1100,\ldots_2 = \frac{0}{2} + \frac{1}{4} + \frac{\frac{1}{8} + \frac{1}{16} + \frac{0}{32} + \frac{0}{64}}{1 - \frac{1}{16}} = .25 + \left(\frac{1}{8} + \frac{1}{16}\right) \cdot \frac{16}{15} = .25 + \frac{3}{15} = .45$$

Feladatok 1.

1. Egyértelműek-e az u, K, b_{-i}, a_{-j} számok a felírásban?

Vegyük észre, hogy

.01, 1100, 1100, $\ldots_2$= .011, 1001, 1001, $\ldots_2$ = .01, 11001100, 11001100,$\ldots_2$.

2. Az 1 táblázatban mi a tízes számrendszerbeli értéke a 10-nek a különböző oszlopokban?

Az alábbi példákban és a feladatokban csak a számítástechnikában használatos számrendszerekre szorítkozunk.

Példák

1. Mi lesz a tizenhatos számrendszerbeli alakja az 11110100111111010_2 számnak?

Megoldás:

A kettes számrendszerbeli szám számjegyeit jobbról balra haladva négyes csoportokra osztjuk, és megadjuk a csoportokhoz tartozó tizenhatos számrendszerbeli számokat. (Ha az utolsó számnégyes nem teljes, akkor nullákkal egészítjük ki az elejét.)

$0011\,1101\,0011\,1111\,1010_2$ 2 → $3D3FA_{16}$

2. Mi lesz a kettes számrendszerbeli alakja az $1AC95_{16}$ számnak?

Megoldás:

Megadjuk a tizenhatos számrendszerbeli szám számjegyeinek megfelelő kettes számrendszerbelieket:

$1AC95_{16}$ → $1\,1010\,1100\,1001\,0101_2$

A fenti példákban a bináris és a hexadecimális számok közötti azon speciális kapcsolatot (egy hexadecimális szám *4* biten ábrázolható) alkalmaztuk, miszerint a számjegyek átírása egyben a szám konverzióját is jelenti. (Egyszerű ellenpéldával igazolhatjuk, hogy a bináris és decimális számok között nincs ilyen kapcsolat.) Az átírás jogosultságát az alábbiakban igazoljuk:

2.27. egyenlet - 27_egyenlet

$$\sum_{k=-m}^{n} a_k \cdot 2^k = \sum_t \sum_{l=0}^{3} a_{4t+l} \cdot 2^{4t+l} = \sum_t 2^{4t} \sum_{l=0}^{3} a_{4t+l} \cdot 2^l = \sum_t 16^t \cdot b_t$$

Feladatok

1. *Felvetődik a kérdés, hogy milyen más esetekben működik az utóbbi módszer? Próbáljuk ki az oktális esetet!*
2. *Tetszőleges számrendszerben adott számú pozíción melyik a legnagyobb és legkisebb leírható szám?*
3. *A (11_egyenlet) összegeit írjuk fel a Σ jel segítségével!*
4. Bizonyítsuk be, hogy

2.28. egyenlet - 28_egyenlet

$$b_{-2} \cdot p^{-1} + \ldots + b_{-M} \cdot p^{-M+1} < 1$$

minden $0 \leq b_i \leq p$ esetén!

5. *Konvertáljuk tízes számrendszerbe az alábbi számokat:*

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

6. *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.

7. *Konvertáljuk bináris számrendszerbe az alábbi decimális számokat:*

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

8. *Konvertáljuk hexadecimális számrendszerbe az alábbi decimális számokat:*

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

9. *Bizonyítsuk be, hogy bináris és oktális szám között is alkalmazható a számjegyenkénti átírás! Milyen csoportokat kell képezni?*

3. 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:

```
 1001.01                1001.11
+1001.10               -1001.10
 1010.11                  0.01
```

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

   Megoldás:

```
ABCD.EF                1AB2C.2
+1923.7A               -AB3C.25
 C4F1.69                FFEF.FE
```

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.

3. fejezet - A számítógép mint adatfeldolgozó eszköz

1. Történeti áttekintés

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

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

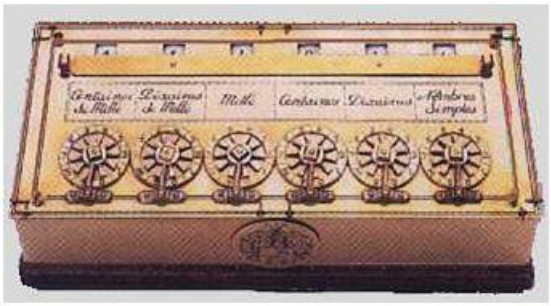
Talán az első igazi újkori matematikai fogalomalkotás, amely a görögök és az arabok számára elképzelhetetlen lett volna, a logaritmus fogalmának megadása volt. Egyszerre két tudós is foglalkozott vele, egymástól függetlenül: *John Napier* (1550–1617) skót báró és *Jobst Bürgi* (1552-1623) svájci órásmester és matematikus. Jost Bürgi készítette az első logaritmustáblázatot, de nem publikálta időben, így Napier táblázata vált előbb ismertté.

A logaritmus fogalma létrejöttének köszönhető mechanikus segédeszköz a logarléc. 1622-ben *William Oughtred* (1574-1664) alkalmazott először logaritmus skálát a két, egymáson elcsúsztatható vonalzókon. 1650-ben készítette *Pattridge* 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, négyalapműveletes masinát szerkesztett. A gép működésének elve a John Napier által készített Napier-csontok számolási eljárásait követi. A szorzás műveletének megkönnyítésére Napier feltalált egy, elefántcsont rudakból álló számolószerkezetet, amelyet Napier-pálcáknak, vagy Napier-féle csontoknak neveztek. Ez a logarléc elődjének tekinthető. Schickard gépe (lásd a következő ábrát) számtárcsákkal tárolja a részeredményeket, és a túlcsordulást egy kis csengő megszólaltatásával jelzi.

8. Ábra

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



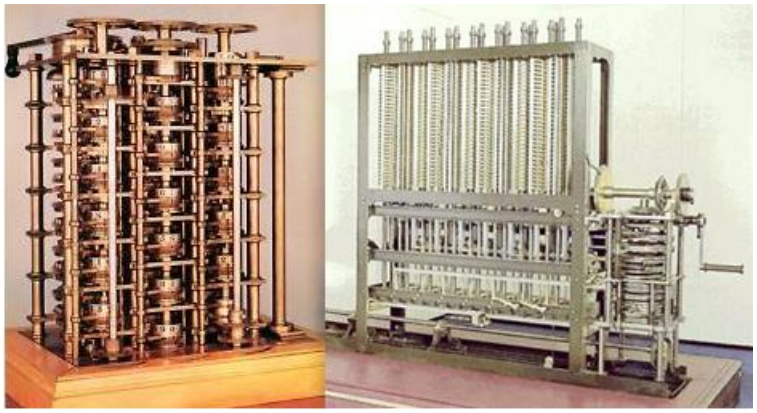
9. Ábra

Gottfried Wilhelm Leibnitz (1646-1716) német tudós 1672-ben mechanikus számológépet (lásd a következő ábrát) épített. Az első, valódi négyalapműveletes 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.

10. Ábra

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.

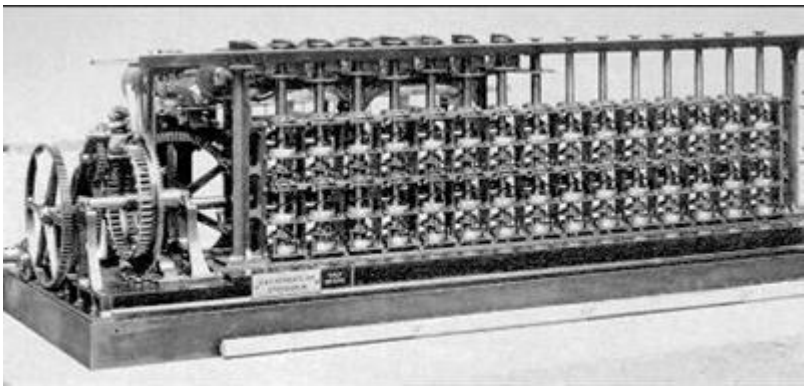
1820-as évek elején *Charles Babbage* (1782-1871) megtervezte a Difference Engine-t (differenciagépet), amely logaritmus táblázatok pontos és gyors elkészítését tette lehetővé. A bal oldali képen a Differencial Engine számolóművének 1832-ben összeszerelt részlete látható, a jobb oldalin pedig a londoni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



11. Ábra

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. Ennek lényege a szukcesszív differencia (successive difference), azaz az egymást követő különbségek képzése, amit a fejezet végén egy példán keresztül mutatunk be.



12. Ábra

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.

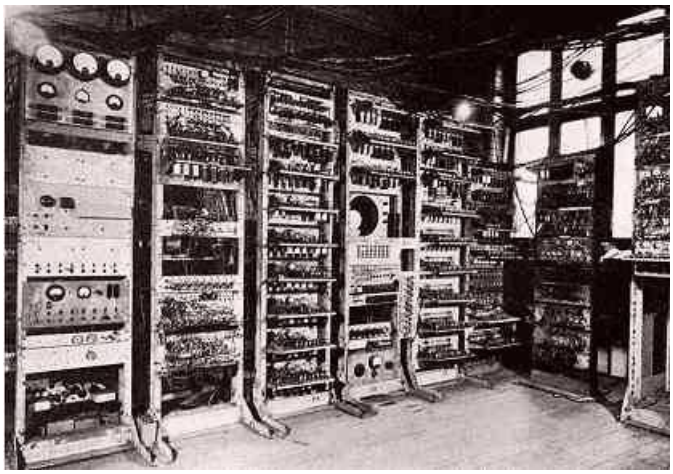
Augusta Ada King (született Lovelace Byron) grófnő (1815–1852) főként arról ismert, hogy leírást készített a Charles Babbage által tervezett Analitycal Engine-hez.

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.



13. Ábra

Példa

Ha az n^2–et akarjuk kiszámítani, akkor a szorzás műveletet kiválthatjuk az n-nél kisebb négyzetszámok különbségeivel.

Megoldás:

Tekintsük a négyzetszámokat: *0, 1, 4, 9, 16, 25, ...* Ezek különbsége rendre: *1, 3, 5, 7, 9, ...* Nem nehéz észrevenni, hogy ha az így kapott sorozat különbségei vesszük, mindig *2* lesz az eredmény. Ez általában is igaz:

3.1. egyenlet - 29_egyenlet

$$[n^2 - (n-1)^2] - [(n-1)^2 - (n-2)^2] = 2.$$

Ebből n^2–et kifejezve kapjuk, hogy

3.2. egyenlet - 30_egyenlet

$$n^2 = (n-1)^2 + [(n-1)^2 - (n-2)^2] + 2.$$

Konkrétan legyen $5^2 = 4^2 + 7 + 2 = 25$. A képletünket alkalmazhatjuk a 4^2–re is, majd a 3^2–re, és így tovább, míg el nem jutunk a *0*-nak és az *1*-nek a négyzetéhez. Az algoritmus persze a *0*-nak és az *1*-nek a négyzetéből indul és számítja a 2^2–t és így tovább. A többszörös differencia képzés módszerét gyakran használják függvények értékeinek kiszámítására is. Többek között a polinomok értékeit tudjuk így számítani.

Feladatok

1. *Az n^3 kiszámításához adjuk meg a szukcesszív differencia algoritmust! (Képezzük a differenciát háromszor!)*

2. *Az n^2–hez 2-szer kell differenciát képezni és a konstans 2, az n^3 kiszámításához háromszor és a konstans 6. Mi a kapcsolat a hatvány, a differenciálás rendje és a konstans között?*

2. 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 Pennsylvaniai 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.

14. Ábra

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.

15. Ábra

Neumann és Goldstine személyesen először 1944-ben találkozott. 1948-ban megfogalmazták az elektronikus digitális számítógépekkel, az úgynevezett Neumann-elvű gépekkel szembeni követelményeket. (John von Neumann, First Draft of a Report on the EDVAC, M.D. Godfrey and D.F. Hendry, The Computer as von Neumann Planned It," IEEE Annals of the History of Computing, Vol. 15, No. 1, 1993, pp. 11-21.)

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 hajtsa 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 tárakat 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 ősének tekinthető MONITOR. Ez egy, a memóriában tartózkodó 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 meghaladhatják 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. Bár már 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 távlati 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. A hardver és szoftver mellett egyre inkább a firmware (még magyar írásmódja sincs) kerül előtérbe, ami egy olyan szoftverfajta, amely a hardvereszközbe van beépítve, és a hardver működtetéséhez szükséges legalapvetőbb feladatokat látja el.

4. fejezet - 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* - *bi*nary digi*t*. 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.

1. Számábrázolás

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

1.1. Fixpontos számábrázolás

Ez a számábrázolási mód minden számot tizedes vessző (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:

	0	0	1	1	1	1	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:

egészrész	törtrész

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 (vagy többletes) ábrázolásnál a kettes számrendszerbeli értékből ki kell vonnunk egy megállapodás szerinti értéket.

Például 8 bit esetén:

00000000= -128

00000010=-126

10000000=0

11000000=64

Komplemens vagy egyes komplemens ábrázolás esetén a pozitív számot binárisan adjuk meg, a negatívot bitenként negáljuk.

Például *8* bit esetén: *11111110= -1*

A kettes komplemens ábrázolás esetén a pozitív számot binárisan adjuk meg, a negatívot bitenként negáljuk, majd hozzáadunk *1*-et.

Például *8* bit esetén: *11111111= -1*

Az előjeles abszolútérték és az egyes komplemens esetén a *0* kétféleképpen ábrázolható, továbbá a műveletek elvégzése is nehézkes. A kettes komplemens ábrázolás megoldja ezeket a problémákat. Nézzük először a *0* egyértelműségét. Ha összeadunk egy n bites bináris számot az inverzével, akkor eredményként egy olyan számot kapunk, amelynek minden bitje *1*. A bináris szám inverzét úgy képezzük, hogy a bitek tartalmát ellenkezőjére változtatjuk: a *0*-át *1*-re, az *1*-et *0*-ra.

Példa

Legyen *N* egy természetes szám, ami a bináris alakjából

4.1. egyenlet - 31_egyenlet

$$N = \sum_{i=0}^{n} a_i \cdot 2^i,$$

Az egyes komplemense

4.2. egyenlet - 32_egyenlet

$$N = \sum_{i=0}^{n} \mathring{a}_i \cdot 2^i,$$

az összegük pedig

4.3. egyenlet - 33_egyenlet

$$\sum_{i=0}^{n} a_i \cdot 2^i + \sum_{i=0}^{n} \mathring{a}_i \cdot 2^i = \sum_{i=0}^{n} 2^i = 2^{n+1} - 1.$$

Tehát, ha a bináris számhoz hozzáadjuk az egyes komplemensét és még *1*–et, akkor egy olyan bináris számot kapunk, amelyiknek az *(n+1).* helyén *1*-es áll, a többi helyen nulla.

Végezzük ezt el *8* biten, ahol a *8*. bit az előjelbit. A *N* szám, a komplemense és még *1* összeadásának eredménye egy olyan szám, amelyben a *8*. biten *1*-es, a többi helyen nulla áll. Ha a komplemens + *1*-et negatív *N*-nek *(-N)* definiáljuk, és az előjel bitjét *1*-esre állítjuk, az összeadás eredménye csupa nulla lesz (a túlcsordulás miatt!). A helyes definíciója a *-N*–nek tehát a *N* egyes komplemense *7* biten + *1*, és az előjelbit *1*-es. A *-N* ábrázolása a *N*-nek megfelelő bitsorozatból úgy is elvégezhető, hogy jobbról indulva az első *1*-es bitig, azt is beleértve, másoljuk, a további biteket pedig az ellenkezőjére változtatjuk.

Ha például $N = 25$, akkor

$25_{10} = 00011001_2$,

az egyes komplemense 11100110_2

a kettes komplemense pedig $-25_{10} = 11100111_2$.

A kettes komplemens esetén a nulla egyértelműen ábrázolható. A legkisebb ábrázolható szám *8* biten *-128*, a legnagyobb *127*. Az összeadás bináris számrendszerben jegyenkénti átvitellel ugyanúgy végezhető el, mint a decimális esetben:

011	+6	0110	+3	0011	-:
010	-3	1101	-6	1010	-:
101	+3	0011	-3	1101	-:

Az összeadás szabálya érvényesült. Például: *3-6=3+(-6).*

1.2. Lebegőpontos számábrázolás

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

$szám = (-1)^s M p^k$

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

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

$szám = (-1)^s (1.M) (2^{k-127})$

alakban adjuk meg, és az alábbi módon ábrázoljuk:

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 kettedes pont helyét. A karakterisztikát eltolt nullpontú (vagy többletes) 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:

$szám = (-1)^s (1+M) (2^{k-127})$

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 10100000000000000000 módon, 32 biten?

Megoldás:

Látható, hogy

s = 1;

$k = 10000111_2 = 135_{10}$;

$M = 0.101_2 = 0{,}625_{10}$,

tehát

$x = -1{,}625*2^8 = -1{,}625*256 = -416$.

Feladatok

1. *Mely számot ábrázoltuk 32 biten az alábbi módokon?*

   0 01111111 00000000000000000000000

   0 01110101 01010100000000000000000

   0 10110000 10101000000000000000000

   1 00101010 11100000000000000000000

   1 10000010 00010100000000000000000

2. *Ábrázoljuk a következő decimális számokat:* 1; 300; -8,625; -16,125; 32,25.

1.3. Kódolt számábrázolás

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=1byte):

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	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx
(null)	32	20	040		Space	64	40	100	@	@	96	60
(start of heading)	33	21	041	!	!	65	41	101	A	A	97	61
(start of text)	34	22	042	"	"	66	42	102	B	B	98	62
(end of text)	35	23	043	#	#	67	43	103	C	C	99	63
(end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64
(enquiry)	37	25	045	%	%	69	45	105	E	E	101	65
(acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66
(bell)	39	27	047	'	'	71	47	107	G	G	103	67
(backspace)	40	28	050	(	(	72	48	110	H	H	104	68
(horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69
(NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A
(vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B
(NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C
(carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D
(shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E
(shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F
(data link escape)	48	30	060	0	0	80	50	120	P	P	112	70
(device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71
(device control 2)	50	32	062	2	2	82	52	122	R	R	114	72
(device control 3)	51	33	063	3	3	83	53	123	S	S	115	73
(device control 4)	52	34	064	4	4	84	54	124	T	T	116	74
(negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75
(synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76
(end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77
(cancel)	56	38	070	8	8	88	58	130	X	X	120	78
(end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79
(substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A
(escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B
(file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C
(group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D
(record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E
(unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F

16. Ábra

Példa egy kiterjesztésre: *A 437-es kódlap*

44	É	161	í	177	▒	193	┴	209	╤	225	ß
45	æ	162	ó	178	▓	194	┬	210	╥	226	Γ
46	Æ	163	ú	179	│	195	├	211	╙	227	π
47	ô	164	ñ	180	┤	196	─	212	╘	228	Σ
48	ö	165	Ñ	181	╡	197	┼	213	╒	229	σ
49	ò	166	ª	182	╢	198	╞	214	╓	230	µ
50	û	167	º	183	╖	199	╟	215	╫	231	τ
51	ù	168	¿	184	╕	200	╚	216	╪	232	Φ
52	_	169	⌐	185	╣	201	╔	217	┘	233	Θ
53	Ö	170	¬	186	║	202	╩	218	┌	234	Ω
54	Ü	171	½	187	╗	203	╦	219	▉	235	δ
56	£	172	¼	188	╝	204	╠	220	▄	236	∞
57	¥	173	¡	189	╜	205	═	221	▌	237	ϕ
58	_	174	«	190	╛	206	╬	222	▐	238	ε
59	ƒ	175	»	191	┐	207	╧	223	▀	239	∩
60	á	176	▒	192	└	208	╨	224	α	240	≡

17. Ábra

A szoftverek nemzetközivé válása során kiderült, hogy a sokféle karakterkészlet a számítástechnika fejlődésének egyik gátlója. Megoldást egy olyan kódolás adhat, amely képes az összes nyelv összes karakterét ábrázolni. Ezért született meg a *16* bites Unicode (UCS - Universal Character Set).

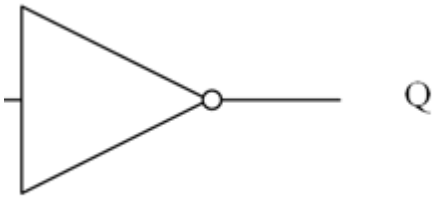
Az összes korábbi *8* bites kódkészletben megtalálható karakter belefért a Unicode kezdeti alsó *65536*-os tartományába, amelyet Basic Multilingual Plane-nek (BMP) is neveznek. Az alsó *128* érték megegyezik a hagyományos ASCII-val. Sőt, az alsó *256* megegyezik a Latin-1-gyel (az ASCII ékezetes betűs bővítése). A magyar ő és ű betűk tehát *256*-nál nagyobb azonosítót kaptak. A Unicode értékeket általában hexadecimálisan, nagy ritkán decimálisan adjuk meg. A különféle egzotikusabb betűírásokon (cirill, héber, arab stb.) túl tartalmazza a kínai, japán, koreai (ezeket együtt szokták angolul CJK-nak rövidíteni) írásjeleket, és számos vezérlő karaktert, melyekkel például a jobbról balra írás kapcsolható be és ki, vagy éppen a sortörés lehetséges helyei adhatók meg.

2. Műveletek a számítógépen

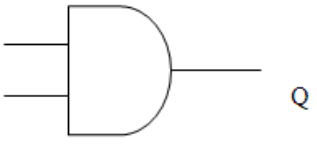
A digitális számítógépeket a kettes számrendszer alapján építik, mivel egy számjegy egy kétállapotú egységgel megvalósítható. A digitális logikai szintet a kapuáramkörök alkotják, amik analóg alkatrészekből épülnek fel, de működésükkel a bináris rendszer alapját képezik. A két állapot megkülönböztetésére két jelszintet alkalmaznak: az alacsony a hamis vagy *0* értéket, a magas az igaz vagy *1* értéket jelenti.

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, 2^n kimenete van, amik egy 2^n soros táblázattal adható meg, amit igazságtáblának nevezünk. Az egyváltozós művelet, ami megfordítja a bemenet (A) értékét, azaz a kimenet (Q) *0*-ra *1*-et, *1*-re *0*-t ad, a negáció, **NEM**, vagy angolul **NOT**, függvény logikai áramköri rajzjele és **igazságtáblája** az alábbi:

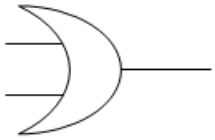


A kétváltozós műveletek mindkét bemenete kétféle lehet. A bemenetek közötti műveletek az ÉS, VAGY és KIZÁRÓ VAGY. Az **ÉS**, vagy angolul **AND** művelet a konjunkció, ami csak akkor ad egyet, ha az egyik (A) és a másik (B) bemenete is egy.



A	B
0	0
0	1
1	0
1	1

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



A	B	
0	0	
0	1	
1	0	
1	1	

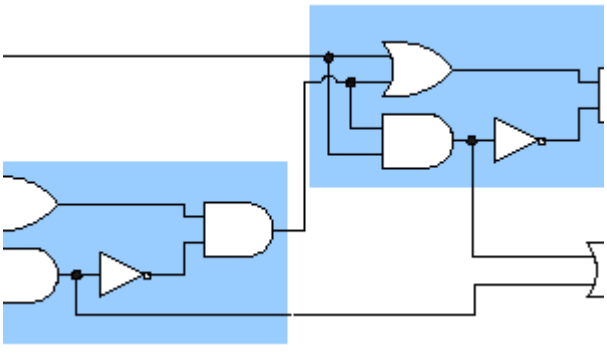
Ezekkel az alapműveletekkel már a többi kétváltozós művelet is megadható. Például az előző műveletekkel kifejezve a **KIZÁRÓ VAGY**, vagy **XOR** művelet akkor ad *1*-et, ha a két bemenete (A, B) különböző:

A **XOR** B = (A **AND NOT** B) **OR** (**NOT** A **AND** B).

Az **XOR** művelet jelentősége az összeadás műveletének logikai kapukkal történő megvalósításában mutatkozik meg. Ha összeadunk két egy-egy biten ábrázolt digitalis értéket *x*–et és *y*-t akkor az eredmény *s* (sum) megjelenik egy biten és kapunk még egy *c* (carry) továbbvivendő bitet. Az alábbi táblázat s oszlopa pontosan az XOR műveletnek felel meg.

y	*s*	
0	*0*	
1	*1*	
0	*1*	
1	*0*	

Az alábbi ábra pedig mutatja a teljes összeadás egy lépését, amikor nemcsak a két összeadandó van hanem a korábbi összeadásból származó *c* is.



18. Ábra

32 biten az összeadás a fenti áramkörök összefűzéséből végezhető el

x_{31} x

y_{31} .... y

---

s_{31} .... s

Az utolsó sorban a *c* a túlcsordulás áldozata lesz. Ha *c* nem *0* akkor értékes jegy vész el és az eredmény természetesen hibás lesz.

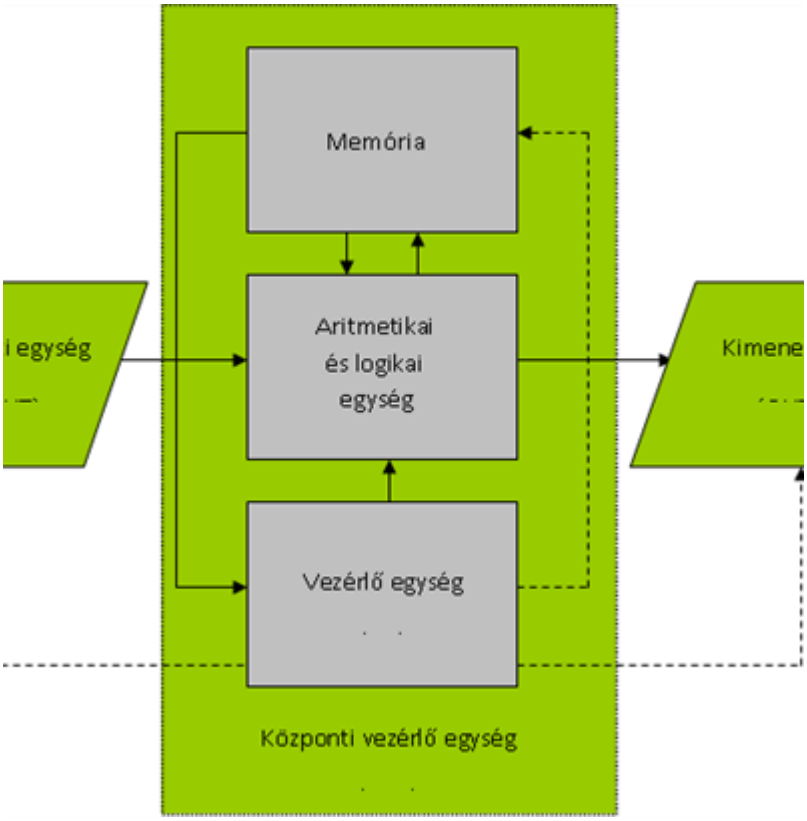
Feladat:

*1. Igaz-e, hogy A **XOR** B = (A+B)*NOT(A*B)*

2. Adjuk meg a teljes összeadás logikai sémáját.

5. fejezet - A számítógép felépítése

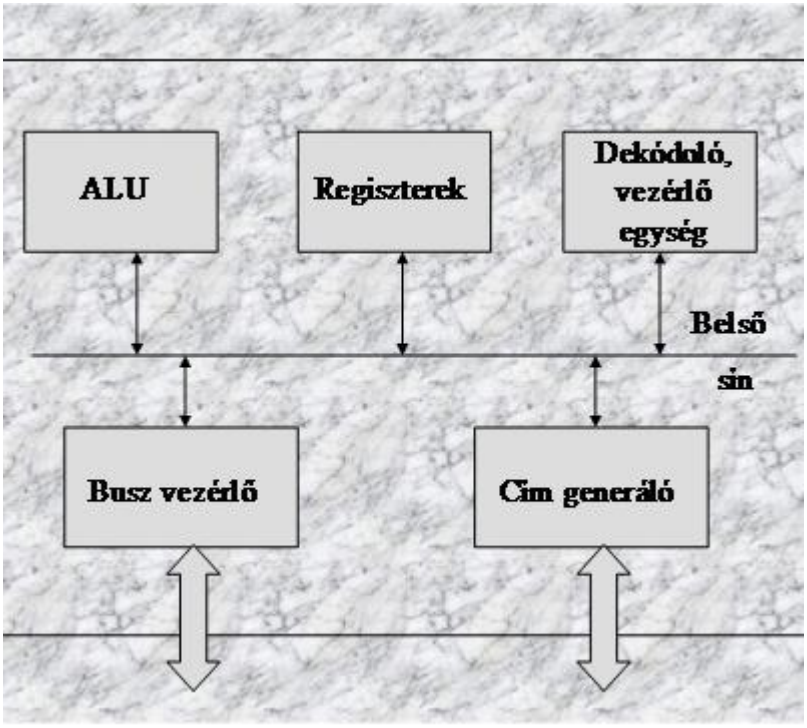
A számítógép működésének megértéséhez szükséges, hogy ismerjük a hardver felépítését, és tisztában legyünk a hardverelemek funkcióival. A következő ábra a számítógép funkcionális felépítését, a Neumann-modellt mutatja:



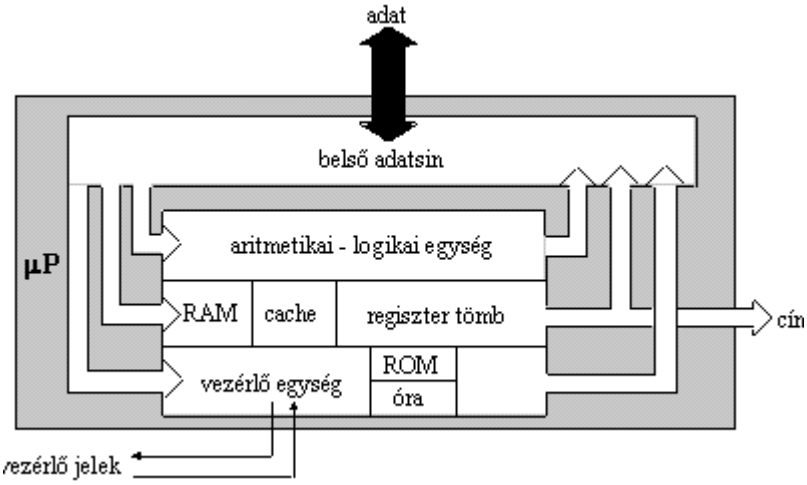
19. Ábra

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.



20. Ábra



21. Ábra

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) és a FLOPS (Floating Point Operation 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.

1. 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 ≈ 10^3 byte);

1 **Mbyte** (megabyte) = 1024 kbyte (1024 x 1024 byte = 1 048 576 byte ≈ 10^6 byte);

1 **Gbyte** (gigabyte) = 1024 Mbyte (1024 x 1024 x 1024 byte = 1 073 741 824 byte ≈ 10^9 byte);

1 **Tbyte** (terabyte) = 1024 Gbyte (1024 x 1024 x 1024 x 1024 byte = 1 078 036 791 296 byte ≈ 10^{12} 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, akárhány 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** (Electrononically Erasable PROM) elektromos úton, a processzor utasításai alapján törölhető és programozható át, tehát nem kell a gépből kiszerelni 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.)

2. 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) **alrendszert** 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).

USB

Az **USB** (Universal Serial Bus) olyan csatolási 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.

22. Ábra

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, üzem közbeni illeszthetőségével, az illesztett eszközök önműködő beállításával kategóriájában egyedülálló illesztési szabvány.

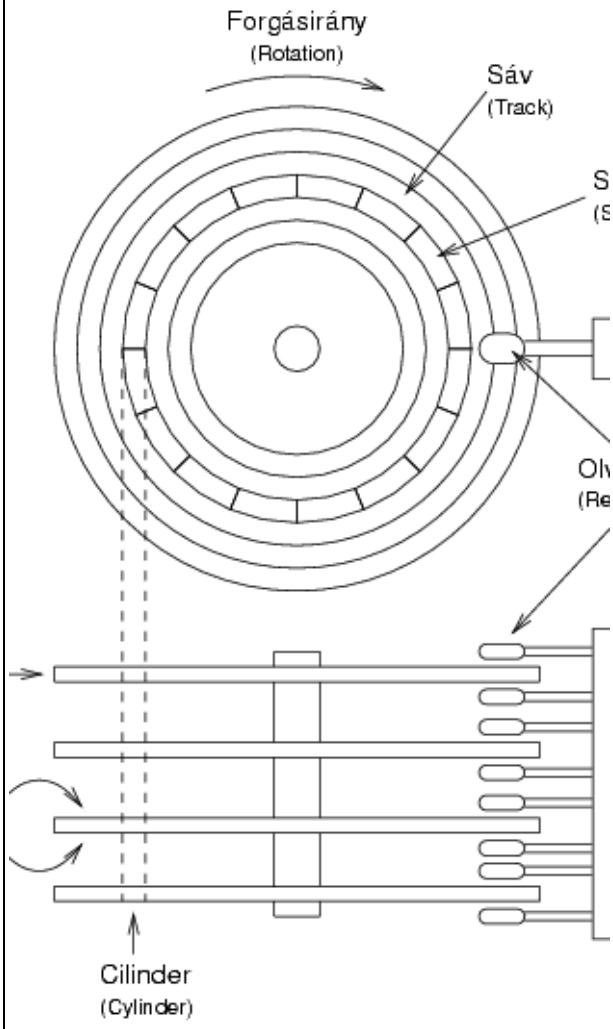
23. Ábra

2.1. 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 **mágneslemezeken.** 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.



24. Ábra

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.

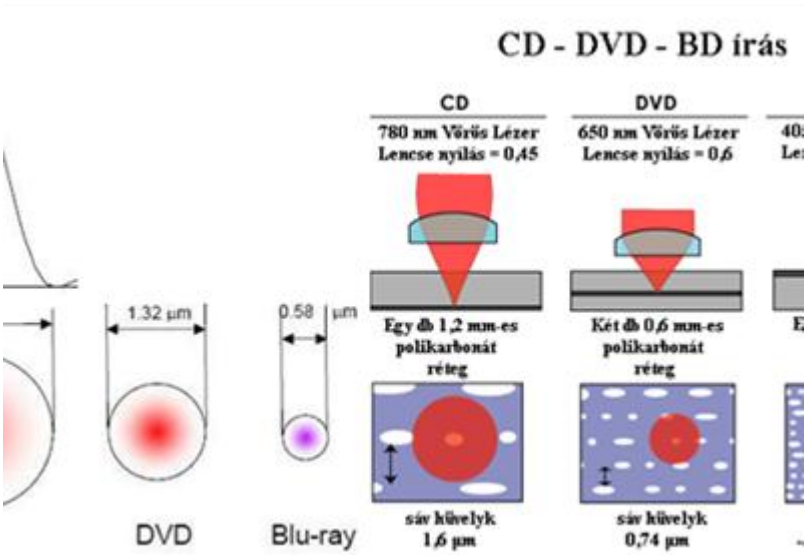
Az optikai háttértárolók

Az optikai elven működő háttértárak 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-vel, a sávok azonban sokkal sűrűbben helyezkednek el.
- A **Blu-ray Disc** (**BD** vagy **BR**) nagy tároló kapacitású digitális optikai tárolóeszköz-formátum. Az elnevezésben a „blue” (kék) a használt lézer színére utal, a „ray” pedig az optikai sugárra. (Az „e” betűt a „blue” szóból azért hagyták el, mert egy hétköznapi szót nem lehet levédeni.)

A CD, DVD és BD írásának összehasonlítása:



25. Ábra/a 25. Ábra/b

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.

26. Ábra

Pendrive, flashdrive

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

27. Ábra

2.2. 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 megjelentetésére is képessé tettek. 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 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.

A képen egy João Sabino által tervezett billentyűzet táska látszik:

28. Ábra

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.

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

29. Ábra

2.3. Egyéb beviteli eszközök

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

30. Ábra

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

31. Ábra

A **digitalizáló táblát** egy adatbeviteli eszköz, amely lehetővé teszi, hogy közvetlenül, kézzel vigyünk be adatokat a számítógépbe úgy, mintha papírra írnánk vagy rajzolnánk.

32. Ábra

Az **interaktív tábla** mind az üzleti szférában, mind az oktatásban használt olyan bemeneti eszköz, amely egy szoftver segítségével kapcsolja össze a táblát egy számítógéppel (és egy projektorral) úgy, hogy annak vezérlése a tábláról is történhet, illetve a táblára került tartalmak háttértárolóra menthetővé válnak.

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

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

3. 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 begépelt 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 voltak. 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 videókártyákat. 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 szolgáltattak 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 szín 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:

- katódsugárcsöves (CRT);
- plazmakijelzős (LED);
- folyadékkristályos(LCD);
- vékonyfilm tranzisztoros(TFT).

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

Nyomtatók

Amióta számítógépek vannak, 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:

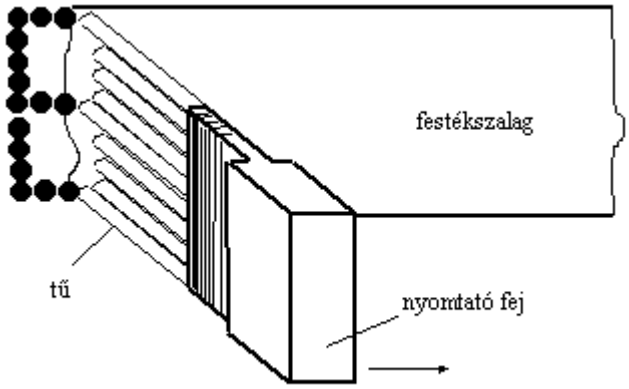
- karakternyomtatók;
- sornyomtatók;
- lapnyomtatók.

Az írásminőség alapján:

- levélminőségű (LQ = Letter Quality);
- közel levélminőségű (NLQ = Near Letter Quality);
- piszkozati minőségű (Draft).

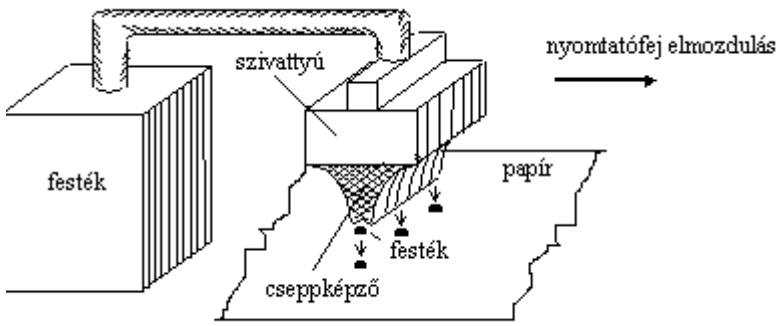
A napjainkban leggyakrabban használt nyomtatók a mátrix-, a tintasugaras és a lézernyomtatók.

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).



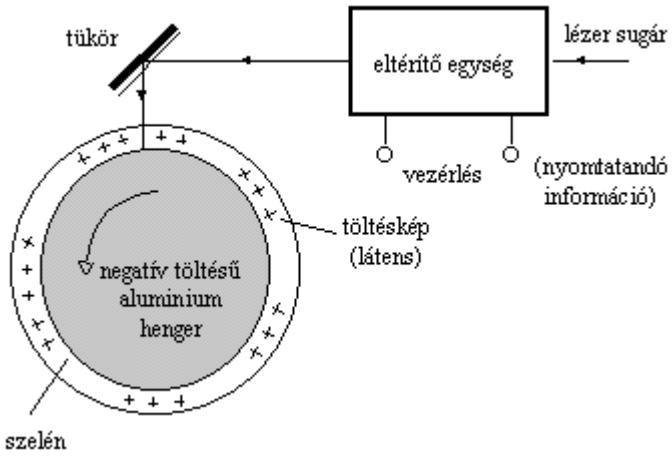
33. Ábra

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 szivacsszerű 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.



34. Ábra

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 festékpor (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.



35. Ábra

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.

36. Ábra

6. fejezet - 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

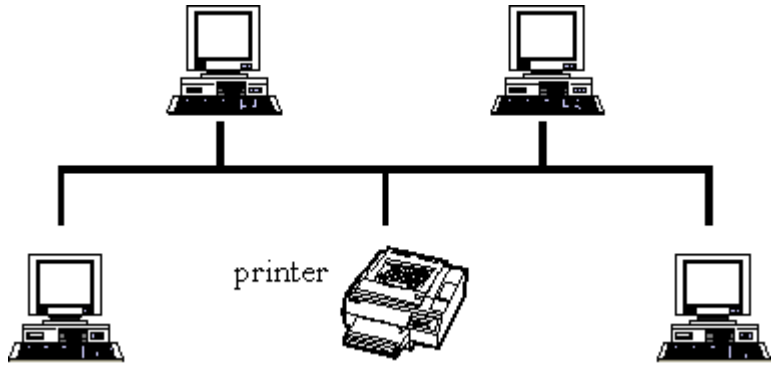
A legegyszerűbb és legolcsóbb hálózati elrendezés a sín vagy busz topológia. Ez az elrendezés egyetlen, busznak nevezett átviteli közeget használ. A buszon lévő mindegyik számítógépnek egyedi címe van, ez azonosítja a hálózaton.

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 lévőkkel, 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 kábelen halad, egyre inkább gyengébb lesz. Ha több számítógépet csatlakoztatnunk 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:



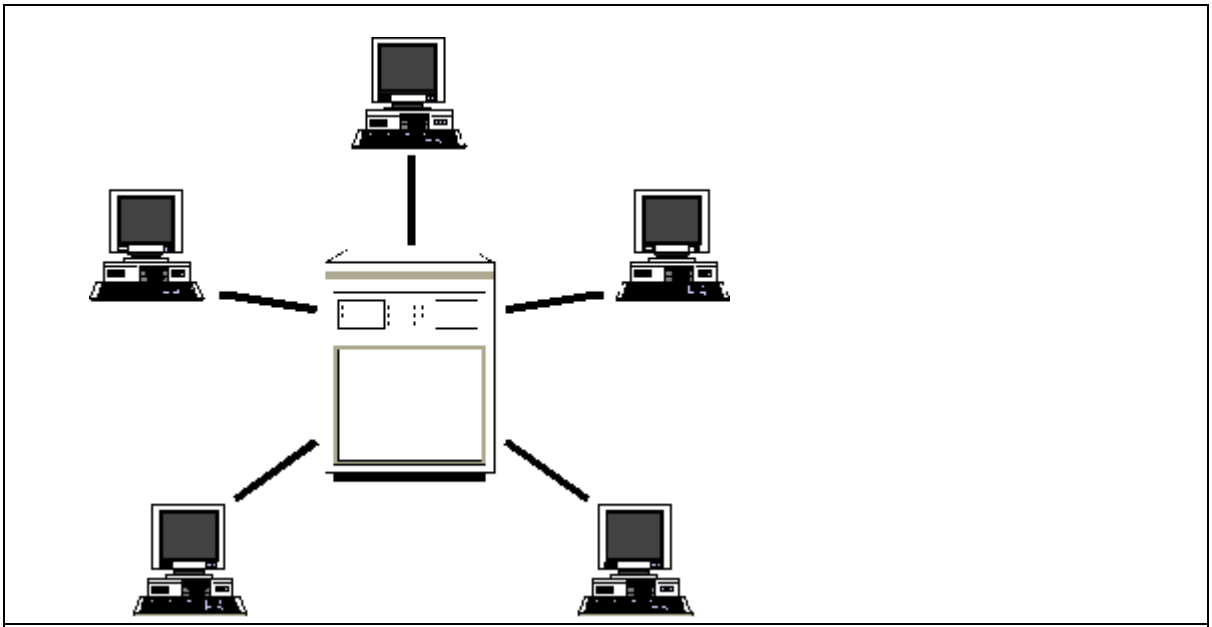
37. Ábra

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 vezérlőnek 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:



38. Ábra

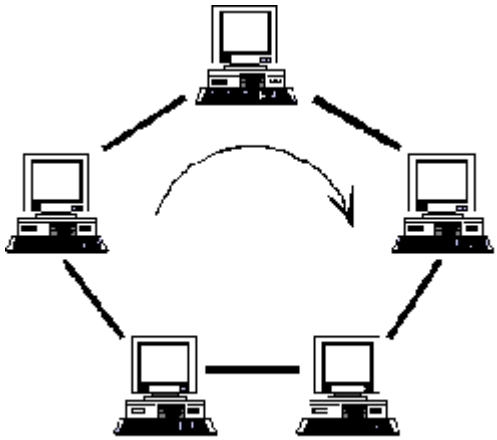
Gyűrű topológia

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 szólt, 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.

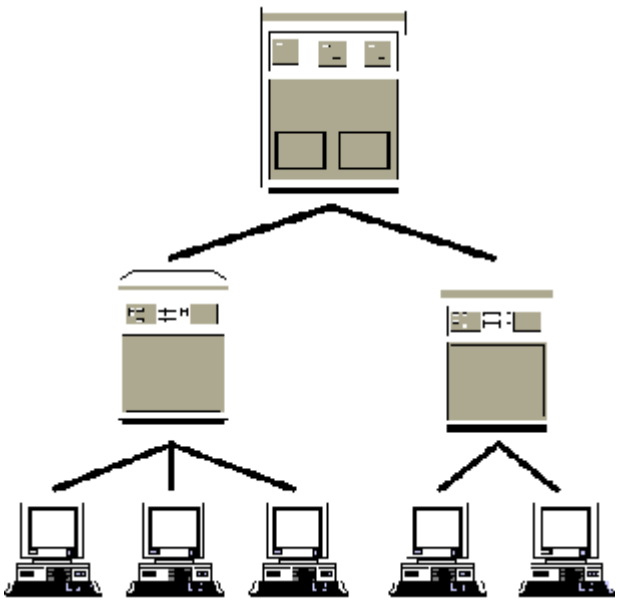
A gyűrű alakú topológia esetén a hálózati kommunikáció lehet csomagkapcsolt és vezérjel (Token Ring) elve alapján működő. Ez utóbbi esetben egy vezérjel kering körbe a vonalon, és csak az a gép küldhet üzenetet, amelynél éppen a vezérjel van. A küldő gép csak az üzenetküldés után továbbítja a vezérjelet.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



39. Ábra

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 tönkretehet. Az alábbi ábra példa egy fa topológiájú hálózatra:



40. Ábra

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;

- nagykiterjedésű hálózatokat (WAN - Wide Area Network), melyek mérete az 50 km-t meghaladja és a vezetékes összeköttetést felvált(hat)ja a rádiós vagy mikrohullámú adatátvitel.

A **fizikai távolság** (processzorok közti távolság, illetve a processzorok elhelyezkedése) szerinti osztályozás:

6.1. táblázat - Táblázat 4

Távolság	Hely	Hálózat típusa
0.1 m	egy kártyán	adatfolyam gép
1 m	egy rendszerben	multiprocesszor
10 m - 1000 m	egy szobában, épületben, egyetemen	helyi hálózat
10 km	egy városban	városi hálózat
100 km - 10000 km	egy országban, kontinensen, bolygón	nagykiterjedésű hálózat

Az **ISO** (International Standards Organization) – modell logikailag rétegekre bontja a hálózatot az alábbi elvek szerint:

- Hasonló funkciókat azonos rétegbe kell gyűjteni.
- Lényegesen különböző funkciókat különböző rétegekbe kell csoportosítani.
- A réteghez tartozó funkciók és protokollok cseréjét engedélyezni kell anélkül, hogy más rétegeket befolyásolnának.
- A réteg teljes mértékben újratervezhető és protokolljai cserélhetők legyenek úgy, hogy az architektúra a hardver- vagy szoftver-technológia előnyeit kihasználhassa a felső szomszédos réteg számára szükséges szolgálatok maradéktalan biztosítása mellett.
- Új réteget kell létrehozni ott, ahol az adat kezelése különböző szintű absztrakciókat igényel.
- Minden réteg csak a közvetlen szomszédjával határos.
- Meg kell hagyni azokban a pontokban a határokat, amelyeket a gyakorlati tapasztalat régebbről indokol.
- Határvonalat kell létrehozni abban a pontban, ahol a szolgálatok leírása rövid és a határon keresztüli interakciók száma minimális.
- Határvonalat kell létrehozni ott, ahol a szükséges interfészre szabványosítási igény van kilátásban.
- Ne legyen olyan sok réteg, hogy a rendszerintegrációs leírások és a rétegek integrációja bonyolultabbá váljon, mint amennyire az szükséges.

A számítógépes hálózat, tekinthető úgy, mint egy bizonyos számú réteg. Minden egyes szint, amely egyben hardver, és szoftver segítségével megvalósított funkcionális réteg, egy funkció készletet biztosít.

Az adatok továbbítása a fizikai hálózaton **protokollok** szerint történik. A kommunikációs rendszer minden egyes szintjén egy jól definiált szabály halmazt kell használni ahhoz, hogy a kommunikáció sikeres legyen. Egy adott rétegen a szabályok összességét protokollnak nevezzük.

Az **OSI** (Open System Interconnection) modell logikailag hét rétegre bontja a hálózatot:

1. A **fizikai réteg** (physical layer) a bitek kommunikációs csatornára való kibocsátásáért felelős. Biztosítania kell, hogy az adó által küldött jeleket a vevő is azonosként értelmezze (ha 1-t adnak, akkor a vevő is 1-et vegyen).
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. Repeater (jelismétlő) vagy bridge (hídaló) tartozik ide és az Ethernet/IEEE 802.3

3. A **hálózati réteg** (network layer) a kommunikációs alhálózatok működését vezérli, feladata az útvonalválasztás a forrás és acélállomás között. Ez lehet statikus vagy dinamikus. Fontos a torlódás megakadályozása. E réteg feladata bizonyos számlálási és számlázási feladatok biztosítása. Ha az útvonalban eltérő hálózatok vannak, akkor fregmentálást, protokoll átalakítást is végez. Az utolsó olyan réteg, amely ismeri a hálózat topológiáját. **IP** (Internet Protocol) Ez a protokoll gondoskodik a csomagok átviteléről a hálózaton. Ez egy kapcsolat nélküli protokoll, azaz a kommunikációhoz nem szükséges előzetes kapcsolatfelvétel. Ebből adódik, hogy tartalmaznia kell a kézbesítéssel összefüggő információkat, a teljes és pontos címet. Adatátvitel szempontjából nem megbízható, a csomagokkal bármi megtörténhet: elveszhetnek, megsérülhetnek, sorrendjük összekeveredhet. Még nagyobb gond, hogy ha nem sikerült az adatok továbbítása, akkor nem értesíti a küldő alkalmazást.

4. A **szállítási réteg** (transport layer) alapvető feladata üzenetek fogadása a viszonyrétegtől, ezek széttördelése, elküldése (különböző csomagok különböző útvonalon), és annak biztosítása, hogy minden darab hibátlanul megérkezzék. Már nem ismeri a topológiát, csak a két végpontban van rá szükség. Feladata az összeköttetések felépítése, bontása, csomagok sorrendbe állítása. A **TCP** (Transmission Control Protocol - átvitelvezérlő protokoll), a megbízható protokoll, az IP-t használja az adatok továbbításához. A protokollok tervezői a TCP-be építik be azokat a megbízhatósági jellemzőket, amelyek az IP-bõl hiányoznak. Egy másik protokoll ami ide tartozik az UDP (User Datagram Protocol).

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 átviendő információ szintaktikájával és szemantikájával foglalkozik. Az egyetlen olyan réteg, amely megváltoztathatja az üzenet tartalmát. Tömörít, rejtjelez (adatvédelem és adatbiztonság miatt), kódcserét (pl.: ASCII - EBCDIC) végez el.

7. Az **alkalmazási réteg** (application layer) a felhasználóknak nyújt közvetlen szolgáltatásokat. Protokollokat tartalmaz. Tipikus alkalmazási rétegfeladatok pl. az állománytovábbítás, elektronikus levelezés, távoli munkabevitel. LAN-ok esetén ezt a munkaállomásokra és a szerverre telepített hálózati alkalmazások biztosítják.

1. 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 körbevesszük, 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 ismétlőket 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 kábellel. 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 lézerdióda, mely villamos áram hatására fényimpulzusokat bocsát ki magából. A fényérzékelő egy fotodióda, amely fény hatására villamos jeleket állít elő. Az optikai szál egyik végére LED-et vagy lézerdiódát, a másik végére fotodiódát 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.

2. Vezeték nélküli adatátviteli közegek

Bár a legtöbb kommunikációs rendszer rézhuzalt vagy optikai szálat 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 energiakoncentrálás 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.

Lokálisan nagyon elterjedt a WIFI. A Wi-Fi (WiFi, Wifi vagy wifi) az **IEEE** által kifejlesztett vezeték nélküli mikrohullámú kommunikációt (**WLAN**) megvalósító, széleskörűen elterjedt szabvány **IEEE 802.11** (semmilyen angol kifejezésnek nem rövidítése, csupán szójáték a **hifi** szó analógiájára).

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.

3. Adatátviteli vezérlő egységek

Repeater

A **repeater** (jelismétlő) helyi hálózat egy szegmenséről kapott jeleket újraidőzíti 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.

4. Az Internet

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 protokolloknak 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 fejlécet leválasztjuk, majd a csomagokból visszaállítjuk az eredeti információt.

RFC

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

IP Címzés

A TCP/IP hálózatokban a számítógépeket egységes címzési rend alapján azonosítjuk. Minden egyes gép egyedi hálózati címmel, az ún. IP-címmel rendelkezik. A címek 32 bitesek. A felírás 4 darab 8 bites decimális szám formájában történik. (Az internet dokumentációkban a byte helyett az oktett kifejezést használják a 8 bites számokra. Ez azért van így, mert a TCP/IP-t olyan számítógépek is használják, amelyek architektúrájába a byte nem 8 bites számot jelöl.) A 4 byte-os, IPv4 - cím byte-jait pontokkal választjuk el egymástól (pl. 192.168.50.130). Az egyes elemek értéke 0-255-ig terjedhet. Az IP-címekben a 0 és a 255 speciális jelentéssel bír. A 0 az olyan gépek számára van fenntartva, amelyek nem tudják a hálózati címüket. Az IP-címek nem kezdődhetnek se 0-val, se 127-tel, se 223-nál nagyobb számmal. Az ezeket a szabályokat megszegő címekre marslakókként hivatkoznak, mert elterjedt, hogy a Mars Központ Egyeteme a 225-ös hálózatot használja. A gép elsődleges azonosítója valójában az IP-cím. A számok nevekre való lefordítását az ún. domain name szerverek (DNS) végzik. Az IPv6-os 128 bites címzésstruktúrája szemben az IPv4 32 bites címeivel végtelen címzésterületet jelent, amelyek nem kötött struktúrájúak, "osztálymentesek". IPv4 - címek nem elegendőek a várhatóan egymilliárd új csomóponthoz (pl. mobilkészülékek). Az IPv6 meglehetősen új fogalmat honosít meg az összeköttetés mentes IP világban, a folyam fogalmát, ami egy irányba haladó, azonos feladójú és címzetű IP csomagok sorozatát jelenti. Minden IP csomagba kerül egy azonosító, ami azonosíthatja a folyamot, amihez a csomag tartozik. Ezzel nemcsak a route-olás egyszerűsödik, de lehetővé válik a folyamok számára utat kiépíteni, erőforrásokat lefoglalni, továbbá a QoS (Quality of Service).

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 van 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 anonymus 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 elterjedtté.

Az interneten folyamatosan hatalmas mennyiségű információ halmozódott fel, és a rendelkezésre álló állományok nevéről 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 webszerverekkel **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.

7. fejezet - 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.

1. 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 szoftverfejlesztőktől 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 számítógépgép 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 készítenek (szövegszerkesztők, táblázatkezelők, adatbáziskezelők, tudományos szubrutingyűjtemények, 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.

2. 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, licence megjelölés nélkül.

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.

8. fejezet - 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 (rendszerszoftver) 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 gondoskodjon 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 számítógépcsaládra 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 (kötegelt, 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 áthelyezhetőknek** 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).

1. Személyi számítógépek operációs rendszerei

A személyi számítógépek operációs rendszerei a nagyszámítógépekétől 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.

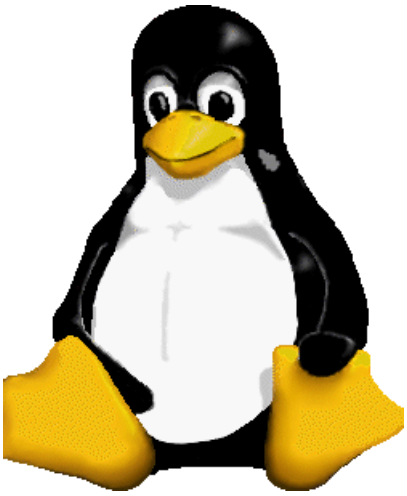
A Microsoft a Windows 3.1 grafikus felület, valamint a Windows NT sikeréből kiindulva azt a célt tűzték ki, hogy olyan átfogó operációs rendszert fejlesztenek ki, amely egyszerűbbé teszi a személyi számítógépek használatát, segíti a hálózatok elérését és biztosítja a kompatibilitást a korábbi operációs rendszerekkel. Így születtek meg a Windows különböző verziói (Windows 95, 98, 2000, XP, 7, 8, Vista).

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:



41. Ábra

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

9. fejezet - Algoritmusok

Az **algoritmus** szó **Abu Abdalláh Muhammad ibn Musa al-Khwarizmi** taskenti bölcs (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.

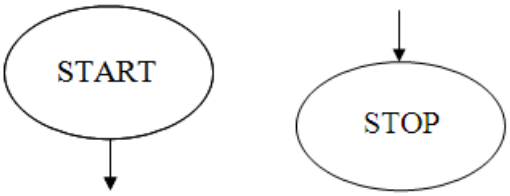
Minden meghatározásból leszűrhetők az algoritmus **tulajdonságai**:

- egy értékhez vagy ezek egy halmazához (**input**) hozzárendel egy értéket vagy ezek egy halmazát (**output**)
- **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).

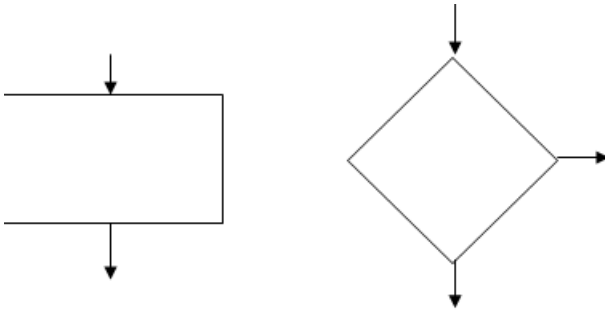
Egy algoritmus **elemi** (azonnal végrehajtható) és **összetett** (elemi tevékenységekből felépülő) műveletek sorozata. Például a rendező algoritmus, lásd később. Az algoritmus struktúráját **szekvenciák, szelekciók,** illetve **iterációk** adják, amelyek tetszőleges mélységben egymásba skatulyázhatók.

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 végrehajtódnak-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.

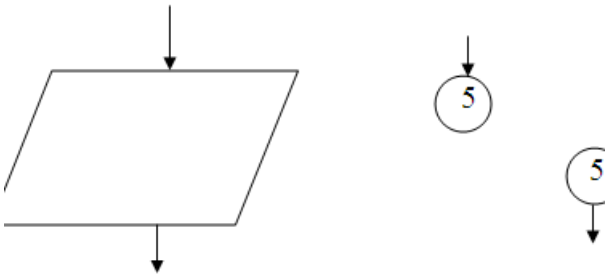
Az algoritmusok tervezésére, szemléltetésére, megadására, áttekinthető formában történő leírására sokféle eszköz létezik. A leggyakrabban használt a **folyamatábra**, amelynek alapelveit Neumann és Goldstine a programozás kapcsán dolgozta ki. (A folyamatábra szabványban meghatározott szimbólumok rendezett 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 folyamatábra 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 **pszeudokód** (leíró nyelv – „álnyelv”, „mondatszerű leírás”), melynek segítségével megfogalmazott algoritmus könnyen átírható általános célú programozási nyelvre. A beszélt nyelvhez hasonló, de annál tömörebb strukturált 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) kifejezések listája.

Szekvencia:

utasítás_1;

utasítás_2;

. . .

utasítás_n.

Szelekció:

if [ha] feltétel **then** [akkor]

utasítás[ok]

endif [elágazás vége]

if feltétel **then**

utasítás[ok]1

else [egyébként]

utasítás[ok]2

endif [elágazás vége].

Iteráció (hátul tesztelős, elől tesztelős, előírt lépésszámú):

repeat [ismétlés]

utasítás[ok]

until [amíg] feltétel.

while [amíg] feltétel **do** [végezd]

utasítás[ok]

endwhile [vége ismétlés].

for [minden] i:=1, n **do** [végezd]

utasítás[ok]

endfor [vége ismétlés].

Az Algoritmusok című referencia mű behúzást alkalmazva jelzi, hogy mely utasítások képezik a szelekció, illetve az iterációk utasításmagvait. Amint fentebb láthattuk, a jobb áttekinthetőség kedvéért (főleg, ha a kód átnyúlik a következő oldalra is), mi le is zárjuk a szelekciókat és az iterációkat. Hasonló módon járunk el az eljárások (procedure) és függvények (function) esetében is.

További jellegzetességei az általunk használt pszeudokódnak:

Értékadás: **változó**: = **kifejezés** (a változó felveszi a kifejezés értékét).

Aritmetikai operátorok: +, -, *, /, **DIV** (egész osztás), **MOD** (osztási maradék), **[ ]** (egész rész), √ (négyzetgyökvonás).

Összehasonlítási operátorok: =, ≠, ≤, <=, >, >=.

Logikai operátorok: **AND** (logikai ÉS), **OR** (logikai VAGY), **NOT** (logikai NEM).

Tömbök:

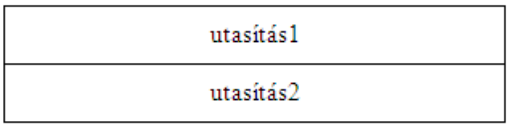
a[1..n] (n elemű tömb 1-től n-ig indexelve);

a[i] (az a töm i-edik eleme).

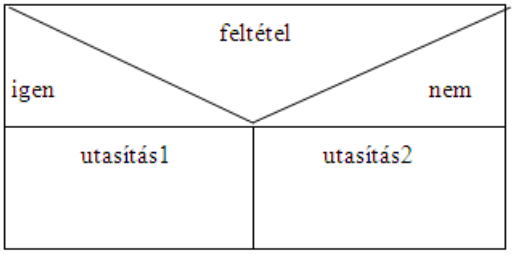
A harmadik eszköz a **struktogram** (NSD, **Nassi-Shneiderman diagram**), amelyet Ike Nassi és Ben Shneiderman vezetett be a hetvenes évek elején, a strukturált programozás algoritmusleíró eszköze. Az egyes szerkezeti elemeket téglalapba foglalható ábrákkal jelöljük. A szerkezetek egymásba ágyazhatók,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

Példák

Szekvencia:



Szelekció:



Iteráció (repeat-until):

1. Elemi algoritmusok

Programozás során gyakran oldunk meg olyan feladatokat, amelyekben előfordulnak több programban is alkalmazható utasításcsoportok. Az alábbiakban ilyen elemi algoritmusok pszeudokódját adjuk meg.

Példák

1. *Két változó tartalmának kicserélése*

   - *harmadik változó segítségével:*

     c := a;

     a := b;

     b := c.

   - *harmadik változó nélkül:*

     a := a + b;

     b := a – b;

     a := a – b.

2. *Adott egy osztályzat az 1...5 intervallumból. Határozzuk meg a megfelelő szöveges minősítést.*

```
read jegy;
if jegy = 1 then write "elégtelen” else
 if jegy = 2 then write "elégséges" else
 if jegy = 3 then write "közepes" else
   if jegy = 4 then write "jó" else
    write "jeles"
   endif
  endif
```

```
 endif
endif
```

3. *Adott egy n elemű számsorozat. Számítsuk ki a számsorozat elemeinek összegét.*

```
összeg := 0;
read n;
for i := 1 to n do
 read szám;
 összeg := összeg + szám;
endfor
write összeg.
```

4. *Adott egy természetes szám. Határozzuk meg a szám számjegyeinek szorzatát.*

```
szorzat := 1;
read szám;
while szám > 0 do
 szorzat := szorzat * (szám MOD 10);
 szám := szám DIV 10;
endwhile
write szorzat.
```

5. *Természetes számokat olvassunk be nulla végjelig. Számítsuk ki a számsorozat páros elemeinek átlagát.*

```
számláló := 0;
összeg := 0;
read szám;
while szám ≠ 0 do
 if szám MOD 2 = 0 then
  összeg := összeg + szám;
  számláló := számláló + 1;
 endif
 read szám;
endwhile
if számláló > 0 then
 write összeg/számláló;
else
 write "Nincs páros szám";
endif.
```

6. *Adott az n érték, illetve egy n elemű számsorozat. Határozzuk meg a számsorozat maximumának értékét.*

```
read n;
read szám;
max := szám;
for i := 2 to n do
 read szám;
 if szám > max then
  max := szám;
 endif
endfor
write max.
```

7. *Adott két nullától különböző természetes szám. Határozzuk meg a legnagyobb közös osztójukat.*

```
read szám1, szám2;
while szám1 ≠ szám2 do
 if szám1 > szám2 then
  szám1 := szám1 - szám2
 else
  szám2 := szám2 - szám1;
 endif
```

```
endwhile
write szám1.
```

8. *Adott két nullától különböző természetes szám. Határozzuk meg a legkisebb közös többszörösüket.*

```
read szám1, szám2;
összeg1 := szám1;
összeg2 := szám2;
while összeg1 ≠ összeg2 do
 if összeg1 < összeg2 then
  összeg1 := összeg1 + szám1;
 else
  összeg2 := összeg2 + szám2;
 endif
endwhile
write összeg1.
```

9. *Adott az n érték, illetve egy n elemű számsorozat. Számoljuk meg a számsorozat prím-értékeit.*

   Megjegyzés: A *0* és *1* számok nem prímek. Egy *1*-nél nagyobb szám akkor prím, ha nincs *2* és a gyöke közt egyetlen osztója sem.

```
számláló := 0;
read n;
for i:=1 to n do
 read szám;
 if szám = 0 then prím := 0;
 else if szám = 1 then prím := 0;
  else
  prím := 1; j := 2;
  while (j ≤ [√szám]) AND (prím = 1) do
   if szám MOD j = 0 then
    prím := 0;
   endif
   j := j + 1;
  endwhile
 endif
 endif
 if prím = 1 then számláló = számláló + 1;
 endif
endfor
write számláló.
```

Feladatok

1. *Készítse el a fenti algoritmusok folyamatábráját.*

2. *Készítsen algoritmust sorba kötött ellenállások értékeinek (r_i, i = 1 ,2, ...) file-ból való nulla végjelig történő beolvasására, majd az eredő ellenállás (R) kiíratására.*

   Megjegyzés: $R = \sum r_i$.

42. Ábra

3. *Készítsen algoritmust sorba kötött kondenzátorok kapacitásának (c_i, i = 1, 2, ...) file-ból való nulla végjelig történő beolvasására, majd az eredő kapacitás-érték (C) kiíratására.*

   Megjegyzés: $1/C = \sum 1/c_i$.

43. Ábra

4. *Legyártunk n darab, P paraméter értékű „identikus" tranzisztort. Készítsen algoritmust, amely file-ból beolvassa az n és P értékeket, valamint a tranzisztorok valós (mért) értékeit (p_i, i = 1, 2, ..., n), és meghatározza, hogy hány százalékuk selejt.*

   Megjegyzés: Az a tranzisztor selejt, amelynek a valós (mért) értéke a P elvárt értékéhez képest az 5%-os tűréshatáron kívül esik.

44. Ábra

5. *Mit ír ki az alábbi algoritmus, ha n értékeként 10-et olvasunk be?*7

```
read n;
k := 0;
i := 1;
while (k < n) do
 j := 1;
 while (k < n) AND (j ≤ i) do
  write j;
  j := j + 1;
  k:= k + 1;
 endwhile
 i := i + 1;
endwhile.
```

6. *Mekkora összeggé gyarapodik h hónap alatt egy T kezdőtőke, ha minden következő hónapban hozzáadódik az előző havi összeg k kamatlábnyi százaléka?*

   Megjegyzés: Mivel éves kamatlábat szokás megadni, ezért a havi kamatláb k/12.

7. *Adott egy kör a síkban középpontjának koordinátáival és sugarával. Számoljuk meg, hogy hány darab egész koordinátákkal jellemezhető koordinátapont esik a körön belülre.*

8. *Számítsuk ki és másodpercenkénti bontásban táblázatosan adjuk meg a v_0 kezdősebességű, a gyorsulással egyenletesen gyorsuló mozgást végző test által t idő alatt megtett utat.*

   Megjegyzés: A v_0, az a és a t bemenő adatok.

9. *Olvassuk be egy sakkfigura (vezér, király, bástya, futó, huszár, gyalog) aktuális koordinátáit a sakktáblán. Írassuk ki a sakkfigura által támadható mezők koordinátáit.*

2. 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 nehézkesebb, 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.

Megjegyzés: A *n* jelöli a számsorozat elemeinek számát, az *a[1..n]* tömb tárolja a számsorozat elemeit, *x* a keresett érték. Eredményként a keresett elem tömbbeli indexét írassuk ki, amennyiben az elem megtalálható.

```
i := 1;
while (i ≤ n) AND (a[i] ≠ x) do
     i := i + 1;
endwhile
if i ≤ n then write i
else write "A keresett szám nem található meg a számsorozatban."
endif.
```

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 kettesalapú logaritmusával arányos, ami nagy elemszámok esetén lényegesen kisebb lehet, mint a lineáris keresés esetén.

Megjegyzés: A *n* jelöli a számsorozat elemeinek számát, az *a[1..n]* tömb tárolja a számsorozat elemeit, *x* a keresett érték. Eredményként a keresett elem tömbbeli indexét írassuk ki, amennyiben az elem megtalálható.

```
eleje := 1;
vége := n;
repeat
   közepe := (eleje + vége) / 2;
```

```
   if x < a[közepe] then vége := közepe – 1
      else if x > a[közepe] then eleje := közepe + 1
           endif
   endif
until (x = a[közepe]) OR (eleje > vége);
if x = a[közepe] then write közepe
else write "A keresett szám nem található meg a számsorozatban."
endif.
```

Rendező algoritmusok

Beszúrás

Van egy *n* elemű rendezett tömbünk. Be akarunk szúrni egy *n+1*-edik elemet a rendezettség megtartásával. Egyik lehetőség, hogy végigszaladunk a halmazon, és betesszük a beszúrandó elemet az elé az elem elé, ami már nem előzi meg. Átlagosan *(n+1)/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 $[log_2 n]+1$, ha *n* nem 2 hatványa.

Megjegyzés: A *n* jelöli a számsorozat elemeinek számát, az *a[1..n]* tömb tárolja a számsorozat elemeit, *x* a beszúrandó érték. A kereséséhez a logaritmikus algoritmust használjuk.

```
eleje := 1;
vége := n;
repeat
   közepe := (eleje + vége) / 2;
   if x < a[közepe] then vége := közepe – 1
      else if x > a[közepe] then eleje := közepe + 1
           endif
   endif
until (x = a[közepe]) OR (eleje > vége);
if x ≤ a[közepe] then helye := közepe;
else helye := közepe + 1;
endif
for i := n downto helye do
   a[i+1] := a[i];
endfor
a[helye] := x.
```

Beszúrásos 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 beszúrjuk. Ez a rendezés túl sok mozgatást igényel.

Megjegyzés: A *n* jelöli a számsorozat hosszát, az *a[1..n]* tömb tárolja a számsorozat elemeit.

```
for i := 2 to n do
   x := a[i];
   eleje := 1;
   vége := i - 1;
   repeat
      közepe := (eleje + vége) / 2;
      if x < a[közepe] then vége := közepe – 1
         else if x > a[közepe] then eleje := közepe + 1
              endif
      endif
   until (x = a[közepe]) OR (eleje > vége);
   if x ≤ a[közepe] then helye := közepe;
   else helye := közepe + 1;
   endif
   for j := i-1, helye downto
```

```
      a[j+1] := a[j];
   endfor
   a[helye] := x;
endfor.
```

Shell rendezés

A beszúrásos módszer lépésenként finomított változata. A tömbnek csak minden *d*-edik elemét vizsgáljuk, azaz *d* lépésközzel végezzük el a beszúrásos 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.

Megjegyzés: A *n* jelöli a számsorozat hosszát, az *a[1..n]* tömb tárolja a számsorozat elemeit. A belső *while* ciklus lineáris beszúrást alkalmaz.

```
d := n / 2;
while d ≥ 1 do
   for j := d + 1 to n do
      i := j - d;
      while (i ≥ 1) AND (a[i+d] < a[i]) do
         c := a[i];
         a[i] := a[i+d];
         a[i+d] := c;
         i := i - d;
      endwhile
   endfor
   d := d / 2;
endwhile.
```

Rendezés közvetlen kiválasztással és cserével (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 mindenkori legkisebb elem keresése lineáris kereséssel történik.

Megjegyzés: A *n* jelöli a számsorozat hosszát, az *a[1..n]* tömb tárolja a számsorozat elemeit.

```
for i := 1 to n-1 do
   min_index := i;
   for j := i+1 to n do
      if a[j] < a[min_index] then
         min_index := j;
      endif
   endfor
   c := a[i];
   a[i] := a[min index];
   a[min_index] := c;
endfor
```

Buborékrendezés

Az egyik leggyakrabban használt rendező algoritmus. A rendezés a tömb elejéről és végéről is indulhat. Az *a[1..n]* tömböt legrosszabb esetben *(n-1)*-szer kell bejárni. A tömb elejéről induló esetben az első bejárásban összehasonlítjuk az *a[1]*-t az *a[2]*-vel, majd az *a[2]*-t az *a[3]*-mal, …, végül az *a[n-1]*-et az *a[n]*-nel. Általánosan az *a[j]*-t hasinlítjuk az *a[j+1]*-gyel, ahol *j = 1, 2 ... , n-1*. Valahányszor úgy találjuk, hogy *a[j] > a[j+1]*, felcseréljük őket. A tömb elejéről induló rendezésnél a legnagyobb elem biztosan a helyére fog kerülni. A többi elem azonban még nem feltétlenül került a végleges helyére. (A tömb végéről induló rendezésnél mindig a legkisebb elem kerül biztosan az adott rendezési ciklusban a helyére.) A következő bejárásokban hasonlóképpen járunk el, és további elemek kerülnek a helyükre, ahhoz hasonlóan, ahogy a levegőbuborékok szoktak feljönni a víz felszínére. 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. Ha megjegyezzük az utolsó csere helyét (*ucs*), akkor a következő bejárás csak addig megy.

```
u := n;
repeat
   ucs := 0;
   for j := 1 to u-1 do
      if a[j] > a[j+1] then
         c := a[j];
         a[j] := a[j+1];
         a[j+1] := c;
         ucs = j;
      endif
   endfor
   u := ucs;
until ucs=0.
```

Összefésülés

Van két rendezett tömbünk, *a[1..n]* és *b[1..m]*. Ezeket kell összefésülni egy *c[1..n+m]* tömbbe. Az *i* változó az *a* tömb, a *j* pedig a *b* tömb elemein fog lépegetni. Az *a[i]* és *b[j]* elemek közül mindig a kisebbiket tesszük a *c* tömbbe, majd lepünk egyet abban a tömbben, amelyből „kikerült" az az elem, amelyet a *c* tömbbe helyeztünk.

```
i := 1;
j := 1;
k := 1;
while (i ≤ n) AND (j ≤ m) do
   if a[i] ≤ b[j] then
      c[k] := a[i];
      i := i + 1;
      k := k + 1;
   else
      c[k] := b[j];
      j := j + 1;
      k := k + 1;
   endif
endwhile
while i ≤ n do
   c[k] := a[i];
   i := i + 1;
   k := k + 1;
endwhile
while j ≤ m do
   c[k] := b[j];
   j := j + 1;
   k := k + 1;
endwhile.
```

Összefésüléses rendezés

Az eddig ismertetett rendező algoritmusoktól eltérő, rekurzív elven működik. A *c[1..n]* tömb rendezése rekurzívan visszavezethető a *c[1..n/2]* és a *c[n/2+1..n]* tömbszakaszok külön-külön történő rendezésére, és az ezt követő összefésülésükre. Az általános rendez eljárást egy *c[i..j]* tömbszakasz rendezésére kell megírnunk. A feladat akkor triviális, ha a rendezendő tömbszakasz egyelemű *(i = j)*.

Megjegyzés: A *rendez* eljárást kezdetben a teljes tömbre hívjuk meg, *rendez (c[1..n])*;. Az *összefésül* eljárás az *a* és *b* segédtömböket használja.

```
procedure rendez (c[i..j]);
   if i < j then
      k := (i + j) / 2;
      rendez (c[i..k]);
      rendez (c[k+1..j]);
      összefésül (c[i..j]);
   endif
```

```
endprocedure



procedure összefésül (c[i..j])
   k := (i + j) / 2;
   for p := i to k do
      a[p] := c[p];
   endfor
   for p := k+1 to j do
      b[p] := c[p];
   endfor
   a[k+1] := ∞; b[j+1] := ∞; {strázsák}
   ii := i; jj := k+1;
   for p := i to j do
      if a[ii] < b[jj] then
         c[p] := a[ii];
         ii := ii + 1;
      else
         c[p] := b[jj];
         jj = jj + 1;
      endif
   endfor
endprocedure
```

Szétválogatás (előrendezés)

Az *a[1..n]* tömb elemeit válogassuk szét „az első eleme szerint” (az első elemnél kisebb elemek előtte, a nagyobbak mögötte legyenek). Egy *i*, illetve egy *j* változót a tömb elejére, illetve végére állítunk. Kezdetben az elem, amely szerint a szétválogatás történik (továbbiakban kulcselem), nyilván az *i*-edik pozícióban van. Miközben *i* áll, jövünk előre *j*-vel (magunk mögött hagyva a kulcselemnél nagyobb elemeket), míg nem találunk egy *a[j]*-t, ami kisebb, mint az *a[i]*. Felcserélve egymás közt az *a[i]*-t és az *a[j]*-t, a kulcselem hátra kerül a *j*-edik pozícióba, a nála kisebb elem pedig előre az *i*-edikbe. Ekkor „váltás” történik. Miközben *j* áll, *i*-vel megyünk hátrafele (magunk mögött hagyva a kulcselemnél kisebb elemeket), míg nem találunk egy *a[i]*–t, ami nagyobb, mint az *a[j]*. Ismét felcserélve egymás közt az *a[i]*-t és az *a[j]*-t, a kulcselem visszakerül az *i*-edik pozícióba, a nála nagyobb elem pedig hátra a *j*-edikbe. Most újra *j*-vel jövünk előre (1. eset), majd ismét *i*-vel megyünk hátra (2. eset) mindaddig, amíg *i* és *j* találkozik. A kulcselem mindig az álló index irányában van, az előre lepegető *j* a kulcselemnél nagyobb, a hátrafele mozgó *i* pedig a kulcselemnél kisebb elemeket hagyja maga mögött. A kulcselem a helyét, amelyet a rendezett számsorozatban foglalna el, ott találja meg, ahol találkoznak az *i* es *j* indexek.

```
i := 1; j := n;
eset := 1;
   while i < j do
      if a[i] ≤ a[j] then
         if eset = 1 then j := j - 1
         else i := i + 1
         endif
      else
         c := a[i]; a[i] := a[j]; a[j] := c;
         if eset = 1 then eset := 2
         else eset := 1
      endif
   endif
endwhile.
```

Gyorsrendezés (quick-sort)

Ez az algoritmus is rekurzív elven működik. Szétválogatjuk az aktuális tömbszakasz (*a[i..j]*) elemeit a szakasz első eleme (az a*[i]* kulcselem) szerint, majd folytatjuk a rendezést a szakasz kulcselem előtti, illetve mögötti elemei rendezésével.

Megjegyzés: A *rendez* eljárást kezdetben a teljes tömbre hívjuk meg, a *szétválogat* függvény adja vissza a helyére került kulcselem pozícióját.

```
procedure rendez (a[i..j])
   if i < j then
      k := szétválogat (a[i..j]);
      rendez (a[i..k-1]);
      rendez (a[k+1..j]);
   endif
endprocedure

function szétválogat (a[i..j])
   ii := i;
   jj := j;
   üzemmód := 1;
   while ii < jj do
      if a[ii] ≤ a[jj] then
         if üzemmód = 1 then jj := jj - 1
         else ii := ii + 1
         endif
      else
         veder=a[ii]; a[ii]=a[jj]; a[jj]=veder;
         if üzemmód = 1 then üzemmód := 2
         else üzemmód := 1
         endif
      endif
   endwhile
   return ii;
endfunction.
```

3. 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, vezérlőszerkezetek, 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 (gépközeli) 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.

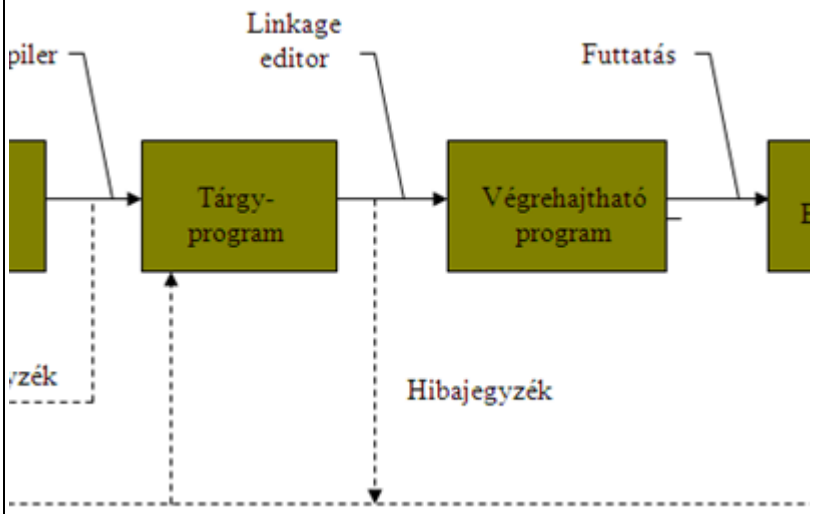
Azonban még ez a programozási mód is túl nehézkes, és nagyobb feladatok megoldására alkalmatlannak bizonyult. Ezért fejlesztették ki a **magas szintű programozási nyelveket**, melyek segítségével a megoldandó feladatot könnyebben és tömörebben lehet megfogalmazni. A programnyelvek **harmadik generációját** 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 átvihetőek 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:



45. Ábra

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 (rutingyűjteményből), 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 programrészleteket. 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 tárgyhoz. 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.

10. fejezet - Programozás C nyelven

Egy számítógépes program alapvetően egy utasítássorozat (valamely programozási nyelv elemeiből összeállítva), amely a számítógépet a feladat megoldási algoritmusában előírt műveletek végrehajtására utasítja. Követve az alábbi példákban az algoritmusok lépései, és az algoritmusok szerint megírt C nyelvű programok utasításai közötti megfeleléseket, ízelítőt kaphatunk a C nyelv utasításkészletéből is.

Példák

1. Az „Elemi algoritmusok” című alfejezet 1. példája:

Megjegyzés: Egy C program fő része a **main** függvény. A függvények magját képező utasításokat kapcsos zárójelek közé tesszük. A **return** 0 utasítás azt jelzi, hogy hiba nélkül futott le a program. A változókat definiálni kell, megadva a típusukat (deklaráció). Az **int** egész típust jelöl. Az azonosítók csak az angol ABC kis- és nagybetűit, aláhúzás jelet és számjegyeket tartalmazhatnak, de nem kezdődhetnek számjeggyel. Az *stdio.h* header állomány a **scanf** és **printf** függvényeket tartalmazza, amelyek adatoknak a standard input eszközről való beolvasására, illetve a standard output eszközre való kiíratására szolgálnak. A % jellel bevezetett *i* formázó karakterek azt jelzik, hogy a megfelelő változók **int** típusúak. Egy változónév előtt az & jel a változó memória címére való hivatkozást teszi lehetővé.

```
read a, b;
c := a;
a := b;
b := c;
write a, b;
```

```
#include <stdio.h>
main()
{
   int a, b, c;
   scanf("%i%i", &a, &b);
   c = a;
   a = b;
   b = c;
   printf("%i, %i", a, b);
   return 0;
}
```

2. Az „Elemi algoritmusok” című alfejezet 2. példája:

Megjegyzés: A C nyelv összehasonlító operátorai a ≤, >, ≤=, >=, == és !=. Ha egy többirányú elágazás ugyanazon egész kifejezés különböző értékei szerint történik, akkor használhatjuk a **switch** utasítást. A **break** lehetővé teszi a kiugrást **switch**, illetve ciklusutasításokból.

```
read jegy;
if jegy=1 then write "elégtelen"
else if jegy=2 then write "elégséges"
 else if jegy=3 then write "közepes"
  else if jegy=4 then write "jó"
   else write "jeles"
   endif
  endif
 endif
endif
```

```
#include <stdio.h>
main()
{
int jegy;
scanf ("%i", &jegy);
if (jegy==1) printf ("elégtelen");
else  if (jegy==2) printf ("elégséges");
 else if (jegy==3) printf ("közepes");
```

```
  else if (jegy==4) printf ("jó");
   else printf ("jeles");
return 0;
}

read jegy;
if jegy=1 then write "elégtelen"
else if jegy=2 then write "elégséges"
 else if jegy=3 then write "közepes"
  else if jegy=4 then write "jó"
   else write "jeles"
   endif
  endif
 endif
endif

#include <stdio.h>
main()
{
int jegy;
scanf ("%i", &jegy);
switch (jegy)
{
 case 1: printf ("elégtelen"); break;
 case 2: printf ("elégséges"); break;
 case 3: printf ("közepes"); break;
 case 4: printf ("jó"); break;
 case 5: printf ("jeles"); break;
 default printf ("Helytelen jegyérték.");
}
return 0;
}
```

3. Adott az n érték, illetve egy n elemű számsorozat. Határozzuk meg a számsorozat maximumának értékét.

Megjegyzés: A *++i* utasítás az *i* változó értékének *1*-el való növelését jelenti.

```
read n;
read szám;
max := szám;
for i := 2 to n do

 read szám;
 if szám > max then
  max := szám;
 endif
endfor
 write max;

#include <stdio.h>
main()
{
 int n, szam, i;
scanf ("%i", &n);
scanf ("%i", &szam);
max = szam;
for (i = 2; i <= n; ++i)
{
 scanf ("%i", &szam);
 if (szám > max)
  max = szam;

}
printf ("A legnagyobb érték: %i", max);
return 0;
}
```

4. Az „Elemi algoritmusok” című alfejezet 7. példája:

Megjegyzés: Ha egy változó értékéhez hozzá kell adni, vagy ki kell vonni belőle egy értéket, akkor használhatjuk a +=, illetve a –= összevont operátorokat.

```
read szám1, szám2;
while szám1 ≠ szám2 do
 if szám1 > szám2 then
  szám1 := szám1 - szám2
 else
  szám2 := szám2 - szám1;
 endif
endwhile
write szám1;

#include <stdio.h>
main()
{
int szam1, szam2;
scanf ("%i%i", &szam1, &szam2);
while (szam1 != szam2)
 if (szam1 > szam2)
  szam1 -= szam2;
 else
  szam2 -= szam1;
printf ("%i", szam1);
return 0;
}
```

5. Adott egy természetes szám. Ellenőrizzük, hogy prím-e.

Megjegyzés: A C nyelv logikai operátorai az && (ÉS/AND), a || (VAGY/OR) és a ! (NEM/NOT). Minden nem nulla érték a logikai IGAZ, a nulla pedig a logikai HAMIS. A *math.h* header állomány tartalmazza a matematikai függvényeket, így az **sqrt** négyzetgyök függvényt is.

```
számláló := 0;
read szám;
if szám = 0 then prím := 0;
else if szám = 1 then prím := 0;
 else
 prím := 1; j := 2;
 while (j ≤ [√szám]) AND (prím = 1) do
  if szám MOD j = 0 then
   prím := 0;
  endif
 j := j + 1;
 endwhile
 endif
endif
if prím = 1 then write "A szám prím!"
 else write "A szám prím!"
endif

#include <stdio.h>
#include <math.h>
main()
{
 int szamlalo=0, szam, prim, j;
 scanf ("%i", &szam);
 if (szam == 0) prim = 0;
 else if (szam == 1) prim = 0;
  else
  {
  prim = 1; j = 2;
  while (j ≤ sqrt(szám) && prim)
  {
   if (!(szam % j))
```

```
   prim = 0;
   ++j;
  }
 }
 if (prim) printf ("A szám prím!");
 else printf ("A szám prím!");
 return 0;
}
```

6. Olvassunk be a kereses_be.txt input állományból egy n természetes számot (n≤100), majd egy n elemű számsorozatot egy egydimenziós tömbbe, és keressünk meg a számsorozatban egy x értéket, amelyet ugyancsak az állományból olvasunk be. A keresés eredményét a kereses_ki.txt állományba írjuk ki. (Lineáris keresés.)

Megjegyzés: A állománymutatókat **FILE*** típusúnak kell deklarálni. Mielőtt egy állományból/állományba olvasnánk/írnánk, meg kell nyitni olvasásra (*r*)/írásra (*w*). A C nyelvben a tömbök indexe 0-val kezdődik. A **break** utasítás lehetővé teszi a ciklusból való kiugrást.

```
read n;
for i := 1 to n do
 read a[i];
endfor
read x;
i := 1;
while (i ≤ n) AND (a[i] ≠ x) do
 i := i + 1;
endwhile

if i ≤ n the write i
else write "Nincs"
 endif

#include <stdio.h>
main()
{
 int n, a[100], i, x;
 FILE *be, *ki;
 be = fopen ("kereses_be.txt", "r");
 ki = fopen ("kereses_ki.txt", "w");
 fscanf (be, "%i", &n);
 for (i = 0; i < n; ++i)
  fscanf (be, "%i", &a[i]);
 fscanf (be, "%i", &x);
 for (i = 0; i < n; ++i)
  if (a[i] == x) {
   fprintf (ki, "%i", i + 1); break;
   }
 if (i == n) fprintf (ki, "Nincs");
 fclose (be); fclose (ki);
 return 0;
}
```

7. Összefésüléses rendezés.

Megjegyzés: A program végrehajtása a **main** függvénnyel kezdődik. A **main** függvény meghívja a **rendez** eljárást. A **rendez** eljárás meghívja kétszer önmagát rekurzívan, majd az **összefésül** eljárást. Az eljárások (procedure) *C* nyelven **void** típusú függvények, amelyek elvégeznek egy részfeladatot. A valódi függvények (function) kiszámítanak egy eredményt, amit a hívó függvénynek adnak vissza. A hívó függvény paraméterlistán keresztül adhat át adatokat a meghívott függvénynek. Az *INTMAX* konstansa *limits.h* header állományban van.

```
procedure összefésül (x[i..j])

 k := (i + j) / 2;
 for p := i to k do
  a[p] := x[p];
```

```
 endfor
 for p := k+1 to j do
  b[p] := x[p];
 endfor
 a[k+1] := ∞; b[j+1] := ∞;
 ii := i; jj := k+1;
 for p := i to j do
  if a[ii] < b[jj] then
   x[p] := a[ii];
   ii := ii + 1;
  else
   x[p] := b[jj];
   jj = jj + 1;
  endif
 endfor
endprocedure

procedure rendez (x[i..j])
 if i < j then
  k := (i + j) / 2;
  rendez (x[i..k]);
  rendez (x[k+1..j]);
  összefésül (x[i..j]);
 endif
endprocedure

#include <stdio.h>
#include <limits.h>
void összefésül (int i, int j, int x[]){
 int k, a[100], b[100], p, ii, jj;
 k = (i + j) / 2;
 for (p = i; p <= k; ++p)
  a[p] = x[p];
 for (p = k +1; p <= j; ++p)
  b[p] = x[p];
 a[k+1] = b[j+1] = INTMAX;
 ii = i; jj = k+1;
 for (p = i; p <= j; ++p)
  if (a[ii] < b[jj])
  {
   x[p] = a[ii];
   ii = ii + 1;
  }
  else
  {
   x[p] = b[jj];
   jj = jj + 1;
  }
}

void rendez (int i, int j, int x[]) {
 int k;
 if (i < j) {
  k = (i + j) / 2;
  rendez (i, k, x);
  rendez (k + 1, j, x);
  összefésül (i, j, x);
  }
 }

main()
{
 int n, a[100], i;
 FILE *be, *ki;
 be = fopen ("rendezes_be.txt", "r");
 ki = fopen ("rendezes_ki.txt", "w");
 fscanf (be, "%i", &n);
 for (i = 0; i < n; ++i)
  fscanf (be, "%i", &a[i]);
 rendez (0, n - 1, a);
 for (i = 0; i < n; ++i)
  fprint (ki, "%i ", a[i]);
```

```
 fclose (be); fclose (ki);
 return 0;
}
```

Feladatok

1. Írjunk C programot az „Elemi algoritmusok" és „Nevezetes algoritmusok" alfejezetekben szereplő további megoldott, illetve kitűzött feladatokra.

11. fejezet - 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.

1. 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

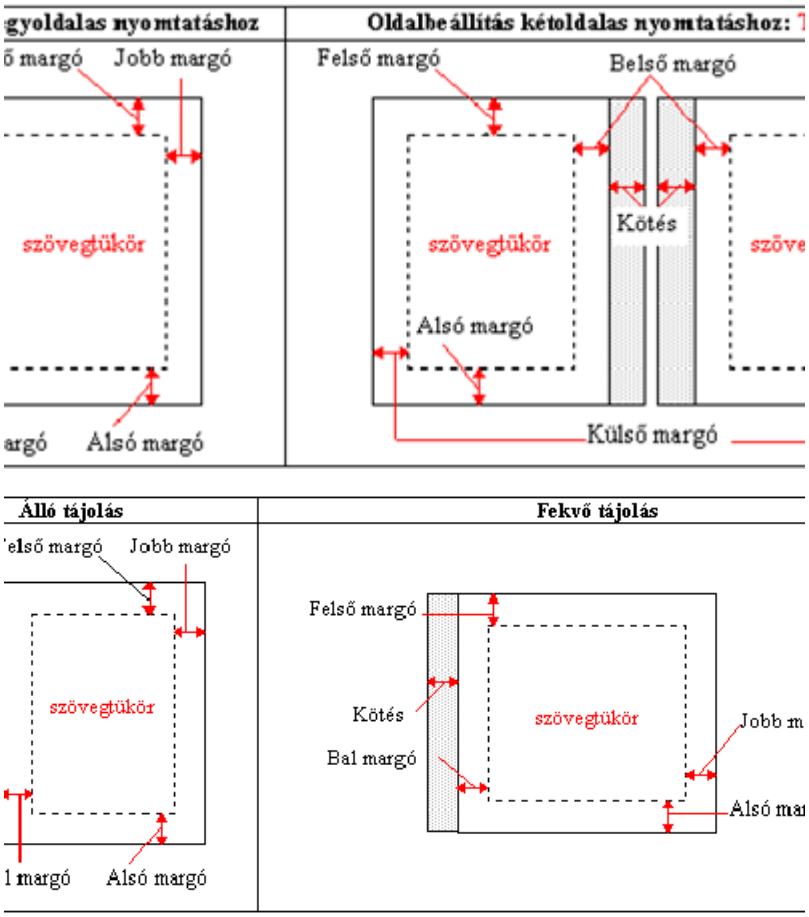
1. megnyitás (új, régi);
2. a szöveg begépelése;
3. formázás;

4. mentés (mentés másként);

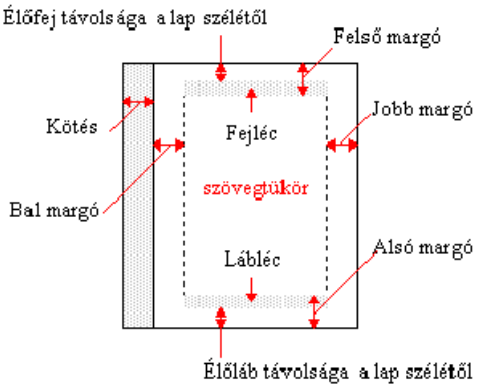
5. nyomtatás.

Tipográfiai ismeretek

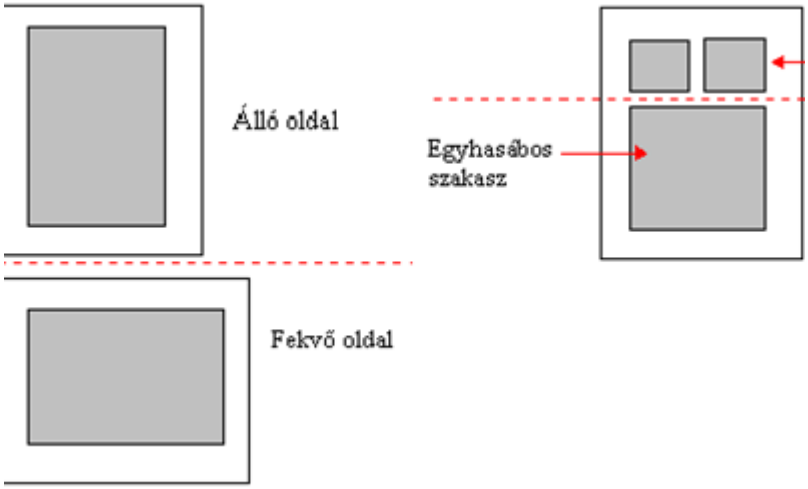
- a **papír beállításai** (oldalbeállítások - papírméret, margók, befűzés, tájolás);



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



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



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

Jellemző formázási lehetőségek

jellegét (alapvető alakját) határozza meg. A betűtípusokat karakter tábláknak is neve
e betűtípust (font készletet) lehet kapni lemezen (floppyn vagy CD-n).
l Europa=Közép-Európa) jelölésű betűtípusok tartalmaznak valódi ékezetes betűket.
om betűtípus általában minden Windows-os gépen megtalálható:

an betűtípus:	Courier betűtípus	Arial betűtípus
LMNOPRSTUVWX nnoprstuvwxyz	ABCDEFGHIJKLMNOPRSTUVWXYZ abcdefghijklmnoprstuvwxyz 1234567890	ABCDEFGHIJKLMN YZ abcdefghijklmno 1234567890
újság egyszerűen	Hasonlít az írógép betűihez.	Egyszerű egyenes bet
ségű karakterek	• Egyforma széles karakterek	• Eltérő szélességű

jellemzők: Ezek a jellemzők minden betűtípuson beállíthatók.

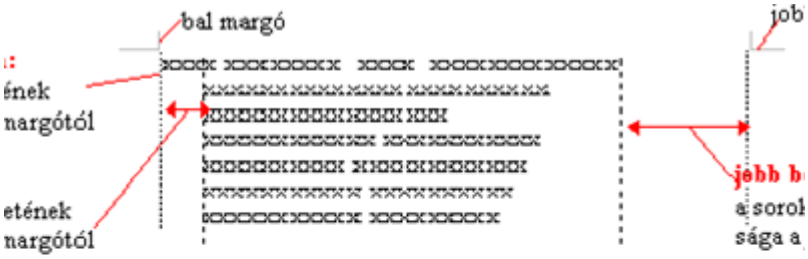
	Minta	Megjegyzés
	Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg	nagyobb betűtávolság kisebb betűtávolság
nalú lapvonalú	Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg	
t ketehangyák öröshangyák tér	Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg	

jellemzők: Ezek a jellemzők minden betűtípuson beállíthatók.

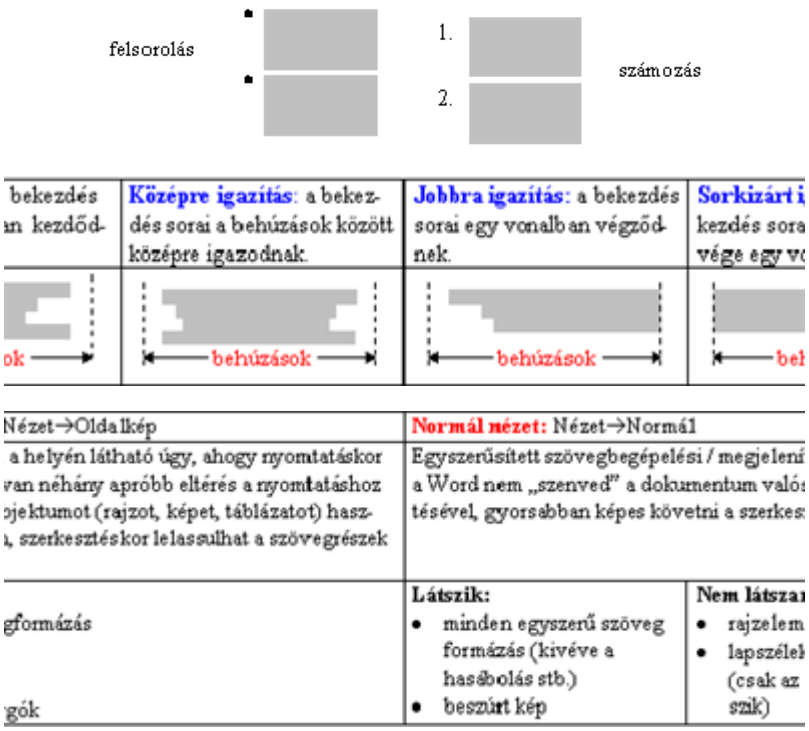
	Minta	Megjegyzés
	10, 12, 14, 16, 18...	Általában minden bet méretezhető. A betűm mértékegysége a pont
	Mintaszöveg *Mintaszöveg* Mintaszöveg	A betű stílusok szöve kiemelésére használat jellemzők .
nal	Mintaszöveg Mintaszöveg Mintaszöveg Mintaszöveg Mintaszöveg Mintaszöveg Mintaszöveg Mintaszöveg Mintaszöveg	további szövegrész kie gek ←a szóközöket nem hú
	Normálszöveg → Minta szöveg	
k: izott ;	Normálszöveg → ~~Mintaszöveg~~ Normálszöveg → **~~Mintaszöveg~~** Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → Normálszöveg → MINTASZÖVEG Normálszöveg → Mintaszöveg Normálszöveg → **Mintaszöveg** Normálszöveg → Mintaszöveg Normálszöveg → Mintaszöveg Normálszöveg → MINTASZÖVEG	felfelé tolt alapvonalú lefelé tolt alapvonalú (nem látszik (de ott van a kisbetűket kisebb na níti meg

formai jelek:

j sorban kezdődik.	A bekezdések első sora vízszintesen el van tolva a többi sorhoz képest.	A bekezdéseket soksz választják el.
1. bekezdés 2. bekezdés	1. bekezdés 2. bekezdés	



dés formák a behúzások szabályozásával:



Feladatok

1. Körlevélkészítés:

a) *Készítse el az alábbi táblázatot Excel-ben!*

11.1. táblázat - Táblázat 5

Név	Lakóhely	Cég	Telephely	Beosztás	Fizetés
Dombi Ákos	Téglás	Beléndek Rt.	Debrecen	osztályvezető	180000 Ft
Kovács Róbert	Budapest	Mikroklíma Bt.	Győr	mérnök	153000 Ft
Benke Zsolt	Hatvan	Parma Kft.	Budapest	osztályvezető	186000 Ft
Herczeg Richárd	Debrecen	Beléndek Rt.	Debrecen	igazgató	240000 Ft
Nagy Lajos	Budapest	Mikroklíma Bt.	Budapest	igazgató	195000 Ft
Koczka Árpás	Siófok	Parma Kft.	Győr	igazgató	205000 Ft
Leveles Ede	Debrecen	Beléndek Rt.	Debrecen	mérnök	84000 Ft
Zsiga Sándor	Polgár	Mikroklíma Bt.	Debrecen	osztályvezető	173000 Ft
Kósa János	Miskolc	Parma Kft.	Miskolc	szakmunkás	950000 Ft
Vadász Márton	Tiszaújváros	Parma Kft.	Miskolc	igazgató	2250000 Ft

b.) Rendezze az adatokat cég szerint növekvő sorrendbe, ezen belül telephely szerint növekvő sorrendbe, ezen belül pedig név szerint növekvő sorrendbe!

c.) A 100000,-Ft-nál kevesebb fizetéseket dinamikusan emelje meg 10 %-kal!

d.) Gyűjtse ki azoknak a nevét, akiknek a lakhelye a cége telephelyén van!

e.) Készítsen diagramot, amely összehasonlítja a cégenkénti átlagfizetéseket!

A táblázat adatait adatforrásként felhasználva készítsen körlevelet A4-es méretben a következő minta alapján! A körlevelet a cégek igazgatóinak és osztályvezetőinek küldje el!

Az Egészséges Életért Alapítvány

<<Név>>

<<Cég>><<Telephely>>

Tisztelt *<<Beosztás>>* Ur!

A ma Magyarországon meglevő és egyre szükülő anyagi források ellenére a szociális gondoskodás és az oktatás egyes ágainak segitésére jött létre az alapítvány. Fő célkitüzésünk az „EGÉSZSÉGES ÉLET" mint szemlélet és mint életforma propagálása. Emblémánk:

Ha jobban meg szeretne ismerkedni alapítványunk müködésével, keresse fel irodánk[1] valamelyikében munkatársunkat:

Az Egészséges Életért Alapítvány Központjai			
Iroda	**Cím**	**Telefonszám**	**Irodavezető**
Debrecen	Derék u. 57.	435-829	Nagy Zsolt
Szeged	Béke tér 5.	210-578	Kiss Anna

Ha döntenie kell, az egészséges életet válassza!

Debrecen, 2010. június 21.

Tisztelettel:

Sós Ottó sk.

elnök

Az Egészséges Életért Alapítvány

[1] Irodáink munkanapokon 8^{00}-tól 16^{00}-ig vannak nyitva.

2. Reprodukció:

a.) Reprodukálja az alábbi meghívót szövegét a „Meghívó"-tól a „Tóth Balázs."-ig!

Debreceni Egyetem
Informatikai Kar
4032 Debrecen, Egyetem tér 1.

Meghívó

A Debreceni Egyetem Informatikai Kara 2010. március 18-án tartja következő *Tudományos Szemináriumát.*

A szeminárium vendége

Nagy Pista István

professzor, aki

Hogyan vegyük rá az egyetemi hallgatókat a tanulásra?

címmel tart előadást az M/105-106 teremben 17^{00}-tól. A szeminárium 16^{30}-kor teázással, kávézással kezdődik.

A szeminárium házigazdája *Kis Pista István.*

Kérjük, hogy jelenlétével tisztelje meg Ön is a Szemináriumot.

— o —

A következő három Szeminárium időpontja és előadója:

Március 24.: Alma Ata;
Március 31.: Kis Bence;
Április 7.: Tóth Balázs.

b.) Másolja át a kész szöveget még három új oldalra úgy, hogy egy négyoldalas dokumentumot kapjon!

c.) A képet és a mellette levő címet helyezze el az oldalakon az alábbiak szerint:

1. oldalon: két hasábban;

2. oldalon: egysoros, kétoszlopos táblázatban;

3. oldalon: a szöveg szövegdobozban kerüljön a kép mellé;

4. oldalon: a szöveg képkörülírással kerüljön a kép mellé.

12. fejezet - Feladatgyüjtemény

1. Számrendszerek

1.1. Átváltások különböző számrendszerekben

1. Számoljuk át tízes számrendszerbe az alábbi számokat:
   a. 1011100.101_2=
   b. 1221.12_3=
   c. 31232.12_4=
   d. 42341.23_5=
   e. 152.43_6=
   f. 4645.24_7=
   g. 173.104_8=
   h. 841.47_9=
   i. $14A2E.24_{16}$=
2. Írjuk át kettes és nyolcas számrendszerbe a tizenhatos számrendszerbeli, illetve nyolcas és tizenhatos számrendszerbe a kettes számrendszerbeli számokat:
   a. $3BCF_{16}$=
   b. $BF29_{16}$=
   c. $48C5_{16}$=
   d. $63AE_{16}$=
   e. $1110\ 1001\ 1100\ 0011_2$=
   f. $1011\ 0111\ 0101\ 0100_2$=
   g. $1000\ 1101\ 1111\ 0011\ 1101_2$=
   h. $1010\ 1011\ 0011\ 1110\ 0001\ 0101_2$=
3. Számoljuk át kettes számrendszerbe az alábbi tízes számrendszerbeli számokat:
   a. 1862_{10}=
   b. 93281_{10}=
   c. $39871{,}64_{10}$=
   d. $49322{,}1813_{10}$=
4. Számoljuk át hetes számrendszerbe az alábbi tízes számrendszerbeli számokat:
   a. 1951_{10}=
   b. 82718_{10}=

c. $417{,}18_{10}$=

d. $13791{,}27_{10}$=

5. Számoljuk át kilences számrendszerbe az alábbi tízes számrendszerbeli számokat:

a. 2334_{10}=

b. 83191_{10}=

c. $218{,}92_{10}$=

d. $5245{,}67_{10}$=

6. Írjuk át tízes számrendszerbe az alábbi számokat:

a. $0{,}33'123'123\ldots_{4}$=

b. $0{,}42'62'62\ldots_{7}$=

c. $0{,}25'175'175\ldots_{8}$=

d. $0{,}57'83'83\ldots_{9}$=

e. $7B.73'5'5\ldots_{16}$=

1.2. Aritmetikai műveletek

1. Végezzük el az alábbi összeadásokat a megadott számrendszerbe!

a. $100011011_{2} + 101001101_{2}$=

b. $11010011011_{2} + 10101101101_{2}$=

c. $12112_{3} + 212221_{3}$=

d. $32121_{4} + 123321_{4}$=

e. $2342120_{5} + 20143_{5}$=

f. $52341_{6} + 334215_{6}$=

g. $234653_{7} + 3624025_{7}$=

h. $1347653_{8} + 7624025_{8}$=

i. $3467251_{9} + 8276573_{9}$=

j. $A3075AD3_{16} + 65ABCD74_{16}$=

2. Végezzük el az alábbi kivonásokat a megadott számrendszerbe!

a. $10101011011_{2} - 101001101_{2}$=

b. $1010101011011_{2} - 110101101101_{2}$=

c. $1021012_{3} - 2120221_{3}$=

d. $3021201_{4} - 1023321_{4}$=

e. $203402120_{5} - 201403_{5}$=

f. $5203401_{6} - 30340215_{6}$=

g. $20340653_7 - 30624025_7 =$

h. $103470653_8 - 70624025_8 =$

i. $3406702501_9 - 8276573_9 =$

j. $A03075AD3_{16} - 650ABCD74_{16} =$

1.3. Számábrázolás

1. Melyik IEEE 754 számot ábrázoltuk? A végeredményt decimális számrendszerben, egészrész és törtrész alakban adja meg!

   a. 1 10001110 01010100111111111000000

   b. 0 11000111 00101101001110000000000

   c. 1 10001011 00010101001111000100000

   d. 0 10010110 01001010100101100000000

2. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint! A végeredményt hexadecimális számrendszerben adja meg!

   a. 1248.24_{10}

   b. 732.32_{10}

   c. 3628.78_{10}

   d. 3218.58_{10}

3. Adja meg a következő tízes számrendszerbeli számok 1-es komplemens, 2-es komplemens, 127-többletes és 128-többletes ábrázolásait 8 biten (a végeredményt hexadecimális számrendszerben adja meg)!

   a. 45_{10}

   b. 83_{10}

   c. -37_{10}

   d. -94_{10}

4. Ábrázoljuk fixpontosan 8 biten az alábbi tízes számrendszerbeli számokat! (előjeles abszolút értékes, 1-es komplemens, 2-es komplemens, 127 többletes, 128 többletes)

   a. 67_{10}

   b. 108_{10}

   c. -99_{10}

   d. -117_{10}

5. Ábrázoljuk fixpontosan 16 biten az alábbi tízes számrendszerbeli számokat! (előjeles abszolút értékes, 1-es komplemens, 2-es komplemens, 2^{15} többletes, 2^{15}-1 többletes)

   a. 987_{10}

   b. 8789_{10}

   c. -356_{10}

   d. -2736_{10}

6. Adjuk meg a BCD kódját az alábbi tízes számrendszerbeli számoknak (negatívaknál a 9, és 10-es komplemest)

   a. 378_{10}

   b. 5643_{10}

   c. -864_{10}

   d. -8327_{10}

1.4. UTF-8, Unicode

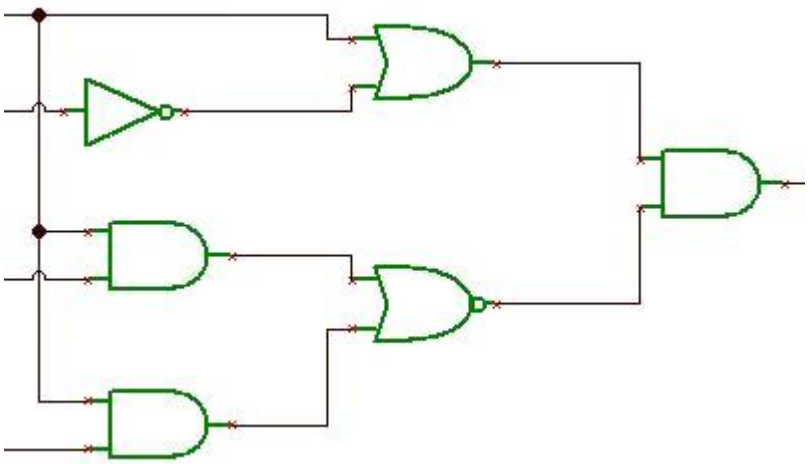
1. Adja meg a BMP sík megadott karakterének Unicode értékét és UTF-8 ábrázolását tizenhatos számrendszerben!

み ネ

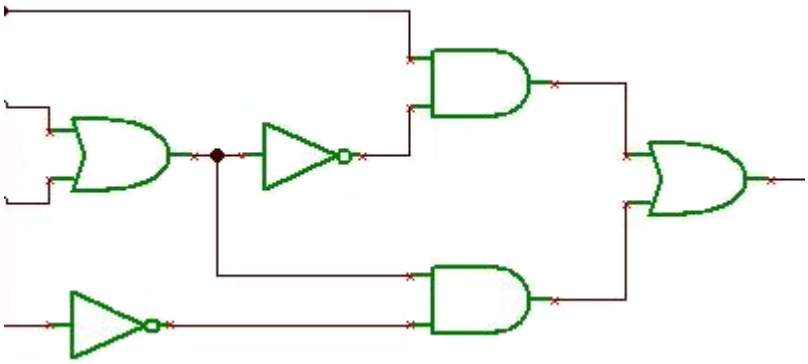
	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x3040		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
0x3050	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
0x3060	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
0x3070	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
0x3080	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
0x3090	ゐ	ゑ	を	ん	ゔ					゛	゜	ゝ	ゞ	ヽ	ヾ	

1.5. Műveletek a számítógépen

1. Írja le kétféle jelölésmódot használva az alábbi áramkört! Mi a kifejezés értéke, ha A=1, B=1, D=0 ?



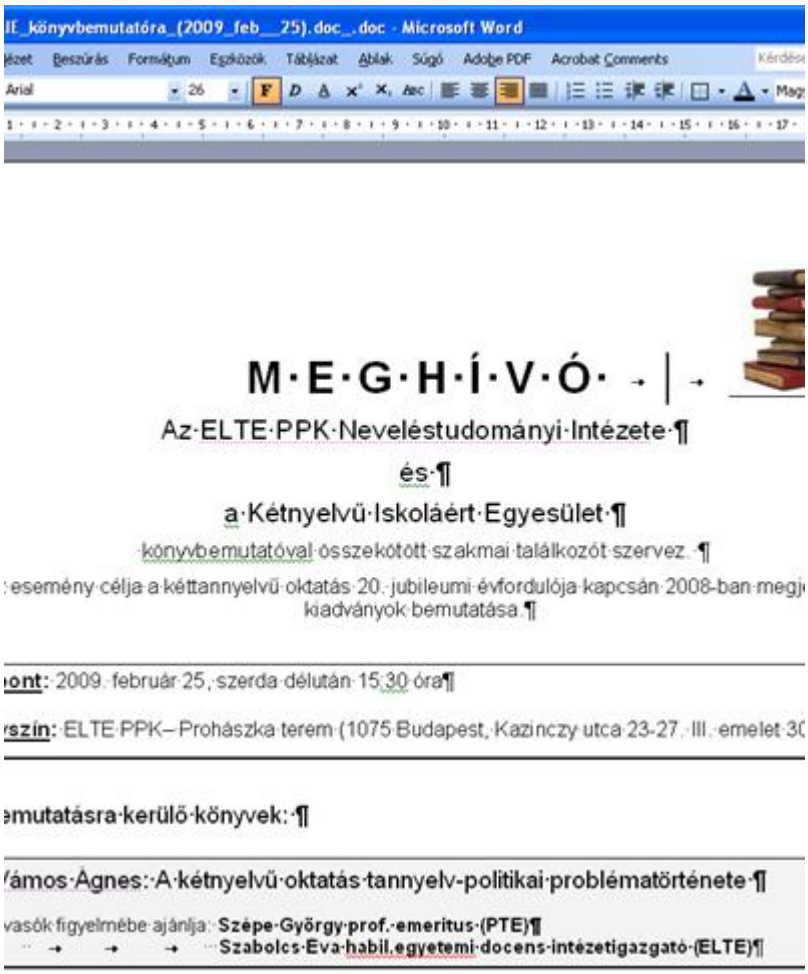
2. Írja le kétféle jelölésmódot használva az alábbi áramkört! Mi a kifejezés értéke, ha A=0, B=1, D=0 ?



2. Alkalmazások

2.1. Szövegszerkesztés

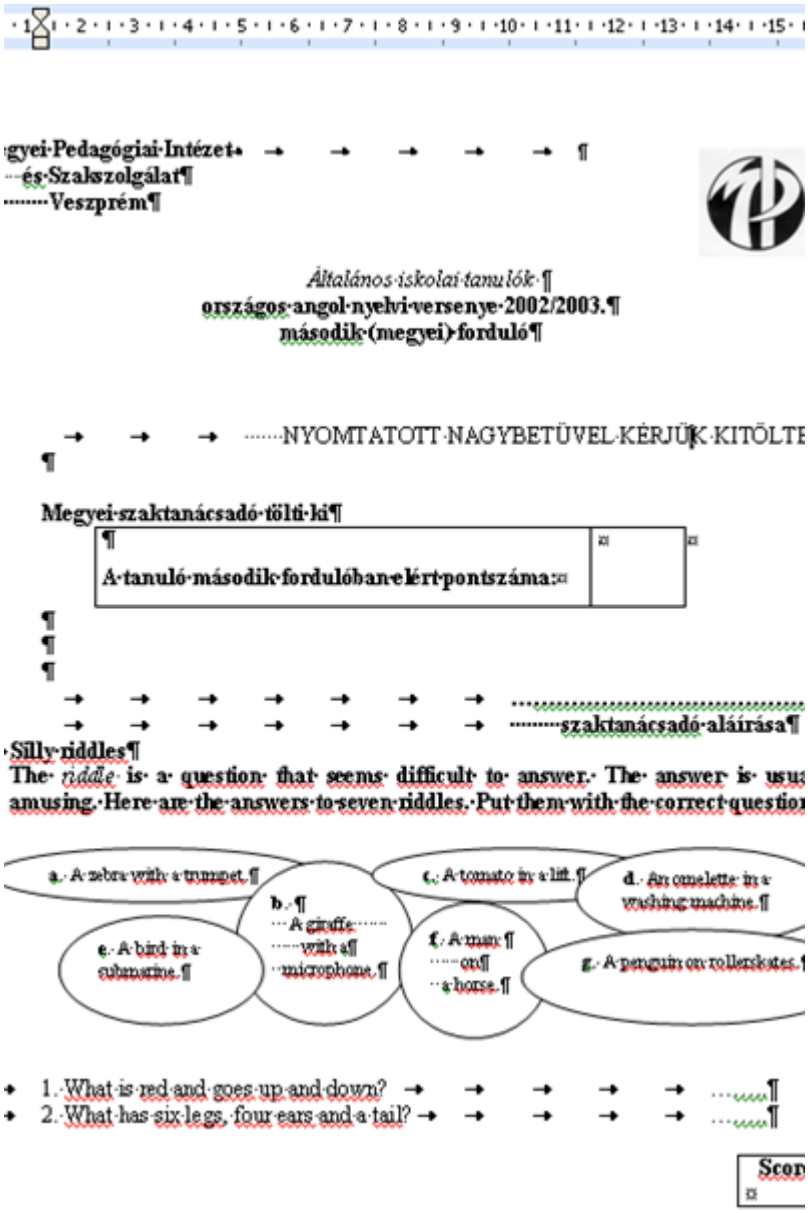
1. Sorolja fel, milyen hibákat tartalmaz az alábbi szövegrészlet! A táblázat első oszlopában nevezze meg hibát, a másodikban adja meg a hiba típusát, a harmadikban pedig a helyes megoldást! Az ábrán jelölje a hibákat az ábécé betűivel!



Hiba	Hiba típusa	Helyes megoldás
a)		
b)		

c)		
d)		
e)		
f)		
g)		
h)		
i)		
j)		

2. Sorolja fel, milyen hibákat tartalmaz az alábbi szövegrészlet! A táblázat első oszlopában nevezze meg hibát, a másodikban adja meg a hiba típusát, a harmadikban pedig a helyes megoldást! Az ábrán jelölje a hibákat az ábécé betűivel!



Hiba	Hiba típusa	Helyes megoldás
a)		
b)		

c)		
d)		
e)		
f)		
g)		
h)		
i)		
j)		

2.2. Táblázatkezelés

Figyelem! Az Excel feladatok megoldásához – paraméterezésükre vonatkozó megkötések miatt – nem használhatóak a darabteli, szumha, átlagha, szorzatösszeg, átlaghatöbb, darabhatöbb, stb., fkeres, vkeres és az adatbázis függvények!

	A	B	C	D	E	F
1	NAME	RANK	AGE	BIRTHPLACE	SERVED WITH	INCIDENT
2	Limbu, Sachin	Rifleman	23	Nepal#Rajghat, Morang in Nepal	Army	Hostile: Explosion
3	King, John	Pte	19	County Durham#Darlington	Army	Hostile: Explosion
4	Downing, Anthony	Sqn Ldr	34		RAF	Hostile: Explosion
5	Jennings, Tom	Capt	29		Royal Marines	Hostile: Explosion
6	Bond, Elijah	Sapper	24	Hampshire#Havant	Army	Hostile: Explosion
7	Steel, Sheldon	Rifleman	20	West Yorkshire#Leeds	Army	Hostile: Explosion
8	Lake, Thomas	Pte	29	Hertfordshire#Watford	Army	Hostile: Explosion
9	Boyce, David	Lt	25	Hertfordshire#Welwyn Garden City	Army	Hostile: Explosion
10	Scanlon, Richard	L/Cpl	31	Caerphilly#Rhymney	Army	Hostile: Explosion
11	Eustace, Peter	L/Cpl	25	Merseyside#Liverpool	Army	Hostile: Explosion
12	Thornton, Matthew	Pte	28	South Yorkshire#Barnsley	Army	Hostile: Explosion
13	Haseldin, Matthew	Pte	21	Yorkshire#Settle	Army	Hostile: Shot
14	Rai, Vijay	Rifleman	21	Nepal	Army	Hostile: Shot
15	Fairbrother, David	Marine	24	Lancashire#Blackburn	Royal Marines	Hostile: Shot
16	McKinlay, Jonathan	L/Cpl	33	Germany	Army	Hostile: Shot
17	Weston, Barry	Sgt	40	Berkshire#Reading	Royal Marines	Hostile: Explosion

Megjegyzés: Az utolsó rekord a munkalap 325. sorában található.

1. Írjon összetett képletet, amely visszaadja a születési hely (D oszlop) második felét. Abban az esetben is működjön a képlet, ha nincs megadva születési hely, illetve akkor is ha csak egy részből áll. A két részt a # választja el egymástól.

2. Adja meg a baleset típusát (F oszlop) a G1 cellában és egy kezdőbetűt a G2 cellában! Írassa ki egyetlen összetett függvénnyel azon katonák átlag életkorát, akik neve a megadott betűvel kezdődik és a megadott típusú balesetben vesztették életüket!

3. Adjon meg két eltérő egységet (E oszlop) a H1 és a H2 cellákban! Írassa ki azoknak a katonáknak számát, akik a két egység valamelyikében szolgáltak!

	A	B	C	D	H
1	Ki locsolt?	Hány éves	Kit locsolt?	Hány éves	Hány percet töltött ott?
2	Ács Máté	20	Belák Diána	21	15
3	Ács Máté	20	Laki Zsófia	19	15
4	Ács Máté	20	Pete Diána	20	20
5	Ács Máté	20	Pittnauer Emma	21	15
6	Ágoston Róbert	15	Bálint Zsófia	18	5
7	Ágoston Róbert	15	Hertz Csilla	22	10
8	Ágoston Róbert	15	Radnóti Andrea	22	5
9	Ágoston Róbert	15	Szegedi Zsuzsa	19	5
10	Alexa Gusztáv	22	Borbély Csilla	22	5
11	Alexa Gusztáv	22	Gáncs Ildikó	22	5
12	Alexa Gusztáv	22	Huszár Katalin	16	15
13	Alexa Gusztáv	22	Vadász Tamara	21	5

1. Mit csinál az alábbi összetett függvény?

   =HA(MARADÉK(SOR();3)=2;SZUM(HA(BAL(J2;HOSSZ(J2)-9)=A$2:A$661;H$2:H$661));HA(MARADÉK(SOR();3)=0;"";KEREKÍTÉS(ÁTLAG(HA(BAL(J2;HOSSZ(J2)-6)=A$2:A$661;H$2:H$661));0)))

	A	C	D	E	F	G	H
1	KSH_Település	Népesség (KSH)	Elterjedtség 1000 lakosra	Lakossági előfizetők	Vállalati előfizetők	Közszféra előfizetők	"Közháló program" előfizetők
2							
3	Dunaalmás	1516	108	141	20	2	2
4	Dunabogdány	2934	151	403	35	5	2
5	Dunaegyháza	1483	52	69	6	2	1
6	Dunafalva	1044	39	38	2	0	1
7	Dunaföldvár	9145	96	799	78	1	3
8	Dunaharaszti	16561	196	2980	261	7	5
9	Dunakeszi	29453	243	6267	899	0	9

1. Válaszoljon a következő rövid kérdésekre! A táblázat utolsó sora a 67. sorban található.

   a. Írjon képletet, amely megadja településenként az előfizetők számát!

   b. Adja meg képlettel a legnagyobb elterjedtséget!

   c. Adja meg képlettel a táblázatban szereplő települések számát!

   d. Adja meg képlettel a lakossági előfizetők számát!

   e. Adja meg képlettel azon települések számát, amelyekben a közszféra előfizetők száma 2!

2. K1 cellába írassa ki a következő szöveget: „x darab olyan település van a listában, amely népessége négy-jegyű szám.”, ahol x azoknak a településeknek a számát adja meg, amelyek népessége négy-jegyű szám.

3. Gépeljen egy ZZZ települést a V1 cellába, gépeljen egy XXX előfizetői típust a W1 cellába. Írjon egy képletet, amely megadja ZZZ településen hány darab BBB típusú előfizetést rendeltek!

4. Készítse el az előző feladathoz tartozó szöveget az alábbi formátumban. „ZZZ településen rendelt XXX típusú előfizetések száma:”.

3. Programozás

Az alábbi programozási feladatok erőssége és az igényelt eszköztár túlmutat a jegyzetben leírtakon. A hiányzó ismereteket a gyakorlatokon szerzik meg a hallgatók.

1. Írjunk programot, amely beolvas két valós számot két változóba, kicseréli a változók tartalmát, majd pedig kiírja a számokat fordított sorrendben. Próbáljuk megoldani a feladatot úgy is, hogy ne használjunk harmadik változót a cseréhez.

2. Írjunk programot, amely beolvas egy öt számjegyű természetes számot egy egész változóba, előállítja a fordítottját egy másik változóban, majd kiírja a két számot egymás alá a képernyőre.

3. Írjunk programot, amely beolvas négy, három számjegyű természetes számot, és kiírja a számjegyeik összegének átlagát.

4. Írjunk programot, amely kiszámítja egy adott sugarú gömb, illetve adott sugarú és magasságú egyenes henger és kúp felszínét és térfogatát. Az eredményeket táblázatos formában jelenítsük meg.

5. Írjunk programot, amely kiszámolja és kiírja egy gépkocsi féktávolságát a sebesség és az útviszonyok függvényében. A feltételezett lassulás:

   a. normál úton 4,4 m/s^2,

   b. vizes úton 3,4 m/s^2,

   c. vizes, nyálkás úton pedig 2,4 m/s^2.

   A reakcióidő 1 másodperc. A gépkocsi kezdősebessége bemeneti adat.

6. Írjunk programot, amely beolvassa egy derékszögű háromszög egyik szögének értékét fokokban, az átfogót cm-ben, és kiírja a háromszög befogóinak hosszát és a háromszög köré írható kör területét és kerületét.

7. Olvassuk be a képernyőről egy piskótatorta méreteit – átmérőjét és magasságát –, valamint a ráteendő krém vastagságát cm-ben. Számoljuk ki, mennyi krémre van szükség a torta bevonásához, ha 5%-os ráhagyással dolgozunk (gyerekek is vannak a családban...)!

8. Rajzoljuk fel logikai sémákkal az alábbi programrészleteket !

a.
```
if (F1)
if (F2) U2;
else U1;
```

b.
```
if (F1)
{if (F2) U2;}
else U1;
```

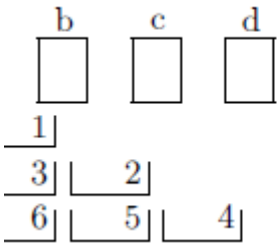
9. Írjunk programot, amely beolvas négy valós számot, és megszámolja, hány negatív!

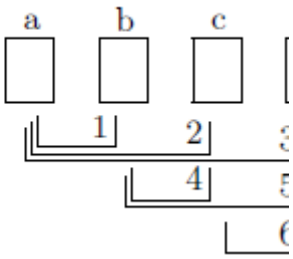
10. Írjunk programot, amely beolvas egy négyjegyű természetes számot, és összeadja külön a páros, illetve páratlan számjegyeit!

11. Írjunk programot, amely beolvas négy ötjegyű természetes számot, megszámolja, melyikben található több 5-ös számjegy, és kiír egy megfelelő üzenetet! (Használjunk long változót a számok eltárolására.)

12. Írjunk programot, amely beolvas egy négyjegyű természetes számot, és kiszámítja a prímszám jegyeinek számtani közepét!

13. Írjunk programot, amely beolvas négy valós számot, és rendezi őket csökkenő sorrendbe az alábbi stratégiák szerint:





14. Írjunk programot, amely a koordinátáival megadott P1, P2, P3 pontokról eldönti, hogy egy egyenesen találhatók-e! A kollinearitás feltétele:

$$\begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} = 0.$$

15. Írjunk programot, amely egy, a csúcspontjainak koordinátáival megadott háromszögről eldönti, hogy egyenlő szárú-e!

16. Olvassuk be egy háromszög 3 oldalát cm-ben (egy oldal legfeljebb 255 cm lehet)! Amennyiben szerkeszthető e három adatból háromszög, számítsuk ki a területét!

17. Adott két szakasz (AB, CD) a végpontjaik koordinátái által. Döntsük el, hogy metszik-e egymást, és ha igen, állapítsuk meg, melyik pontban!

Ötlet: A metszés feltétele, hogy ne legyenek párhuzamosak, és az A, illetve B pontok legyenek a CD egyenes különböző oldalain, valamint a C, illetve D pontok az AB egyenes más-más oldalán. Két pont akkor van egy egyenes különböző oldalain, ha koordinátáikat behelyettesítve az egyenes egyenletébe, ellenkező előjelű értékeket adnak.

18. Olvassunk be egy karaktert! Írjuk ki az ASCII kódját, a következő és az előző karaktert az ASCII táblában, valamint azt, hogy a beolvasott karakter nagybetű-e vagy nem!

19. Olvassunk be egy karaktert, majd írjuk ki, hogy nagybetű, kisbetű, szám, speciális karakter vagy egyéb!

20. Olvassunk be egy dátumot: év, hó, nap. Írjuk ki, hogy ez a dátum az év hányadik napja!

Ötlet: Helyezzük egy break nélküli switch utasításba a hónapokat, ezek fordított sorrendje szerint.

21. Adott két egyenes két-két pontjuk koordinátái által. Határozzuk meg egymáshoz viszonyított helyzetüket (párhuzamosak ($m_1=m_2$), metszők, merőlegesek ($m_1m_2=-1$)), ahol m_1 és m_2 a két egyenes iránytényezői.

22. Adottak egy kör középpontjának koordinátái és a sugara. Ellenőrizzük egy pont, egy egyenes, illetve egy másik kör hozzá viszonyított helyzetét!

Ötlet: Kiszámítjuk a kör középpontjának távolságát a ponttól, az egyenestől és a másik kör középpontjától.

23. Ugyanaz a feladat, de a körök három pont által adottak. Adott középpontú és sugarú kör egyenlete:

$$X - x_0)^2 + (Y - y_0)^2 = r^2.$$

Három nem kollineáris ponton átmenő kör egyenlete:

$$\begin{vmatrix} X^2+Y^2 & X & Y & 1 \\ x_1^2+y_1^2 & x_1 & y_1 & 1 \\ x_2^2+y_2^2 & x_2 & y_2 & 1 \\ x_3^2+y_3^2 & x_3 & y_3 & 1 \end{vmatrix} = 0$$

Megjegyzések:

− Csak az ismert utasításokat használjuk!

− Használjunk minimális számú változót!

− Használjuk mind a **scanf**, **printf**, mind az **fscanf**, **fprintf** függvényeket!

24. Generáljuk a következő számsorozatok esetében az első *n* elemet, és egy másik megoldással a sorozat *n*-edik elemét:

a. 1, 2, 3, 4, 5, . . .

b. 1, 3, 5, 7, . . .

c. 0, 2, 4, 6, 8, . . .

d. 2, 3, 5, 7, 11, 13, 17, 19, . . .

e. 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, . . .

f. 1, 1, 2, 1, 2, 3, 1, 2, 3, 4, . . .

g. 1, 2, 1, 3, 2, 1, 4, 3, 2, 1, . . .

h. 1, 2, 2, 3, 3, 3, 4, 4, 4, 4, . . .

i. 2, 3, 3, 4, 4, 4, 5, 5, 5, 5, . . .

j. 0, 1, 2, 1, 2, 2, 3, 2, 3, 3, 3, 4, 3, 4, 4, 4, 4, 5, . . .

k. 1, 1, 2, 2, 1, 2, 3, 3, 3, 1, 2, 3, 4, 4, 4, 4, . . .

25. Számítsuk ki a következő kifejezések értékét:

a. E1 = 1 + 2 + 3 + . . . + n

b. E2 = −1 + 2 − 3 + 4 − 5 + 6 + . . . + $(-1)^n \cdot n$

c. E3 = 1 · 2 + 3 · 4 + 5 · 6 + . . . + (2n − 1) · 2n

d. E4 = 1/(1 + 2) · 2/(2 + 3) · 3/(3 + 4) · . . . · n/[n + (n + 1)]

e. E5 = −2 + 3 − 5 + 7 − 11 + 13 − 17 + 19 − . . . (n-edik tag)

f. E6 = −1/1+(1 · 2)/(1+2)−(1 · 2 · 3)/(1+2+3)+. . .+$(-1)^n$(1 · 2 · 3 · . . . · n)/(1 + 2 + 3 + . . . + n)

g. E7 = 1+2+3−4+5−6−7+8−9−10−11−12+13−14 . . . (n-szer)

h. E9 = 1/2 + 4/3 + 9/5 + 16/7 + 25/11 + 36/13 + . . . (n-edik tag)

26. Egy számsorozat elemein végezzük el az alábbi feladatokat:

Összeg-, átlag-, szorzatszámítás, számoljuk meg a párosok, illetve páratlanok számát, ellenőrizzük, hogy tartalmaz-e teljes négyzetet, állapítsuk meg a minimum értékét, az első minimum pozícióját, az utolsó minimum pozícióját, számítsuk ki a párosok médiáját, a páratlanok maximumát, a prímek számát!

A számsorozat elemeit állítsuk a program rendelkezésére:

a) Véletlenszám-generátorral *n* darab 100-nál kisebb természetes számot!

b) Véletlenszám-generátorral egész számokat a [-10, 10] intervallumból, amíg eltaláljuk a 0-t!

c) Állományból olvasva egy *n* elemű egész számsorozatot!

d) Egész számokat olvasva a billentyűzetről 0 végjelig!

27. Olvassunk be egy 0-val végződő, egészekből álló számsorozatot! Írjuk ki a harmadik legnagyobb elemét és annak sorszámát is! Valamely feltétel nem teljesülése esetén (például, ha 3-nál kevesebb elemünk van) a program adjon hibajelzést.

28. Mit valósít meg az alábbi programrészlet? Írd át úgy, hogy ugyanezt valósítsa meg, de do-while ciklussal ! A feladatot papíron oldjuk meg!

```
int x, k = 0;
scanf("%d", &x);
while(x) k++;
printf("%d", k);
```

29. Melyek a végtelen ciklusok? A feladatot papíron oldjuk meg!

```
a. i = 10; while(i-);
b. while(i = 10) i-;
c. for(x = 1; x = 10; x++);
d. for(x = 1; x == 10; x++);
```

30. Írjunk programot, amely kiszámítja és másodpercenkénti bontásban táblázatosan kiírja a *v0* kezdősebességű, a gyorsulással egyenletesen gyorsuló mozgást végző test által *t* idő alatt megtett utat ! A *v0,* a és *t* bemenő adatok.

31. Írjunk programot, amely a Pascal-háromszög első *m* sorát állítja elő (*m* bemenő adat)! A Pascal-háromszög egyes elemei az úgynevezett *binomiális együtthatók*. Az *n*-edik sor *k*-adik eleme:

$C(n, k) = n! /((n-k)! \cdot k!) = (n \cdot (n-1) \cdot (n-2) \ldots (n-k+1))/(1 \cdot 2 \cdot 3 \cdot \ldots \cdot k)$.

32. Egy *k* egész számot *majdnemprímnek* nevezünk, ha nem prím, de ugyanakkor két prím szorzata. Ha *k* és *k + 1* is majdnemprímek, akkor *iker majdnemprímek*. Írjunk programot, amely 100-ig megkeresi az iker majdnemprímeket!

33. Írjunk programot, amely kiszámítja és kiírja az olyan püthagoraszi számhármasokat, melyeknek egyik tagja sem nagyobb 50-nél!

34. Ábrázoljuk a képernyőn az

$F(x) = (\sin x)/x$

függvényt a $[-2\pi , 2\pi]$ intervallumon úgy, hogy a függőlegesen futó *x* tengely mentén a megfelelő koordinátájú pontokba egy * karaktert helyezünk! Rajzoljuk ki az *x* tengelyt is!

35. Olvassuk be egy sakkfigura (vezér, király, bástya, futó, huszár, gyalog) aktuális koordinátáit a sakktáblán! Írjunk programot, amely kiírja a sakkfigura által támadható mezőket!

36. Írjunk programot, amely for ciklussal számítja ki az m^n értékét, ahol az *m* egy valós szám, az *n* egész típusú!

37. Írjuk ki azokat az 500-nál kisebb, legalább kétjegyű páros számokat, amelyekben a tízesek helyén páratlan szám áll!

38. Adott egy kör a síkban középpontjának és sugarának koordinátái által. Írjunk programot, amely megszámlálja, hogy hány darab egész koordinátákkal jellemezhető koordinátapont esik a körön belülre!

39. Olvassunk be tanulmányi átlagokat, és határozzuk meg a megfelelő minősítéseket: kitűnő [9,50–10], jeles [9–9,50), jó [8–9), közepes [7–8), elégséges [4,50–7), elégtelen [1–4,50).

Megjegyzés. A 40–44 feladatokban használjuk ki a **scanf** és **printf** függvények formázó karakterei nyújtotta lehetőségeket.

40. Olvassunk be decimális számokat nulla végjelig, és alakítsuk oktálissá (8-as számrendszer) őket!

41. Olvassunk be decimális számokat nulla végjelig, és alakítsuk headecimálissá őket!

42. Olvassunk be hexadecimális számokat nulla végjelig, és alakítsuk decimálissá őket!

43. Olvassunk be oktális számokat nulla végjelig, és alakítsuk decimálissá őket!

44. Olvassunk be bináris számokat nulla végjelig, és alakítsuk decimálissá őket!

45. Olvassunk be 1000-nél kisebb decimális számokat nulla végjelig, és alakítsuk binárissá őket!

46. Olvassunk be természetes számokat a billentyűzetről, és számoljuk ki az átlagukat! A számsor végét két egymás utáni 0 jelezze!

47. Írjunk programot, amely beolvas 20 valós számot, és megállapítja a leghosszabb szigorúan növekvő összefüggő részsorozat összegét!

48. Írjunk programot, amely beolvas n egész számot, és kiírja a szomszédos számok különbségét!

49. Adott egy természetes szám. Határozzuk meg a szám számjegyeinek szorzatát, összegét, átlagát, a legkisebbet, a legnagyobbat, valamint a páros, a páratlan és a prímszámjegyek számát.

50. Oldjuk meg a 64. feladatot *n* darab egész szám esetében!

51. Oldjuk meg a 64. feladatot *n* darab véletlenszerűen generált egész számra a (200, 3000) intervallumból!

52. Adott egy természetes szám, képezzük a fordított számot!

53. Hány olyan háromjegyű egész szám van, amely prím, és amelynek a fordítottja is prím?

54. Adott egy természetes szám, képezzük azt a számot, amelyet úgy kapunk, hogy felcseréljük az első és utolsó számjegyeit.

55. Hány olyan háromjegyű egész szám van, amely prím és a belőle az előbbi módon képzett szám is prím?

56. Adott egy *n* elemű számsorozat. Számoljuk meg, hogy a 2-es számrendszerbeli alakjaikban számonként és összesen hány 1-es és hány 0 van!

57. Olvassunk be pozitív egész számokat hexadecimális alakban 0 végjelig, és állapítsuk meg a számsorozat rendezettségét (egyenlő elemekből áll-e, növekvő, csökkenő vagy rendezetlen)! A növekvő, illetve a csökkenő rendezettségnél az egyenlő számokat is megengedjük. Ha a sorozatról menet közben kiderül, hogy rendezetlen, fejezzük be a bevitelt!

58. Írjuk ki az 1 m^3-nél kisebb térfogatú, 10 *cm*-enként növekvő sugarú gömbök térfogatát!

59. Írjuk ki az angol ábécé összes nagybetűjét növekvő, majd csökkenő sorrendben!

60. Olvassunk be egy dátumot: év, hó, nap! Írjuk ki, hogy ez a dátum az év hányadik napja!

61. Rajzoljunk adott méretű X-eket a képernyőre! A méreteket a rajzolás előtt olvassuk be, és amikor 0 méretet olvastunk, a program befejeződik.

62. Tegyünk a képernyő közepére egy jelet, majd fel, le, balra, jobbra nyilak segítségével mozgassuk a képernyőn! A jelet a képernyőről ne engedjük kimenni, s az kezdetben húzzon maga után vonalat! Az ≤insert> gomb hatására, ha eddig volt vonalhúzás, ne legyen, ha nem volt, legyen, ≤enter>-re a program fejeződjön be.

63. Írjunk programot, amely 0 végjelig olvas be számokat! A bevitt számot csak akkor fogadjuk el, ha az előző számtól való eltérés (a két szám különbsége abszolút értékben) annak 20%-ánál nem nagyobb.

64. Generáljunk véletlenszám-generátorral háromjegyű számokat, amíg olyat találunk el, amelynek számjegyei csökkenő sorrendben vannak! Írjuk ki sorszámozva a generált számokat!

65. Generáljunk véletlenszám-generátorral számokat az int tartományból, amíg olyat találunk el, amelynek számjegyei tető alakot írnak le (egy pontig növekednek, majd csökkennek)! Írjuk ki sorszámozva a generált számokat!

66. Adott egy *n* csúcspontú sokszög a csúcsai (az óra járásának sorrendjében) koordinátái által. Írjuk ki a sokszög területét!

$$_{\text{poligon}} = \pm\frac{1}{2} \cdot \sum_{i=1}^{n} \begin{vmatrix} x_i & y_i \\ x_{i+1} & y_{i+} \end{vmatrix}$$

ahol jelölés szerint a képzeletbeli *n+1*-edik pont koordinátái: (x_1, x_2).

67. Adott *n* esetén határozzuk meg az *n*-nél nagyobb legkisebb prím értékét.

68. Határozzuk meg az *n1* és *n2* természetes számok közé eső *ikerprímeket* (*p* és *q* iker-prímek, ha prímek, és $p-q = 2$, amennyiben $p > q$).

69. Számítsuk ki egy személy korát napokban kifejezve. (Adott a születési dátuma és az aktuális dátum.)

70. Adott *n* és *m* természetes számok. Írjuk ki az alábbi számsorozat *n* egymásutáni elemét, az *m*-edik elemmel kezdődően.

1, 2, 3, 2, 5, 2, 3, 7, 2, 4, 3, 2, 5, 11, 2, 3, 4, 6, 13, ...

(Minden összetett számot helyettesítünk a saját osztóival.)

71. Adott *n* és *m* természetes számok. Írjuk ki az alábbi számsorozat *n* egymásutáni elemét, az *m*-edik elemmel kezdődően.

1, 2, 3, 4, 2, 5, 6, 2, 3, 7, 8, 2, 9, 3, 10, 2, 5, 11, ...

(Minden összetett szám után beszúrjuk a prím osztóival.)

72. Adott *n* és *m* természetes számok. Írjuk ki azt az $n \times n$ méretű mátrixot, amelynek elemei (soronkénti bejárás szerint) azonosak az alábbi számsorozat *n2* egymás utáni elemével, az *m*-edik elemmel kezdődően.

1, 2, 2, 3, 2, 3, 2, 3, 3, 3, 2, 3, 2, 3, 3, 3, 2, ...

(Minden prím számot 2-essel és minden összetett számot 3-assal helyettesítettünk)

73. Adott *n* és *m* természetes számok. Írjuk ki azt az $n \times n$ méretű mátrixot, amelynek elemei (soronkénti bejárás szerint) azonosak az alábbi számsorozat első *n2* egymás utáni elemével.

1, 2, 3, 4, 5, 6, 7, 8, 9, 1, 0, 1, 1, 1, 2, 1, 3, 1, 4, ...

(Minden számot helyettesítettünk a számjegyeivel.)

74. Adott *n* és *m* természetes számok. Irjuk ki azt az $n \times n$ méretű mátrixot, amelynek elemei (soronkénti bejárás szerint) azonosak az alábbi számsorozat *n2* egymás utáni elemével, az *m*-edik elemmel kezdődően.

1, 2, 3, 4, 2, 5, 6, 2, 3, 7, 8, 2, 4, 9, 3, 10, 2, 5, 11, ...

(Minden összetett szám után beszúrtuk a saját osztóit.)

75. Adott *n* természetes szám p_1, p_2, ... , p_n számrendszerekbeli ábrázolása. Határozzuk meg a legnagyobb számot, illetve a számok összeget *q* alapú számrendszerben.

76. Adott egy egész szám, amelyet egy long típusú változóba tárolunk el. Forgassuk körkörösen (ami kiesik egyik felől, az jön be a másik felén) jobbra/balra a belső ábrázolása bitjeit, és írjuk ki az így kapott számokat belső ábrázolásukkal együtt!

77. Ugyanaz a feladat, de a forgatást úgy valósítjuk meg, hogy a legkisebb helyértékű és legnagyobb helyértékű bitek fixen maradjanak (a többit forgatjuk maguk között).

78. Adott egy egész szám, amelyet egy long típusú változóba tárolunk el. Állítsuk elő azt a számot, amelyet úgy kapunk, hogy a belső ábrázolása szomszédos bitjeit (0,–1., 2,–3., . . .) kicseréljük egymás között!

79. Adott egy *x* természetes szám az unsigned long tartományból, valamint az *a, b, c, d* 0..31 közti természetes számok. *x* belső ábrázolásának az *a*-adik bitjét állítsuk 0-ra, a *b*-edik helyértékű bitjét 1-re, a *c*-edik bitjét tagadjuk, majd léptessük *d* pozícióval balra. Írjuk ki az eredeti számot és belső ábrázolását, majd az újonnan nyert számot és belső ábrázolását!

80. Adott *n* egész szám az **int** tartományból. Csomagoljuk össze négyenként a belső ábrázolásuk bitjeit egy-egy hexadecimális számjegybe, és írjuk ki az így kapott 16-os számrendszerbeli számot!

81. Adott egy *n* elemű valós számokat tartalmazó sorozat. Ellenőrizzük, szimmetrikus-e, ha nem, írjuk ki a tükörképét.

82. Mit ír ki még az alábbi programrészlet? Fogalmazd meg általánosan! A feladatot papíron oldjuk meg!

```
int i, a[10]={5, 3, 8, 2, 9, 0, 7, 1, 6, 4};
for(i=0; i<10; i++)
{
 if(a[0]>a[i]) {v=a[0]; a[0]=a[i]; a[i]=v;}
 if(a[9]<a[i]) {v=a[9]; a[9]=a[i]; a[i]=v;}
}
printf("%d, %d", a[0],a[9]);
```

83. *Bináris keresés:* Adott egy *n* elemű, szigorúan növekvő egész számokat tartalmazó sorozat, valamint egy *x* egész szám. Keressük meg *x*-et a számsorozatban az alábbi algoritmus szerint. Ha megtalálható, írjuk ki a sorszámát, ha nem, akkor egy megfelelő üzenetet.

− Összehasonlítjuk *x* -et a számsorozat középső elemével.

− Ha egyenlők, akkor megtaláltuk és kiíratjuk a sorszámát.

− Ha nem egyenlők, akkor folytatjuk a keresést (ugyanilyen módon) vagy a felső, vagy az alsó számsorozatszakaszban.

− Ha nulla hosszúságú szakaszhoz jutunk, ez azt jelenti, hogy *x* nem található meg a sorozatban.

84. Karaktereket olvasunk be * végjelig. Készítsünk karakterelőfordulási statisztikát.

85. Adott egy természetes szám, írjuk ki a kettes számrendszerbeli alakját.

86. Írjunk programot, amely megállapítja, hogy egy 50 adatból álló mérési sorozat adatai közül hány tér el 10%-nál kevésbé az átlagértéktől.

87. Írjunk programot, amely egy *n* hosszúságú, egészeket tartalmazó adatsorban megkeresi egy adott érték utolsó előfordulási helyét (sorszámát). Írja ki a program azt is, hogy ez az adott érték hányadik előfordulása.

88. Töltsünk fel egy 20 elemű egész számokat tartalmazó tömböt számokkal. Írassuk ki ezek közül sorrendben a három legkisebb értékűt, és azt is, hogy hányadik helyen helyezkednek el!

89. Adott *n* valós szám. Alakítsuk halmazzá őket (vagyis töröljük azokat az elemeket, amelyek egy bizonyos értéket másodszor, harmadszor stb. Tartalmaznak).

90. Írjuk ki a (1, 2, . . . , n) halmaz összes részhalmazát.

91. Adott két halmaz. Számítsuk ki a két halmaz egyesített halmazát, metszetét, különbségét.

92. Adott két halmaz. Ellenőrizzük, hogy részhalmaza-e valamelyik valamelyiknek.

93. Adott *n* halmaz. Számítsuk ki az *n* halmaz egyesített halmazát, metszetét, különbségét.

94. Adott két halmaz. Számítsuk ki a Descartes-szorzatukat.

95. Írjuk ki egy adott halmaz összes részhalmazát!

96. Adott egy *n* elemű számsorozat. Rendezzük növekvő, illetve csökkenő sorrendbe

a. buborékos rendezéssel,

b. minimum-kiválasztásos rendezéssel,

c. maximum-kiválasztásos rendezéssel.

Megszámoljuk, hány elem kisebb egy adott elemnél, és így megtaláljuk végleges helyét a rendezett sorozatban.

97. Adott egy *n* elemű számsorozat. Rendezzük maguk között külön a páros, illetve külön a páratlan pozíciókban lévőket.

98. Adott egy *n* elemű számsorozat. Rendezzük maguk között külön a pozitívakat, illetve külön a negatívakat.

99. Adott egy *n* elemű számsorozat és egy *k* szám (*n* legyen k többszöröse). Rendezzük úgy a számsorozat első *k* elemét, hogy növekvő részsorozatot alkossanak, a második *k* darab elemét úgy, hogy csökkenő részsorozatot alkossanak, a harmadik *k* darabot megint növekvő sorrendbe, stb.

100. Adott *n* darab szigorúan növekvő számsorozat. Fésüljük össze őket egyetlen szigorúan növekvő számsorozattá.

101. Adott egy *n* elemű számsorozat, minden két eleme közé szúrjuk be az átlagukat

a. segédtömbbel,

b. segédtömb nélkül.

102. Adott egy *n* elemű számsorozat. Töröljük ki belőle a prímeket

a. segédtömbbel,

b. segédtömb nélkül.

103. Adott egy *n* elemű számsorozat. Cseréljük fel egymás között az első minimumot és az utolsó maximumot.

104. Írjunk programot, amely beolvas egy személyi számot (CNP), és kiírja az illető személy nemét és születési dátumát.

105. Melyek helyesek? A feladatot papíron oldjuk meg!

1. char s1[];

2. char s2;

3. char *s3;

4. char s4="A";

5. char s5=’A’;

6. char *s6="A";

7. char s7[]="A";

8. char s8[1]="A";

106. Írjunk programot, amely egy legtöbb 20 elemű karaktersorozatról eldönti, hogy palindrom-e. *Palindromnak* olyan szöveget nevezünk, amely balról jobbra és jobbról balra olvasva ugyanaz, a sorközöktől és írásjelektől eltekintve.

Például: Géza, kék az ég.

107. Írjunk programot, amely a képernyőről olvas be számokat. A bevitt számot csak akkor fogadjuk el, ha az ténylegesen szám, egész és az előző számtól való eltérés 20%-nál nem nagyobb.

108. Adott két azonos hosszúságú, különböző karaktereket tartalmazó karakterlánc (bármely karakter egyszer fordulhat elő), amelyek egymás anagrammái. Ellenőrizzük az adatok helyességét, majd sokszorozzuk meg az első karakterlánc karaktereit annyiszor, ahányadik pozícióban található az illető karakter a második karakterláncban.

Például: "szilva", "vaszil" => "sssszzzzziiiiiillllllvvaaa"

109. Egy bemeneti karakterlánc szavakat tartalmaz szóközzel elválaszva. Töröljük a legrövidebb szavakat.

110. Adott két, azonos hosszúságú karakterlánc. Az első betűket, a második számjegyeket tartalmaz. Ellenőrizzük az adatok helyességét, majd az első minden betűjét sokszorozzuk annyiszor, amennyi a második karakterláncból sorszám szerint megfelelő számjegy értéke.

Például: "abcd" ,"3124" => "aaabccdddd"

111. Egy bemeneti karakterlánc bináris számokat tartalmaz szóközzel elválasztva. Készítsünk egy másik karakterláncot, amelyben megőrizzük (szóközzel elválasztva) azon bináris számok tízes számrendszerbeli alakját, amelyeknek bináris alakja páros számú 1-est tartalmaz.

Például: "10010 1101 10001 111 1000 1100" => "18 17 12"

112. Egy bemeneti karakterláncban helyettesítsük egy adott karakterlánc többszöri előfordulásait egy másik, adott, karakterlánccal.

113. Olvassunk be a billentyűzetről szavakat, és építsünk fel egy karakterláncot, amely betűrendi sorrendben tartalmazza a szavakat.

114. Adott egy szöveg. Ha a szöveg tartalmaz számjegyeket, változtassuk meg úgy, hogy minden számjegyet annyiszor sokszorozzunk meg, mint amennyi a számjegy értéke.

115. Adott egy szöveg. Ha a szöveg tartalmaz számjegyeket, helyettesítsünk a szövegben minden számjegyet annyi (*) karakterrel, mint amennyi a számjegy értéke.

116. Adott egy mondat. Írjuk át „madárnyelvre" (minden ≤magánhangzó>-t helyettesítsünk a ≤magánhangzó>p≤magánhangzó> karakterlánccal).

Például : "Jó reggelt" => "Jópó repeggepelt".

117. Adott egy karakterlánc, amely szavakat tartalmaz szóközzel elválasztva. Tükrözzük külön mindenik szót, és írjuk ki az így nyert karakterláncot. (Az alábbi feladatok esetében az adatokat állományból olvassuk be.)

118. Egy $N \times M$ méretű mátrixot a *V[0 .. N*M-1]* tömbben tárolunk el sorfolytonosan. Tegyük fel, hogy a mátrix *i*-edik sorának *j*-edik eleme a tömb *k*-adik pozíciójába kerül (a sorokat, illetve oszlopokat 0-tól kezdődően számozzuk).

a. Fejezzük ki *k*-t, *i* és *j* függvényében!

b. Adjuk meg *i*-t, illetve *j*-t, *k* függvényében!

c. Ha *N=M*, akkor milyen tömbpozíciókba kerülnek a főátló elemei? (Általánosan fogalmazd meg.)

d. Ha *N=M*, akkor a mátrix mely elemei kerülnek a tömb azon pozícióiba, amelyek indexeinek *N*-nel való osztási maradéka maximális? (Általánosan fogalmazd meg.)

119. Írjunk programot, amely beolvas egy 5×5-ös mátrixot, és kiírja az úgynevezett tükörmátrixát. A tükörmátrix elemeit az eredeti mátrixelemek főátlóra tükrözésével kapjuk. A tükörmátrix az eredeti helyén képződjön.

120. Adott egy $n \times n$ méretű mátrix, amely különböző egész számokat tartalmaz. Ellenőrizzük, hogy bűvös négyzet-e. A bűvös négyzet minden sorának és oszlopának, valamint átlóinak összege azonos.

121. Töltsünk fel egy $n \times n$ méretű tömböt a következő jelentésű számhármasok alapján:

– sorindex (egész szám az *[1, n]* intervallumból),

– oszlopindex (egész szám az *[1, n]* intervallumból),

– érték (valós szám).

A beolvasás 0 végjelig tart. A program a sorindex és oszlopindex ellenőrzése után az általuk meghatározott helyre írja az értéket. Felülírás nem megengedett! A tömb többi eleme 0 legyen.

122. Egy véletlenszerűen feltöltött $n \times n$ méretű karaktermátrix sorait alkotó karakterekből építsünk karakterláncokat, majd ezeket fűzzük össze betűrendi sorrendbe, szóközzel elválasztva.

123. Egy állományból kiolvasott négyzetes karaktermátrix fő- és mellékátlóin található karakterekből építsünk karakterláncokat. Vizsgáljuk meg, hogy ezek tükörszavak-e, illetve, hogy anagrammák-e.

124. Adott *n*, *m* és egy $n \times m$-es mátrix. Döntsük el, hogy tartalmaz-e egynél több nullát a főátló felett vagy a mellékátló alatt.

125. Olvassunk be egy négyzetes mátrixot, amely valós számokat tartalmaz. Állítsuk elő azt a mátrixot, amelyet úgy nyerünk, hogy minden elemét helyettesítjük a float típusnak megfelelő belső ábrázolás 0-ás és 1-es bitjei számának különbségeivel.

126. Adott egy mátrix. Ellenőrizzük, mely sorai vannak növekvő sorrendben, és mely oszlopai csökkenőben.

127. Adott egy mátrix. Rendezzük a sorait külön-külön növekvő sorrendbe.

128. Adott egy mátrix. Rendezzük az oszlopait külön-külön csökkenő sorrendbe.

129. Adott egy mátrix. Rendezzük át az elemeit úgy, hogy sorfolytonos bejárás szerint csökkenő sorrendben legyenek.

130. Adott egy mátrix. Írjuk ki az elemeit csigabejárás, illetve fordított csigabejárás szerint. Oldjuk meg a feladatot egyetlen ciklus-utasítást használva a bejáráshoz.

131. Írjunk programot, amely kiszámítja egy *n*-ed rendű determináns értékét.

Ötlet: a főátló alatt képezzünk nullákat (Gauss-módszer).

132. Írjunk programot, amely egy iterativ algoritmus révén – használva egy megfelelő *c* tömböt – kiszámítja

a) a c[1][n] értéket az alábbi képlet alapján:

```
c[i][i] = 0, bármely i∈{1, 2, ..., n}
c[i][j] = min_{i<=k<j} {c[i][k]+c[k+1][j]+d[i-1]* d[k]*d[j]}
bármely i,j∈{1, 2, ..., n} és i<j;
ahol ismert a d[0..n] tömb tartalma
```

b) a c[n][m] értéket az alábbi képlet alapján:

```
c[i][0] = 0, bármely i∈{1, 2, ..., n}
c[0][j] = 0, bármely j∈{1, 2, ..., m}
c[i][j] = c[i-1][ j-1]+ 1 ha a[i]=b[j]
c[i][j] = min {c[i-1][ j], c[i][j-1]}
ha a[i]≠b[j] bármely i∈{1, 2, ..., n} és j∈{1, 2, ..., m};
ahol az a[1..n] és b[1..m] tömbök tartalma ismertnek tekintendő
```

c) a $\max_{1\leq k\leq n}$ {c[k]} értéket az alábbi képlet alapján:

```
c[n] = 1
c[i] = max_{j=i+1, n} {1+c[j]|a[i]<=a[j]}
bármely i∈{1, 2, ..., n-1}; ahol ismert az a[1..n] tömb tartalma
```

133. Adott egy négyzetes mátrix. Számítsuk ki a determinánsa értékét!

134. Adott n darab számsorozat. Mindenikben határozzuk meg a leghosszabb egymásutáni elemekből álló szigorúan növekvő részsorozatot, majd fésüljük össze ezeket egyetlen növekvő sorozattá.

135. Egy $n \times m$ méretű mátrix sorai egész elemű halmazokként értelmezzük. Határozzuk meg az n halmaz egyesítését, illetve metszetét.

136. Adott N darab $n \times n$ méretű négyzetes mátrix. Határozzuk meg azon mátrixok összegét, amelyek determinánsa nem nulla, illetve azt a mátrixot, amelyiknek determinánsa maximális értékű.

137. Adott n darab halmaz. Jelölje $I(M)$ azon halmazok számát, amelyek valódi részhalmazként tartalmazzák az M halmazt. Határozzuk meg azt az M halmazt, amely esetén maximális az $I(M)$ értékű.

138. Adottak egy $n \times m$ méretű ritka-mátrix nem nulla elemei a (sorindex, oszlopindex, érték) hármasok által. Adjunk össze, illetve szorozzunk össze két ily módon megadott mátrixot.

139. Adott egy n elemű természetes számsorozat. Készítsünk számjegy előfordulási statisztikát.

140. Adott egy n elemű természetes számsorozat. Írjuk ki az egymás után következő tükör-számpárokat. (Például : 3251, 1523)

141. Adott egy n elemű természetes számsorozat. Írjuk ki az összes maximális hosszú egymás utáni elemekből álló csupa prímeket tartalmazó részsorozatot.

142. Adott egy n elemű természetes számsorozat. Írjuk ki az összes maximális hosszú egymás utáni elemekből álló részsorozatot, amelyekben a szomszédos elemek "rokon számok" (ugyanazokat a számjegyeket tartalmazzák; Például: 12545, 5412).

143. Adott egy n elemű természetes számsorozat. Írjuk ki az összes maximális hosszú egymás utáni elemekből álló részsorozatot, amelyekben a szomszédos elemek "idegen számok" (nem tartalmaznak egyetlen közös számjegyet sem; Például: 12545, 7893).

144. Adott egy $n \times n$ méretű négyzetes mátrix (legyen a neve A). Határozzuk meg az A,A2,A3, . . . ,An mátrixokat, illetve ezek összegét.

145. Adott egy n-ed fokú egész együtthatójú polinom vagy a monomai által vagy fokszám és együtthatók által. Határozzuk meg a racionális gyökeit és ezek multiplicitását.

146. Legyen az alábbi algoritmus:

```
0. ALGORITMUS
1.  beolvas n
2.  minden i = 1,n végezd
3.   beolvas v[i]
4.   nrc[i] = 0
5. vége minden
6.  minden i = 1,n-1 végezd
7.   minden j = i+1,n végezd
8.     ha v[i] > v[j]
9.     akkor nrc[i]++
10.    különben nrc[j]++
11.    vége ha
12.   vége minden
13.  vége minden
14. VÉGE ALGORITMUS
```

a. Mennyi lesz az nrc[3] értéke, ha az 5,7,3,1,8,10 értékeket olvassuk be?

b. Mennyi lesz az nrc[3] értéke, ha *n*-nek 5-öt olvasunk be, a *v* vektorba pedig azonos értékeket?

c. Ha azonos értékeket olvasunk be a *v* vektorba, milyen szabályosságot fognak mutatni az nrc tömb elemei?

d. Ha *n=7*, hányszor hajtódik végre a ha utasítás?

e. Fogalmazd meg tömören, mit valósít meg az algoritmus!

f. Hogyan fognak megváltozni az a) – e) pontokra adott feleletek, ha a ha utasítás feltételét *v[i] >= v[j]* változtatjuk?

147. Legyen az alábbi algoritmus:

```
0. ALGORITMUS
1.  beolvas n
2.  minden i = 1,n végezd
3.   minden j = 1,n végezd
4.    a[i][j] = i + j
5.   vége minden
6.  vége minden
7.  minden i = 1,n végezd
8.   a[i][i] = a[i][i] % a[6][4]
9.  vége minden
10. VÉGE ALGORITMUS
```

a. Az *a* tömb hány eleme lesz 3-mal egyenlő értékű az algoritmus végén, ha *n*-nek 10-et olvasunk be?

b. Ha *n*-nek 10-et olvasunk be, mennyi lesz a tömb legnagyobb elemének értéke az algoritmus végén?

c. Ha *n*-nek 10-et olvasunk be, hány maximális eleme lesz a tömbnek az algoritmus végén?

d. Az *a* tömb hány eleme lesz 10 az 1–6 programsorok végrehajtása után, ha *n*-nek 7-et olvasunk be?

e. Milyen halmazból vesznek értéket a főátló elemei az algoritmus végén, ha *n >= 6* értéket olvasunk be?

f. Mi mondható el a mellékátló elemeiről az algoritmus végén, ha *n*-nek páros értéket olvasunk be?

148. Adott *n* személy neve, címe (város, utca, házszám) és telefonszáma egy bemeneti állományban. Írjuk át egy kimeneti állományba névsor szerint a budapestiek nevét és telefonszámát (a rendezést egy pointertömb segítségével valósítsuk meg, anélkül hogy effektíve rendeznénk a struct tömböt, ahova a személyek adatait beolvastuk).

149. Adott *n* személy neve, *k* tantárgy és mindeniknek minden tantárgyból az átlaga. Írjuk ki névsor szerint a személyek nevét és mindeniknek az átlagát csökkenő sorrendben.

150. Milyen értéket ír ki az alábbi programrészlet? Feltételezzük, hogy a **char** 1 bájton, a **long** pedig 4 bájton van tárolva.

```
union elem {char b[4]; long c;} x;
 x.c = 0;
 x.b[0]^=1;
 x.b[1]|=1;
 x.b[2]^=1;
 x.b[3]|=1;
 printf("%ld",x.c);
```

151. Milyen értéket ír ki az alábbi programrészlet? Feltételezzük, hogy a unsigned **char** 1 bájton, a **unsigned long** pedig 4 bájton van tárolva.

```
union elem {
 unsigned char b[4];
 unsigned long c;
} x;
 x.c = 0;
 x.b[0]^=1;
 x.b[1]&=1;
 x.b[2]&=2;
 x.b[3]|=1;
 printf("%ld",x.c);
```

152. Legyen az alábbi C programrészlet:

```
typedef struct
{
 float b[77];
 int a;
} elemke;
elemke w[77], *q = w+1;
```

a. Hivatkozz *w* utolsó elemének, első mezőjének utolsó elemére.

b. Állítsd a *q* címen levő struktúra második mezőjének legkisebb helyértékű bitjét 1-re.

c. Hány bájt memória terület kerül lefoglalásra a fenti definíciók nyomán?

153. Legyen az alábbi C programrészlet:

```
typedef struct
{
 int a;
 double b[100];
} elem1;

typedef struct
{
 int c;
 elem1 d[100];
} elem2;
 elem2 v[100], *p = v;
```

a. Hivatkozz *v* utolsó elemének, második mezőjének, utolsó elemének első mezőjére.

b. Hivatkozz a *p* címen levő struktúra második mezőjének, utolsó elemének, második mezőjének utolsó elemére.

154. Adott *n* személy neve, telefonszáma, születési dátuma és helye, valamint lakcíme.

a. Írjuk ki a neveket és születési dátumokat születési dátum szerint csökkenő sorrendben (azonos év esetén vegyük figyelembe a hónapot, . . .)

b. Hányan laknak a szülővárosukban?

c. Kiknek jelenik meg a telefonszámukban a házszámuk?

155. Adott *n* személy neve és neme.

a. Lehetséges-e őket párosítani egy tánchoz, és ha igen, adjunk meg egy ilyen párosítást.

b. Adjuk meg azt a párosítást, amelyben a lehető legtöbb Jancsi és Juliska, illetve Csongor és Tünde páros van.

156. Adott két bemeneti állomány. Az egyik tartalmazza *n* ország nevét és fővárosát, a másik *m* fővárost és lakosainak számát. Készítsünk kimutatást egy kimeneti állományba azon országokról és fővárosaik lakosságáról, amelyek esetében rendelkezünk ezekkel az adatokkal.

157. Adott egy bemeneti állományban *m* tanuló neve és ezek osztályfőnökei. Készítsünk kimutatást egy kimeneti állományban arról, hogy az egyes osztályfőnököknek hány diákjuk van.

158. Adott három bemeneti állomány. Az első tartalmazza *n* tanfolyam nevét és árát, a második *m* személy nevét és születési évét, a harmadik a beiratkozási nyilvántartást, minden sorban egy tanulót és egy tanfolyamot (egy tanuló maximum 5 tanfolyamra iratkozhat be, minden tanfolyamhoz legalább 4, legfennebb 15 diák szükséges).

a. Ellenőrizzük, hogy a megkötéseket figyelembe vették-e.

b. Mely tanfolyamok a leglátogatottabbak, és melyek a legkevésbé?

c. Kik látogatják a legtöbb tanfolyamot, és kik a legkevesebbet?

d. Mely tanfolyamokat látogatják a legfiatalabb diákok, és melyeket a legidősebbek?

e. Melyik tanfolyam hozza a legtöbb pénzt, és mennyi pénz jön összesen be?

f. Készítsünk kimutatást egy kimeneti állományba a tanfolyamokról, és az egyes tanfolyamokat látogató hallgatókról.

g. Készítsünk kimutatást egy kimeneti állományba a hallgatókról és a hallgatók választotta tanfolyamokról.

159. Adott három bemeneti állomány. Az első tartalmazza az osztályfőnököket és osztályuk megnevezését, a második a tanulókat és hogy melyik osztályba járnak, a harmadik pedig az osztályfelelősök listáját. Írjuk ki az osztályfelelősöket és osztályfőnökeiket.

160. Adott *n* egész szám a **long** tartományból. Titkosítsuk a számsorozatot úgy, hogy a számok belső ábrázolásainak bájtjait annyiszor forgatjuk körkörösen, ahányadik a szám a sorozatban. A páratlan pozíciójúakat balra, a páros pozíciójúakat pedig jobbra forgassuk.

161. Adott egy szöveges állományban *n* valós szám. Titkosítsuk a számsorozatot úgy, hogy a **float**-kénti belső ábrázolásuk középső két bitjét (a 15. és 16. helyértékű biteket) felcseréljük maguk között.

– A titkosítást **union**-ok és bitmezők által valósítsuk meg.

– A titkosításhoz használjuk a bitműveleteket.

(Egyes feladatok esetében szükséges lehet a függvények megtervezése. Például melyek legyenek a paraméterei, hogyan történjen a paraméterátadás stb. Mindenik feladatban, amennyiben erre szükség van, az állományneveket parancssorból vegyük át.)

162. Helyettesítsd a kérdőjeleket úgy, hogy szintaktikailag helyes, működőképes programot kapj, amelyben nem kerül sor egyetlen implicit típuskonverzióra sem.

```
#include <stdio.h>
?
jonatan alma;
main()
{
 printf("%lf",alma(1,1.0,'A'));
 return 0;
}
? alma (? a,? b,? c)
{return ? a +? b+? c;}
```

163. Írjunk függvényt, amely a paraméterként kapott

a. természetes számról eldönti, hogy prím-e,

b. természetes számnak visszatéríti a fordítottját,

c. két természetes számnak meghatározza a legnagyobb közös osztóját,

d. két természetes számnak meghatározza a legkisebb közös többszörösét,

e. természetes számból törli a nulla számjegyeket, és visszatéríti a kitörölt számjegyek számát,

f. természetes számot úgy alakítja át, hogy rendezve legyenek a számjegyei.

164. Írjunk függvényt, amely paraméterként megkapja egy tömb címét és a benne tárolt számsorozat elemeinek számát, és

a. ellenőrzi, hogy a számsorozat elemei növekvő sorrendben vannak-e,

b. képezi a számsorozat szimmetrikusát,

c. rendezi a számsorozatot az elemek utolsó számjegyei szerint,

d. töröl egy harmadik paraméterként kapott értéket a rendezett számsorozatból,

e. beszúr egy harmadik paraméterként kapott értéket a rendezett számsorozatba,

f. számokat olvas be 0 végjelig, és ezeket beszúrja a rendezett számsorozatba.

165. Írjunk függvényt, amely paraméterként megkap egy polinomot és egy értéket, és visszatéríti a polinom érétkét az adott helyen.

166. Írjunk függvényeket, amelyek paraméterként megkapnak két polinomot, és meghatározzák az összeg, különbség, szorzat, hányados és maradék polinomokat.

167. Írjunk függvényt, amely paraméterként megkap

a. egy karakterláncot, és eldönti, palindrom-e,

b. két karakterláncot és eldönti, hogy egymás anagrammái-e.

168. Írjunk saját függvényeket, amelyek az alábbi könyvtári függvényeket valósítják meg: **strlen, strcmp, strcpy, strcat, strchr, strstr.**

169. Írjunk függvényt, amely megvalósítja egy karakterlánc beszúrását egy másik karakterláncba egy adott helyre.

170. Írjunk függvényt, amely kitöröl egy karakterláncból egy adott helyről adott számú karaktert.

171. Írjunk függvényt, amely egy mátrix sorait, illetve oszlopait úgy rendezi át, hogy a főátlón lévő elemek növekvő sorrendben legyenek. Két sor, illetve oszlop felcseréléséhez használjunk segédfüggvényt.

172. Írjunk függvényeket, amelyek két mátrixnak meghatározzák az összegét, különbségét, szorzatát.

173. Írjunk függvényt, amely dinamikusan létrehoz paraméterként megadott méretű egydimenziós tömböt, feltölti véletlen egészekkel egy paraméterként megadott intervallumból, majd visszatéríti a tömb címét.

174. Írjunk függvényt, amely dinamikusan létrehoz paraméterként megadott méretű kétdimenziós tömböt, feltölti véletlen egészekkel egy paraméterként megadott intervallumból, majd visszatéríti a tömb címét.

175. Írjunk függvényt, amely dinamikusan létrehoz egy olyan karakterláncot, amely az angol ábécé nagybetűit tartalmazza ábécésorrendben.

176. Adott n személy neve és címe (város, utca, szám) egy szöveges állományban. Van-e olyan személy, akinek városneve az utcaneve fordítottja? (Függvénnyel ellenőrizd a két karakterlánc palindromságát.) Hány személy lakik prím szám alatt? (Függvénnyel ellenőrizd egy számról, hogy prím-e.)

177. Írjunk olyan kereső függvényt, amely bármilyen típusú tömbben megkeres egy elemet (paraméterként kapjon egy összehasonlító függvényt).

178. Írjunk olyan kereső függvényt, amely visszatéríti bármilyen típusú tömb legkisebb elemének címét (paraméterként kapjon egy összehasonlító függvényt).

179. Rendezzük **qsort**-tal egy kétdimenziós tömb sorait úgy, hogy az első oszlop elemei csökkenő sorrendben legyenek.

180. Adottak *n* személy adatai (név, cím, születési dátum). Rendezzük az adatokat **qsort**-tal a következő kritériumok szerint: név, város, születési év, születési dátum.

181. Adottak *n* személy adatai (név, cím, születési dátum), amelyeket egy **struct** típusú tömbben tárolunk. Egy **struct-pointer** típusú tömb elemeit állítsuk rá a **struct** tömb elemeire. Rendezzük a pointereket **qsort**-tal az általuk megcímzett struct-ok következő mezői szerint : név, város, születési év, születési dátum. Írjuk ki a rendezett adatsorokat (a beolvasáshoz és kiíráshoz is használjunk függvényt).

182. Írjunk függvényt, amely bármilyen típusú mátrix elemeit rendezi sorfolytonos bejárás szerint (paraméterként kapjon egy összehasonlító függvényt).

183. Írjuk meg a **myscanf** függvényt néhány típusra.

184. Írjunk rekurzív függvényt, amely meghatározza egy paraméterként kapott természetes szám

a. számjegyeinek összegét,

b. számjegyeinek szorzatát,

c. eldönti róla, hogy tartalmaz-e 5-ös számjegyet,

d. 1-es számjegyeinek számát,

e. legkisebb számjegyét,

f. legnagyobb számjegyét,

g. fordítottját.

185. Írjunk rekurzív függvényt, amely meghatározza egy paraméterként kapott számsorozat

a. elemeinek összegét,

b. legkisebb elemének értékét,

c. legnagyobb elemének pozícióját,

d. pozitív elemeinek számát,

e. tartalmaz-e páros elemet.

186. Írjunk rekurzív függvényt, amely egy bináris számsorozatnak meghatározza a paritását. A paritás 1, ha az egyesek száma páratlan, ellenkező esetben 0.

187. Írjunk rekurzív függvényt, amely beszúró rendezéssel rendez egy számsorozatot.

188. Írjunk rekurzív függvényt, amely kiírja egy számsorozat elemeit fordított sorrendben.

189. Írjunk rekurzív függvényt, amely egész számokat olvas be nulla végjelig, majd kiírja az utolsó n számot fordított sorrendben.

190. Írjunk rekurzív függvényt, amely kiírja egy számsorozat pozitív elemeit a beolvasás sorrendjében, a negatívakat pedig a beolvasás fordított sorrendjében.

191. Írjunk rekurzív függvényt, amely egész számokat olvas be vakon (írjunk egy ilyen beolvasó függvényt) nulla végjelig, majd kiírja a páratlan sorszámúakat a beolvasás sorrendjében, és alájuk a páros sorszámúakat fordított sorrendben.

192. Írjunk rekurzív függvényt, amely kiírja egy karakterlánc páratlan sorszámú szavait a beolvasás sorrendjében, és alájuk a páros sorszámúakat fordított sorrendben. Írjunk rekurzív függvényt, amely ugyanazt valósítja meg, mint az strcmp könyvtári függvény.

193. Írjunk rekurzív függvényt, amely visszatéríti az *n*-edik Fibonacci számot.

194. Írjunk rekurzív függvényt, amely visszatéríti a *C(n,k)* értékét.

195. Írjunk rekurzív függvényt, amely egy *n* méter hosszú rudat feldarabol egy- és kétméteres rudakra az összes lehetséges módon.

196. Írjunk rekurzív rendező függvényt **(quick sort)** a szétválogatás algoritmusra.

197. Írjunk rekurzív rendező függvényt **(merge sort)** az összefésülő algoritmusra.

198. Adottak *n* személy adatai (vezetéknév, keresztnév, születési dátum – év, hónap, nap –, cím – város, utca, házszám). Írj programot, amely menüből választhatóan rendezi az adatokat valamely mező szerint. Használjuk a **qsort** könyvtári függvényt, és készítsünk egy függvény pointertömböt, amelynek elemei a mezők szerinti összehasonlító függvényekre mutassanak.

199. Mit ír ki az alábbi programrészlet (anélkül válaszoljunk, hogy lefuttatnánk a programot)?

```
int f(int a){return ++a;}
int ff(int *a){return ++(*a);}
int g(int a){return f(++a)++;}
int g1(int *a){return f(++(*a))++;}
main()
{
 int a=1;
 printf("%d\n%d\n%d\n%d",f(a), ff(&a), g(++a), g1(&a));
 return 0;
}
```

200. Egy szöveges állományból kiolvasunk egy *n* elemű egész számsorozatot egy dinamikusan lefoglalt tömbbe.

a. Határozzuk meg a számok legkisebb számjegyeinek az összegét.

b. Rendezzük (**qsort**) a számokat úgy, hogy legkisebb számjegyeik szerint csökkenő sorrendben legyenek! A rendezett számsorozatot írjuk egy kimeneti állományba!

201. Adott *n*, illetve *n*n* tanuló neve és telefonszáma egy szöveges állományban. A tanulók adatait egy $n \times n$ méretű kétdimenziós struct tömbben tároljuk el. Írj egy olyan függvényt, amely különböző számolási műveleteket tud végezni a kétdimenziós tömbben eltárolt struct mátrix főátló alatti háromszögében, attól függően, milyen függvényt kap paraméterként.

a. Számoljuk meg azokat a személyeket a főátló alatt, akik neve négyszavas!

b. Számoljuk meg azokat a személyeket a főátló alatt, akiknek a telefonszáma palindrom?

202. Egy szöveges állományból kiolvasunk egy *n* elemű egész számsorozatot egy dinamikusan lefoglalt tömbbe.

a. Határozzuk meg a számok számjegyei szorzatának az összegét.

b. Rendezzük (**qsort**) a számokat úgy, hogy számjegyeik szorzata szerint növekvő sorrendben legyenek! A rendezett számsorozatot írjuk egy kimeneti állományba!

203. Adott *n*, illetve *n* tanuló neve és telefonszáma egy szöveges állományban. A tanulók adatait egy *n* elemű egydimenziós structtömbben tároljuk el. Írj egy olyan függvényt, amely különböző ellenőrzési műveleteket tud végezni az egydimenziós tömbben eltárolt adatokon, attól függően, milyen függvényt kap paraméterként.

a. Ellenőrizzük, hogy van-e olyan személy a főátló felett, akinek a keresztneve és vezetékneve azonos!

b. Ellenőrizzük, hogy van-e olyan személy a főátló felett, akinek a telefonszáma számjegyei növekvő sorrendben követik egymást!

204. Írj egy függvényt, amely bármely elemtípusú és elemszámú tömb elemeit szétválogatja az első elem szerint, anélkül hogy rendezné a tömböt.

205. Adott egy szöveges állományban *n* apa keresztneve, gyerekei száma és ezek keresztneve.

a. Írjuk ki egy másik állományba az apák adatait gyerekszám szerint csökkenő sorrendbe rendezve (**qsort**).

b. Hány Barni (Barnika, Barna) nevű gyerek van?

c. Készíts statisztikát, hogy az egyes apáknak összesen hány Barni fiuk van.

206. Írj függvényt, amely visszatéríti a paraméterként kapott *n*-nél nagyobb legkisebb prím értékét.

207. Írj függvényt, amely visszatéríti az *n*-edik Fibonacci számot.

208. Írj függvényt, amely eldönti, hogy szám tökéletes-e. (a szám egyenlő a nála kisebb osztói összegével).

209. Írj függvényt, amely implementálja az Euler függvényt. (meghatározza az *n*-nél kisebb *n*-el relatív prím számok számát).

210. Írj függvényt, amely visszatéríti a paraméterként kapott mátrix rangját.

211. Írj függvényt, amely kiszámítja két *p* számrendszerbeli szám összegét *p* számrendszerben.

212. Hogyan működik az alábbi program? (backslash karakterrel több sorba tördelhetők a **#define** sorok.)

```
#include <stdio.h>
#define def_fn(name,operation)\
int name(int a, int b)\
{\
  int c = a operation b;\
```

```
  printf("%d %s(%s) %d = %d\n", a, #name, #operation, b, c);\
  return c;\
}\
def_fn(plus,+);
def_fn(minus,-);
main()
{
 plus(1,1);
 minus(1,1); return 0;
}
```

213. Mit ír ki az alábbi program? Az M_PI konstans a pi értékét tartalmazza. A feladatot fejben oldjuk meg!

```
#include <stdio.h>
#define kerulet(r) 2 * M_PI * r
double terulet(double r){return M_PI * r * r;}
main()
{
 printf("%lf\n%lf", kerulet(1+2), terulet(1+2));
 return 0;
}
```

13. fejezet - Vizsga minták

1. Mérnök informatikus szakon

1.1. I. Minta vizsgafeladatsor

1. Mi a repeater?
2. Írja fel hexadecimális számrendszerben az alábbi decimális számot: 512.41_{10}.
3. Végezze el 16 biten fixpontosan az alábbi kivonást: $1615_{10} - 972_{10}$.
4. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: p and not q or p and (q or p)? Írja le, hogyan gondolkodott!
5. A Neumann-elv.
6. Adatátviteli közegek számítógép-hálózatokban.
7. Milyen információt hordoz egy IP-cím?
8. Mi a vírus, és milyen jelek utalnak vírusfertőzésre?
9. Mi a szemantika?
10. A számítógép elvi felépítése.
11. Rendezze névsor szerint buborékrendezéssel: szilva, alma, körte, dió, meggy, kiwi!
12. Mi az assembly?
13. Mi az egyéni számformátum.
14. A bekezdés és formázási lehetőségei.
15. Mi az URL?

1.2. II. Minta vizsgafeladatsor

1. Írja fel bináris számrendszerben az alábbi decimális számot: 512.41_{10}.
2. Végezze el 16 biten fixpontosan az alábbi kivonást: $1615_{16} - 972_{16}$.
3. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: p or not q or p and (q or p)? Írja le, hogyan gondolkodott!
4. Mi a cashe?
5. A buborékrendezés.
6. Rendezze növekvő sorrendbe a következő mennyiségeket: 5 Mbyte; 5 byte; 5 kbyte; 5 bit; 1 byte; 1 bit; 1 Gbyte!
7. Adjon meg algoritmust, amely eldönti, hogy két nullától különböző egész szám közül melyik a nagyobb! Készítsen folyamatábrát!
8. Mi a szemantika?
9. A lebegőpontos számábrázolás.
10. Mi a mnemonik?

11. A hardver és a szoftver meghatározása.
12. Mi a file?
13. Mi a vágólap?
14. A bekezdés és formázási lehetőségei.
15. Mi a ROM?

1.3. III. Minta vizsgafeladatsor

1. Mi az ASCII és mit tud róla?
2. Írja fel hexadecimális számrendszerben az alábbi decimális számot: 399.27_{10}.
3. Végezze el 16 biten fixpontosan az alábbi kivonást: $1590_{16} - 447_{16}$.
4. A számítógép elvi felépítése.
5. A megszakítási folyamat.
6. Az ISO-OSI modell.
7. Az operációs rendszer és feladatai.
8. Mi a szemantika?
9. Az algoritmus fogalma, tulajdonságai.
10. Egy nevezetes algoritmus ismertetése.
11. A számítógépvírus definíciója.
12. Mi a vágólap?
13. Mi az élőfej és az élőláb? Mikor használjuk?
14. Mit nevezünk relatív cellahivatkozásnak?
15. Mi a .ppt?

1.4. IV. Minta vizsgafeladatsor

1. Mi az ASCII és mit tud róla?
2. Írja fel hexadecimális számrendszerben az alábbi decimális számot: 399.27_{10}.
3. Végezze el 16 biten fixpontosan az alábbi kivonást: $1590_{16}16 - 447_{16}$.
4. A számítógép elvi felépítése.
5. A megszakítási folyamat.
6. Az ISO-OSI modell.
7. Az operációs rendszer és feladatai.
8. Mi a szemantika?
9. Az algoritmus fogalma, tulajdonságai.
10. Egy nevezetes algoritmus ismertetése.

11. A számítógépvírus definíciója.
12. Mi a vágólap?
13. Mi az élőfej és az élőláb? Mikor használjuk?
14. Mit nevezünk relatív cellahivatkozásnak?
15. Mi a .ppt?

1.5. V. Minta vizsgafeladatsor

1. Mi a bit?
2. Írja fel hexadecimális számrendszerben az alábbi decimális számot: 399.27_{10}.
3. Végezze el 16 biten fixpontosan az alábbi kivonást: $1590_{16} - 447_{16}$.
4. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: p and not q or p and (q or p)? Írja le, hogyan gondolkodott!
5. A Neumann-elv.
6. Számítógép-architektúrák.
7. Memóriakezelési technikák.
8. Mit jelent a batch, illetve az interaktív üzemmód?
9. Mi a szemantika?
10. A Turing-gép.
11. A folyamatábra és elemei.
12. A gyorsrendezés.
13. A körlevélkészítés lépései.
14. Mit nevezünk relatív cellahivatkozásnak?
15. Ismertessen öt internet szolgáltatást?

1.6. VI. Minta vizsgafeladatsor

1. Mi a byte?
2. Írja fel decimális számrendszerben az alábbi hexadecimális számot: 512.41_{16}.
3. Végezze el 16 biten fixpontosan az alábbi kivonást: $1952_{10} - 447_{10}$.
4. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: p and not q or p and (q or p)? Írja le, hogyan gondolkodott!
5. A Neumann-elv.
6. A perifériák.
7. Mi a SPAM?
8. Az internet és szolgáltatásai.
9. Mi a szintaktika?

10. A számítógép programozási lehetőségei.
11. A Church-tézis.
12. A gyorsrendezés.
13. A körlevélkészítés lépései.
14. Adatok kitöltése soron vagy oszlopon belül.
15. A tömörítés.

1.7. VII. Minta vizsgafeladatsor

1. Írja fel bináris számrendszerben az alábbi decimális számot: 128.25_{10}.
2. Végezze el 16 biten fixpontosan az alábbi kivonást: $1A1E_{16} - B54_{16}$.
3. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: (p or not q) or not p and not (q or p)? Írja le, hogyan gondolkodott!
4. Melyik szám standard IEEE lebegőpontos ábrázolása a következő:

   1 10000001 10011110000111100100001
5. Rendezze növekvő sorrendbe az alábbi mennységeket: 0,5 Mbyte; 500 kbyte; 500 byte; 0,5 kbyte; 5 Gbyte!
6. Rendezze névsor szerint buborékrendezéssel: szilva, alma, körte, dió, meggy, kiwi!
7. Mi a szemantika?
8. Mit nevezünk relatív cellahivatkozásnak?
9. Mi a vágólap?
10. Mi a hardver?
11. Mi a RAM?
12. Mi az alábbi program outputja és hol jelenik meg:

```
#include <stdio.h>
main()
{
 FILE * f;
 FILE * g;
 int a;
 f = fopen("be.txt", "r");
 g = fopen("ki.txt", "w");
 fscanf(f, "%d", &a);
 fprintf(g, "A szám: %d", a);
 fclose(f); fclose(g);
 return 0;
}
```

13. Mit tud az EPIC-ről?
14. A Nuemann-elv.
15. A számítógépi vírus definíciója és fajtái.

1.8. VIII. Minta vizsgafeladatsor

1. Írja fel bináris számrendszerben az alábbi decimális számot: $128{,}25_{10}$.
2. Végezze el 16 biten fixpontosan az alábbi kivonást: $1A1E_{16} - B54_{16}$.
3. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: (p or not q) or not p and not (q or p)? Írja le, hogyan gondolkodott!
4. Melyik szám standard IEEE lebegőpontos ábrázolása a következő:

   1 10000001 10011110000111100100001
5. Rendezze növekvő sorrendbe az alábbi mennységeket: 0,5 Mbyte; 500 kbyte; 500 byte; 0,5 kbyte; 5 Gbyte!
6. Rendezze névsor szerint buborékrendezéssel: szilva, alma, körte, dió, meggy, kiwi!
7. Mi a szemantika?
8. Mit nevezünk relatív cellahivatkozásnak?
9. Mi a vágólap?
10. Mi a hardver?
11. Mi a RAM?
12. Mi az alábbi program outputja és hol jelenik meg:

```
#include <stdio.h>
main()
{
 FILE * f;
 FILE * g;
 int a;
 f = fopen("be.txt", "r");
 g = fopen("ki.txt", "w");
 fscanf(f, "%d", &a);
 fprintf(g, "A szám: %d", a);
 fclose(f); fclose(g);
 return 0;
}
```

13. Mit tud az EPIC-ről?
14. A Nuemann-elv.
15. A számítógépi vírus definíciója és fajtái.

1.9. IX. Minta vizsgafeladatsor

1. Írja fel decimális számrendszerben az alábbi bináris számot: 1101001.011_2.
2. Végezze el 16 biten fixpontosan az alábbi kivonást: $1219_{10} - 345_{10}$.
3. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: p and q or p and not q or p? Írja le, hogyan gondolkodott!
4. Ábrázolja standard IEEE lebegőpontos formában a következő számot: 64,62510.
5. Rendezze növekvő sorrendbe a következő mennyiségeket: 1 Mbyte; 0,5 kbyte; 1 kbyte; 1 byte; 8 bit; 1 Gbyte!
6. Adjon meg algoritmust, amely eldönti, hogy két nullától különböző egész szám azonos előjelű-e! Készítsen folyamatábrát!

7. Mi az USB?
8. Mi jelent a szintaktikai hiba a programozásban?
9. Jellemezze a trójai faló típusú vírust!
10. Mi a HTML?
11. Milyen a WYSIWYG szövegszerkesztő?
12. Mit csinál az alábbi program:

```
 #include <stdio.h>
main()
{
 int n, p = 1;
 scanf("%d", &n);
 p *= n % 10;
 n /= 10; p *= n % 10;
 n /= 10; p *= n % 10;
 n /= 10; p *= n % 10;
 n /= 10;
 printf("%d", p);
 return 0;
}
```

13. Buborékrendezés.
14. A számítógép elvi felépítése.
15. Számítógépgenerácók.

1.10. X. Minta vizsgafeladatsor

1. Írja fel bináris számrendszerben az alábbi decimális számot: $128{,}25_{10}$.
2. Végezze el 16 biten fixpontosan az alábbi kivonást: $1A1E_{16} - B54_{16}$.
3. Mi az alábbi logikai kifejezés eredménye, ha p=1 és q=0: (p or not q) or not p and not (q or p)? Írja le, hogyan gondolkodott!
4. Melyik szám standard IEEE lebegőpontos ábrázolása a következő:

   1 10000001 10100000000000000000000
5. Rendezze növekvő sorrendbe az alábbi mennységeket: 0,5 Mbyte; 500 kbyte; 5 byte; 500 byte; 0,5 kbyte; 5 bit!
6. Nevezze meg az alábbi elemi algoritmust és rajzolja meg a folyamatábráját:

```
read szám1, szám2;
összeg1 := szám1;
összeg2 := szám2;
while összeg1 ≠ összeg2 do
 if összeg1 < összeg2 then
  összeg1 := összeg1 + szám1;
 else
  összeg2 := összeg2 + szám2;
 endif
endwhile
write összeg1.
```

7. Mi a szemantikai hiba a programozásban?

8. Mit nevezünk abszolút cellahivatkozásnak?

9. Mi a vágólap?

10. Mi a hardver?

11. Mi a RAID?

12. Mi az alábbi program outputja és hol jelenik meg:

```
#include <stdio.h>
main()
{
int n, a[100], i, x;
FILE *be, *ki;
 be = fopen ("kereses_be.txt", "r");
 ki = fopen ("kereses_ki.txt", "w");
 fscanf (be, "%d", &n);
 for (i = 0; i < n; ++i)
  fscanf (be, "%d",&a[i]);
  fscanf (be, "%d", &x);
 for (i = 0; i < n; ++i)
  if (a[i] == x)
   { fprintf (ki, "%d", i + 1); break; }
  if (i == n)
   fprintf(ki, "Nincs.");
 fclose (be); fclose (ki);
return 0;
}
```

13. Mit tud a RISC-ről?

14. A Neumann-elv.

15. Mi az NTFS?

2. Programtervező informatikus szakon

2.1. I. Minta vizsgafeladatsor

1. Számoljuk át tízes számrendszerbe az alábbi számokat:

   a. 143.04_8=

   b. $1B2.C4_{16}$=

2. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat:

   a. $EC85B_{16}$=

   b. 1011010101001101_2=

3. Írjuk át 10-es számrendszerbe:

   a. $2472.24,2,2,\ldots_8$=

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

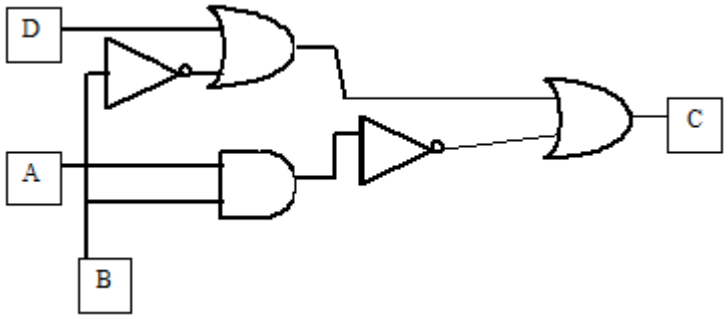
   a. $11111.10_2 + 1111.11_2$=

   b. $1011.10_2 - 1101.00_2$=

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

   a. $1A72C.16_{16} - C853A.25_{16}=$

6. Írjuk le logikai kifejezéssel az alábbi áramkört. Mi a kifejezés értéke, ha A=B=D=0 (hamis)?



7. Melyik IEEE 754 számot ábrázoltuk?

   0 10010011 01111000000000000000000

8. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint:

   -0.5321_{10}

9. Ábrázoljuk nyolcas számrendszerben 107-at: előjelbittel kezdve, a kitevő legyen 1 jegyű (3 bites), 4-többletes, a törtrész 4 jegyű.

10. Ábrázoljuk 16-os számrendszerben −5324197.247-et: előjelbittel kezdve, a kitevő legyen 1 jegyű (4 bites), 8-többletes, a törtrész 4 jegyű.

11. Adjuk meg a –57 abszolútértékes, komplemens ábrázolásait, 1-es, 2-es és 127-többletes.

12. Írjon C programot n! (n faktoriális) rekurzív előállítására.

13. Bizonyítsa be indukcióval, hogy:

$$\sum_{i=0}^{n} 2^i = 2^{n+1} - 1$$

14. Rajzolja fel az első Generáció blokkvázát!

15. Mit valósit meg az álabbi programrészlet? Írjuk át úgy, hogy ugyanezt valósítsa meg, de do-while ciklussal!

```
int x, k = 1;
scanf("%d", &x);
while(x)
{
 scanf("%d", &x);
 k++;
}
printf("%d", k-1);
```

16. Mi lesz a kiírás eredménye?

```
int B[25],  k=10;
while (k < 22)
    B[k] =  (++k, 2*(k+1));
```

```
printf("%d\n %d \n", *(B+14), B[k-2]);
```

17. Írjunk makrót, mely visszaadja két egész szám maximumát.

18. Mi a kiíratás eredménye?

```
i=3; j=1; k=2;
k=-i+j++;
printf("%d  %d  %d\n", i,j,k);
```

19. Mennyi az a változó értéke az alábbi programrészlet végrehajtása után?

```
int a = 8
int *p;
p=&a;
*p=a+*p-2;
printf("%d\n",a);
```

20. Mennyi az s változó értéke az alábbi programrészlet végrehajtása után?

```
double a1=1.0, b1=3.0;
double s;
s = a1 + 1/b1 * 2 + 64;
printf("%x\n",(char)s);
printf("%d\n",(char)s);
```

21. Mi lesz a k változó értéke az alábbi programrészlet végrehajtása után?

```
int i, j, k;
i=7; j=5;
if(i>=j && 5)
 k=-12;
else
 k=1;
printf("%d\n",k);
```

22. Mi lesz a dd változó értéke az alábbi programrészlet végrehajtása után?

```
char q=5;
int m=3;
double dd,d=2.20;
dd = q+m*d;
printf("%d\n", (int)dd);
```

23. Mi a kiíratás eredménye?

```
double matrix [3][3]={1.2, 3.4, 4.5,6.7, 7.8, 8.9};
printf("%.2f, %f\n", matrix[1][1], matrix[2][1] );
```

24. Mi lesz a j változó értéke az alábbi programrészlet végrehajtása után?

```
int k=0, j=0;
int i;
for(i=1;i<=4;i++) j=k++;
```

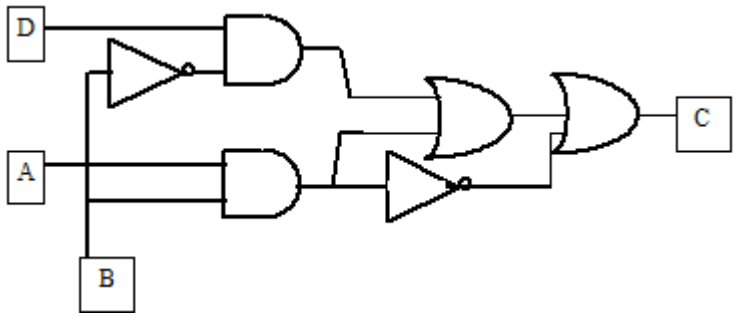
```
printf("%d\n",j);
```

25. Mi lesz a kiírás eredménye?

```
int n5=18;
printf("W= %d\n", (n5&(1UL<<3)));
```

2.2. II. Minta vizsgafeladatsor

1. Számoljuk át tízes számrendszerbe az alábbi számokat:
   a. 110101.101_2
   b. $1D3.A5_{16}$
2. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat:
   a. $E8D6C_{16}=$
   b. $10011110011111010111_2=$
3. Írjuk át 10-es számrendszerbe:
   a. $2301.2,32,32,\ldots_4=$
4. Végezzük el az alábbi műveleteket a bináris számok körében:
   a. $1101.01_2 + 1101.11_2=$
   b. $10010.1100_2 - 11101.0011_2=$
5. Végezzük el az alábbi műveleteket a hexadecimális számok körében:
   a. $AF7B9.2A_{16} - C8B30.19_{16}=$
6. Írjuk le logikai kifejezéssel az alábbi áramkört. Mi a kifejezés értéke, ha A=B=D=1 (igaz)?



7. Melyik IEEE 754 számot ábrázoltuk?

   0 10010010 01100100000000000000000

8. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint: 1238.64_{10}
9. Ábrázoljuk nyolcas számrendszerben 117-at: előjelbittel kezdve, a kitevő legyen 1 jegyű (3 bites), 4-többletes, a törtrész 4 jegyű.
10. Ábrázoljuk 16-os számrendszerben –7253614.121-et: előjelbittel kezdve, a kitevő legyen 1 jegyű (4 bites), 8-többletes, a törtrész 4 jegyű.

11. Adjuk meg a –33 abszolútértékes, komplemens ábrázolásait, 1-es, 2-es és 127-többletes.

12. Írjon C programot a Fibonacci-számok rekurzív előállítására.

13. Bizonyítsa be indukcióval, hogy:

$$2\sum_{i=0}^{n} 3^i = 3^{n+1} - 1$$

14. Sorolja fel az első Generáció főbb jellemzőit.

15. Mit valósit meg az álabbi programrészlet? Írjuk át úgy, hogy ugyanezt valósítsa meg, de do-while ciklussal!

```
int x, k = 0;
scanf("%d", &x);
while(x)
{
 scanf("%d", &x);
 k++;
}
printf("%d", k);
```

16. Mi lesz a kiírás eredménye?

```
int A[50], i=0;
for ( ; i < 50; ++i )
 A[i] = 2*(i-1);
printf("%d\n %d \n", *(A+27), A[i-2]);
```

17. Írjunk makrót, mely visszaadja egy egész szám abszolút értékét.

18. Mi a kiíratás eredménye?

```
i=3; j=1; k=2;
k=++i-j++;
printf("%d %d %d\n", i,j,k);
```

19. Mi lesz az a, b és c változók értéke az egyes kifejezések kiértékelése után?

```
int a, b, c;
a = b = c = 9;
c = (a%3) * b++;
c = a < b ? a+2 : b-4;
b++; --a; c--;
```

20. Mennyi az a változó értéke az alábbi programrészlet végrehajtása után?

```
int a = 4;
int *p;
p=&a;
*p=2*a+*p-12;
printf("%d\n",a);
```

21. Mi lesz a k változó értéke az alábbi programrészlet végrehajtása után?

```
int i, j, k;
i=-2;j=-3;
if(i>=j && 4)
 k=-18;
else
 k=-2;
printf("%d\n",k);
}
```

22. Mi lesz az a változó értéke az alábbi programrészlet végrehajtása után?

```
int m=2;
char b=6;
double a,d=3.33;
a = (int)(b-m*d);
printf("%1.2f\n",a);
```

23. Mi a kiíratás eredménye?

```
double matrix [3][3]={9.8, 7.6, 6.5,4.3, 3.2, 2.1};
printf("%.2f , %f\n", matrix[1][1], matrix[2][1] );
```

24. Mi lesz a j változó értéke az alábbi programrészlet végrehajtása után?

```
int k=1, j=0;
int i=1;
 while (i<=102)
 j= (i++, k++);
printf("j=%d\n",j);
```

25. Mi lesz a kiírás eredménye?

```
int n5=18;
printf("E= %d \n", !!(n5&(1UL<<3)));
```

2.3. III. Minta vizsgafeladatsor

Figyelem! Az Excel feladatok megoldásához – paraméterezésükre vonatkozó megkötések miatt – nem használhatóak a darabteli, szumha, átlagha, szorzatösszeg, átlaghatöbb, darabhatöbb, stb., fkeres, vkeres és az adatbázis függvények!

	A	B	C	D	E	F	G	H	I	J	K	L
1	RN	SN	VN	NAME	LOCATION	STATUS	LATX	NS	LONGX	EW	ELEV	TYPE
128	02	02	-054	Korosi	Africa-E	Holocene	0,77	N	36,12	E	1446	Shield volcano
129	02	02	-055	Ol Kokwe	Africa-E	Holocene	0,62	N	36,08	E	1130	Shield volcano
130	02	02	-056	Nyambeni Hills	Africa-E	Holocene	0,23	N	37,87	E	750	Shield volcano
131	02	02	-06-	Menengai	Africa-E	Tephrochronolo	-0,20	S	36,07	E	2278	Shield volcano
132	02	02	-071	Elmenteita Badlands	Africa-E	Holocene	-0,52	S	36,27	E	2126	Pyroclastic cones

Megjegyzés: A táblázatban a 2. sortól kezdődően 1545 vulkán van felsorolva.

1. Ellenőrizze egyetlen összetett függvénnyel, hogy valamennyi északi féltekén lévő vulkán szélességi foka (G oszlop) pozitív szám-e vagy sem és írassa ki szövegesen a végeredmény!

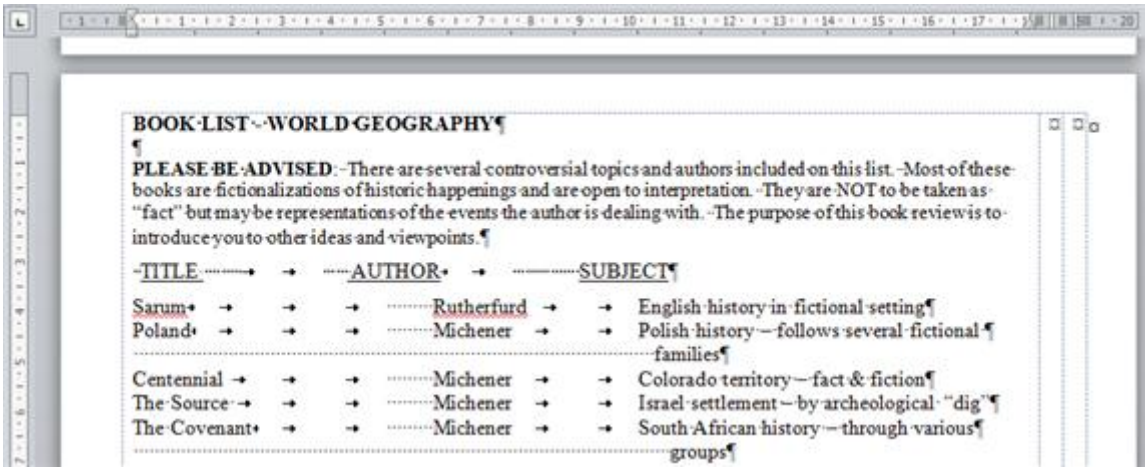
2. Adjon meg két eltérő helyszínt az M1 és az N1 cellákban! Írassa ki egyetlen összetett függvénnyel a két helyszínen (E oszlop) találhatók vulkánok számát!

3. Adjon meg két hosszúsági fokot (I oszlop) az M2 és N2 cellákban! Írassa ki a tartományba eső vulkánok átlagos magasságát!

4. Hasonlítsa össze az index és a hol.van függvényekből létrehozott összetett függvényt az fkeres és vkeres függvényekkel!

5. Adja meg a dokumentum fájl definícióját! Írja le a dokumentum fájlok és az adatfájlok kiterjesztése közötti összefüggéseket!

6. Mi a funkciója a következő billentyűknek Windows–Word környezetben: Insert, Home, PageUp, Delete



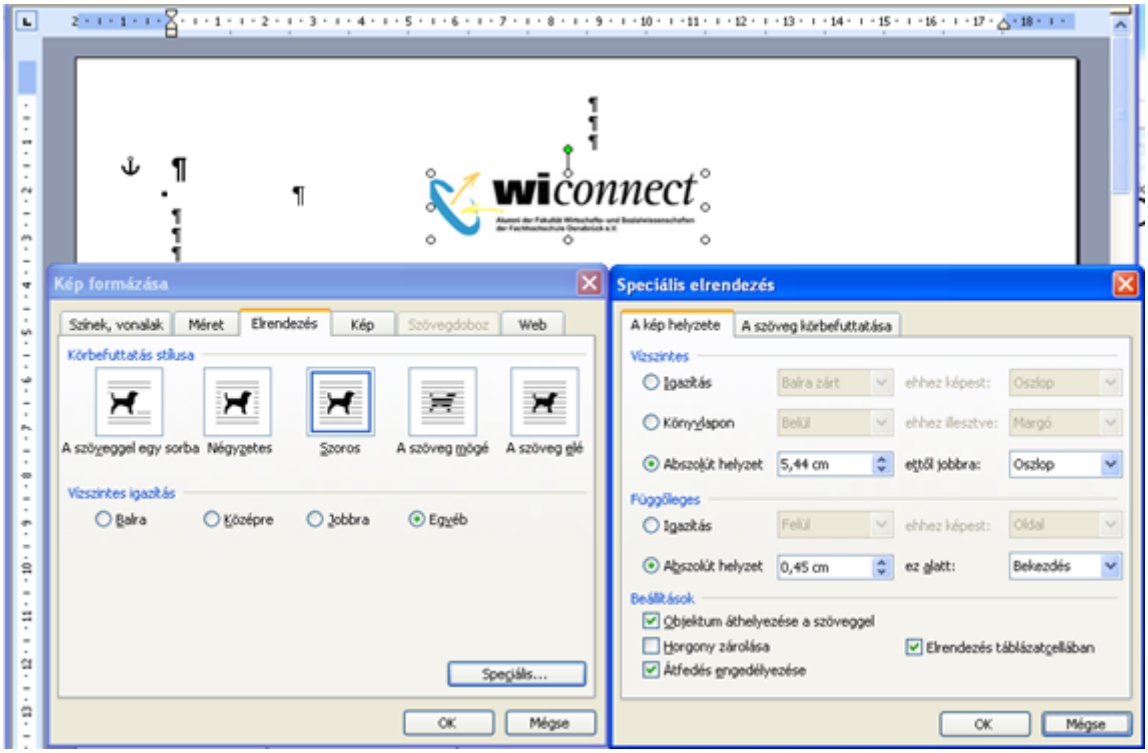
BOOK LIST - WORLD GEOGRAPHY

PLEASE BE ADVISED: There are several controversial topics and authors included on this list. Most of these books are fictionalizations of historic happenings and are open to interpretation. They are NOT to be taken as "fact" but may be representations of the events the author is dealing with. The purpose of this book review is to introduce you to other ideas and viewpoints.

TITLE	AUTHOR	SUBJECT
Sarum	Rutherfurd	English history in fictional setting
Poland	Michener	Polish history – follows several fictional families
Centennial	Michener	Colorado territory – fact & fiction
The Source	Michener	Israel settlement – by archeological "dig"
The Covenant	Michener	South African history – through various groups

7. Sorolja fel, milyen hibákat tartalmaz a fenti ábra szövegrészlete! A táblázat első oszlopában nevezze meg hibát, a másodikban adja meg a hiba típusát, a harmadikban pedig a helyes megoldást! Az ábrán jelölje a hibákat az ábécé betűivel!

Hiba	Hiba típusa	Helyes megoldás
a)		
b)		
c)		
d)		
e)		
f)		



8. Válaszoljon a következő kérdésekre a fenti ábra alapján! Indokolja válaszait!
   a. Helyes a kép elrendezése?
   b. Középre van igazítva a kép?
   c. Élőfejben van a kép?
   d. Melyik bekezdéshez tartozik a kép?
   e. Ezt az elrendezést használva lehetne a kép mellé szöveget írni?
9. Számoljuk át tízes számrendszerbe az alábbi számot (a törtrész maradjon közönséges tört alakban): 23612.352_9
10. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat!
    a. $B7E3DAC_{16}=$
    b. $10100101101111010110111_2=$
11. Írjuk át 10-es számrendszerbe (az eredményt adja meg közönséges tört alakban)!
    a. $31232.20'131'131'...5$
12. Végezzük el az alábbi műveleteket!
    a. $3467251_9+8276573_9=$
    b. $234653_7-3624025_7=$
    c. $A3075AD3_{16}-65ABCD74_{16}=$
    d. $110011011_2+101111101_2=$

13. Melyik IEEE 754 számot ábrázoltuk? A végeredményt decimális számrendszerben, egészrész és törtrész alakban adja meg!

11000110011101011011110000000000

14. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint! A végeredményt hexadecimális számrendszerben adja meg!

−3477.48=

15. Adja meg a −53 1-es komplemens, 2-es komplemens, 127-többletes és 128-többletes ábrázolásait 8 biten (a végeredményt hexadecimális számrendszerben adja meg)!

16. Mi lesz a kiírás eredménye? Indokolja meg válaszát!

```
int aaa (int n)
{
  return n>1 ? aaa(n-1)+n*(n-1) : n;
}
int main() {
    int s;
    s=aaa(5);
    printf("%d\n",s);
    return 0;
}
```

17. Mi lesz a kiírás eredménye?

```
int B[25],  i=0;
for ( ; i < 20; ++i )
    B[i] = 5*(2*i-1);
printf("%d\n%d\n", *(B+10), B[i-5]);
```

18. Mit csinál az alábbi programrészlet?

```
srand(time(NULL));
    int i, b, n, a[10];
    for (i=0; i<n; i++){
        do{
            b=rand()%3-1;
        }while(!b);
        a[i]=b;
        printf("%d\t",a[i]);
    }
    printf("\n");
```

19. Mit csinál az alábbi programrészlet és mi a hibája? Indokolja válaszát és javítsa ki a hibát!

```
int main(){
    float h=0.01;
    printf("%f\t",bbb(h));
    return 0;
}
float bbb(float e){
    float s=1; int i=2, k;
    do{
        k= i%2 ? i : -i;
        i++;
        s+=(float)1/k;
    }while(1/i>e);
    return s;
```

```
}
```

20. Írjon programot, amely bekéri egy forgáshenger magasságát és alapkörének sugarát, majd meghív egy függvényt, amely kiszámítja a henger felszínét. A főprogram írassa ki a bekért értékeket, majd a henger felszínét.

21. Adja meg a BMP sík megadott karakterének Unicode értékét és UTF-8 ábrázolását tizenhatos számrendszerben!

	00	01	02	03	04	05	06	07	08	09	0A	0B	0C	0D	0E	0F	10	11	12	13	14	15	16	17	18	19	1A	1B	1C	1D	1E	1F
A000	ꀀ	ꀁ	ꀂ	ꀃ	ꀄ	ꀅ	ꀆ	ꀇ	ꀈ	ꀉ	ꀊ	ꀋ	ꀌ	ꀍ	ꀎ	ꀏ	ꀐ	ꀑ	ꀒ	ꀓ	ꀔ	ꀕ	ꀖ	ꀗ	ꀘ	ꀙ	ꀚ	ꀛ	ꀜ	ꀝ	ꀞ	ꀟ
A020	ꀠ	ꀡ	ꀢ	ꀣ	ꀤ	ꀥ	ꀦ	ꀧ	ꀨ	ꀩ	ꀪ	ꀫ	ꀬ	ꀭ	ꀮ	ꀯ	ꀰ	ꀱ	ꀲ	ꀳ	ꀴ	ꀵ	ꀶ	ꀷ	ꀸ	ꀹ	ꀺ	ꀻ	ꀼ	ꀽ	ꀾ	ꀿ
A040	ꁀ	ꁁ	ꁂ	ꁃ	ꁄ	ꁅ	ꁆ	ꁇ	ꁈ	ꁉ	ꁊ	ꁋ	ꁌ	ꁍ	ꁎ	ꁏ	ꁐ	ꁑ	ꁒ	ꁓ	ꁔ	ꁕ	ꁖ	ꁗ	ꁘ	ꁙ	ꁚ	ꁛ	ꁜ	ꁝ	ꁞ	ꁟ
A060	ꁠ	ꁡ	ꁢ	ꁣ	ꁤ	ꁥ	ꁦ	ꁧ	ꁨ	ꁩ	ꁪ	ꁫ	ꁬ	ꁭ	ꁮ	ꁯ	ꁰ	ꁱ	ꁲ	ꁳ	ꁴ	ꁵ	ꁶ	ꁷ	ꁸ	ꁹ	ꁺ	ꁻ	ꁼ	ꁽ	ꁾ	ꁿ
A080	ꂀ	ꂁ	ꂂ	ꂃ	ꂄ	ꂅ	ꂆ	ꂇ	ꂈ	ꂉ	ꂊ	ꂋ	ꂌ	ꂍ	ꂎ	ꂏ	ꂐ	ꂑ	ꂒ	ꂓ	ꂔ	ꂕ	ꂖ	ꂗ	ꂘ	ꂙ	ꂚ	ꂛ	ꂜ	ꂝ	ꂞ	ꂟ
A0A0	ꂠ	ꂡ	ꂢ	ꂣ	ꂤ	ꂥ	ꂦ	ꂧ	ꂨ	ꂩ	ꂪ	ꂫ	ꂬ	ꂭ	ꂮ	ꂯ	ꂰ	ꂱ	ꂲ	ꂳ	ꂴ	ꂵ	ꂶ	ꂷ	ꂸ	ꂹ	ꂺ	ꂻ	ꂼ	ꂽ	ꂾ	ꂿ
A0C0	ꃀ	ꃁ	ꃂ	ꃃ	ꃄ	ꃅ	ꃆ	ꃇ	ꃈ	ꃉ	ꃊ	ꃋ	ꃌ	ꃍ	ꃎ	ꃏ	ꃐ	ꃑ	ꃒ	ꃓ	ꃔ	ꃕ	ꃖ	ꃗ	ꃘ	ꃙ	ꃚ	ꃛ	ꃜ	ꃝ	ꃞ	ꃟ

2.4. IV. Minta vizsgafeladatsor

Figyelem! Az Excel feladatok megoldásához – paraméterezésükre vonatkozó megkötések miatt – nem használhatóak a darabteli, szumha, átlagha, szorzatösszeg, átlaghatöbb, darabhatöbb, stb., fkeres, vkeres és az adatbázis függvények!

	A	B	C	D	E	F
1	NAME	RANK	AGE	BIRTHPLACE	SERVED WITH	INCIDENT
2	Limbu, Sachin	Rifleman	23	Nepal#Rajghat, Morang in Nepal	Army	Hostile: Explosion
3	King, John	Pte	19	County Durham#Darlington	Army	Hostile: Explosion
4	Downing, Anthony	Sqn Ldr	34		RAF	Hostile: Explosion
5	Jennings, Tom	Capt	29		Royal Marines	Hostile: Explosion
6	Bond, Elijah	Sapper	24	Hampshire#Havant	Army	Hostile: Explosion
7	Steel, Sheldon	Rifleman	20	West Yorkshire#Leeds	Army	Hostile: Explosion
8	Lake, Thomas	Pte	29	Hertfordshire#Watford	Army	Hostile: Explosion
9	Boyce, David	Lt	25	Hertfordshire#Welwyn Garden City	Army	Hostile: Explosion
10	Scanlon, Richard	L/Cpl	31	Caerphilly#Rhymney	Army	Hostile: Explosion
11	Eustace, Peter	L/Cpl	25	Merseyside#Liverpool	Army	Hostile: Explosion
12	Thornton, Matthew	Pte	28	South Yorkshire#Barnsley	Army	Hostile: Explosion
13	Haseldin, Matthew	Pte	21	Yorkshire#Settle	Army	Hostile: Shot
14	Rai, Vijay	Rifleman	21	Nepal	Army	Hostile: Shot
15	Fairbrother, David	Marine	24	Lancashire#Blackburn	Royal Marines	Hostile: Shot
16	McKinlay, Jonathan	L/Cpl	33	Germany	Army	Hostile: Shot
17	Weston, Barry	Sgt	40	Berkshire#Reading	Royal Marines	Hostile: Explosion

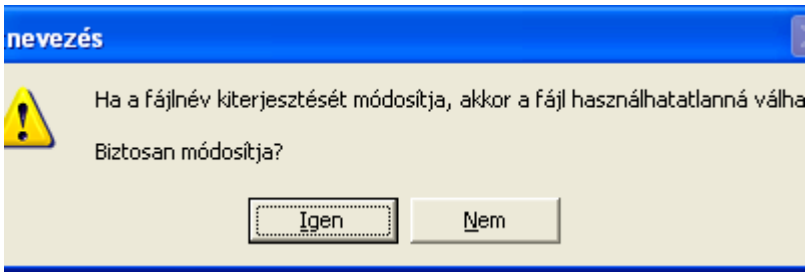
Megjegyzés: Az utolsó rekord a munkalap 325. sorában található.

1. Írjon összetett képletet, amely visszaadja a születési hely (D oszlop) első felét. Abban az esetben is működjön a képlet, ha nincs megadva születési hely, illetve akkor is ha csak egy részből áll. A két részt a # választja el egymástól.

2. Adjon meg két eltérő baleset típust (F oszlop) a G1 és a G2 cellákban! Írassa ki egyetlen összetett függvénnyel azon katonák számát, akik a két típus valamelyikében vesztették életüket!

3. Adjon meg egy életkort (C oszlop) a H1 cellában és egy egységet (E oszlop) a H2 cellában! Írassa ki azoknak a katonáknak az átlagéletkorát, akik idősebbek, mint a H1-ben megadott életkor és a H2-ben megadott egységnél szolgáltak!

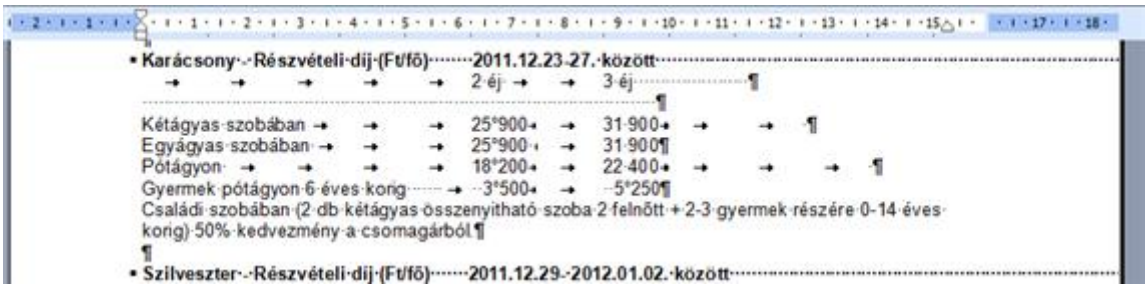
4. Milyen műveletnél jelenik meg és hogyan magyarázná az alábbi üzenetet?



5. Bizonyítsa be teljes indukcióval, hogy

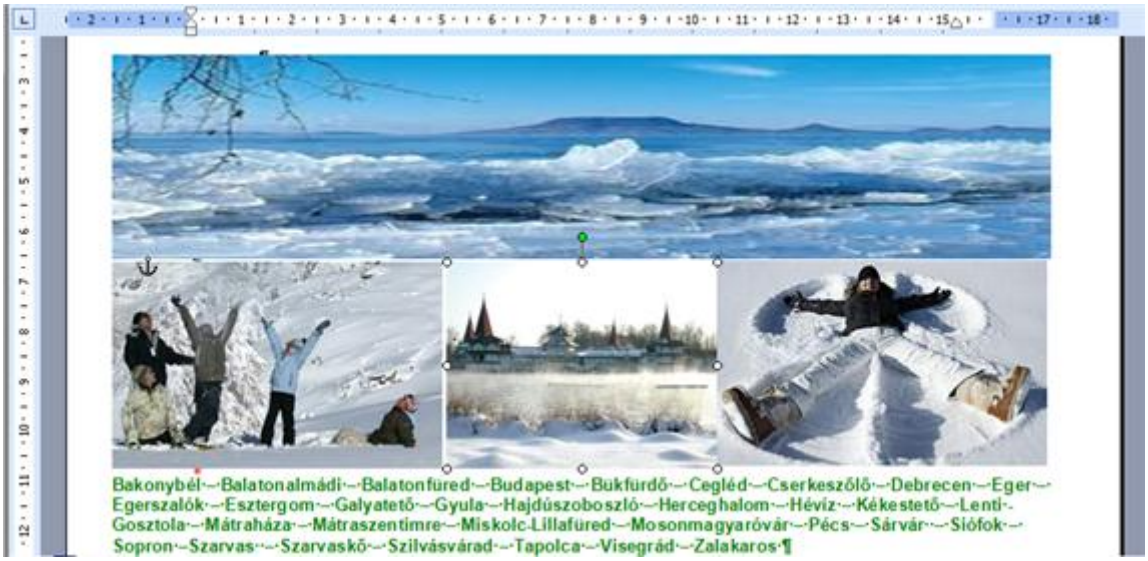
$$\sum_{i=1}^{n} i(3i+1) = n(n+1)^2$$

6. Rajzolja fel a for ciklus blokkdiagramját!



7. Sorolja fel, milyen hibákat tartalmaz a fenti ábra szövegrészlete! A táblázat első oszlopában nevezze meg hibát, a másodikban adja meg a hiba típusát, a harmadikban pedig a helyes megoldást! Az ábrán jelölje a hibákat az ábécé betűivel!

Hiba	Hiba típusa	Helyes megoldás
a)		
b)		
c)		
d)		
e)		
f)		



8. Válaszoljon a következő kérdésekre a fenti ábra alapján! Indokolja válaszait!
   a. Helyes a kijelölt kép elrendezése?
   b. Tartalmaz a szövegrészlet fölösleges entereket?
   c. Milyen bekezdésre vonatkozó formázási hibát tartalmaz a szövegrészlet? Honnan lehet ezt leolvasni?
   d. Melyik bekezdéshez tartozik a kép?
   e. Nem a kép elrendezésével kapcsolatos egyéb, a képhez köthető hiba található a szövegrészleten?
9. Számoljuk át tízes számrendszerbe az alábbi számot (a törtrész maradjon közönséges tört alakban):

   $26152.253_{7}=$

10. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat!
   a. $C7D38AB_{16}=$
   b. $10101101100101110110111_{2}=$
11. Írjuk át tízes számrendszerbe (az eredményt adja meg közönséges tört alakban)!
   a. $12673.42'57'57'\ldots_{9}=$
12. Végezzük el az alábbi műveleteket!
   a. $535675_{8} + 726432_{2}=$
   b. $5220364_{7} - 541216_{7}=$
   c. $A7B6CD2_{16} + 6E43ADF5_{16}=$
   d. $1011100101_{2} - 1101010011_{2}=$
13. Melyik IEEE 754 számot ábrázoltuk? A végeredményt decimális számrendszerben, egészrész, törtrész alakban adja meg!

   11000111001010100111111111000000
14. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint! A végeredményt hexadecimális számrendszerben adja meg!

356748.44_{10}

15. Adja meg a −76 1-es komplemens, 2-es komplemens, 127-többletes és 128-többletes ábrázolásait 8 biten (a végeredményt hexadecimális számrendszerben adja meg)!

16. Mi lesz a kiírás eredménye? Indokolja meg válaszát!

```
int vizsga2(int n);
int main(){
    int tgv; scanf("%d",tgv);
    printf("%d\n",vizsga2(tgv));
    return 0;
}
int vizsga2(int n){
    int i,j,k=0, a[100]; scanf("%d",&i);
    do{
        a[k]=i%n;
        k++;
        i/=n;
        printf(%d\t %d\n",i,a[k]);
    }while(i);
    return k;
}
```

a. Mit csinál a program?

b. Mi a tgv változó szerepe?

c. Mit ír ki a program?

d. Két súlyos hiba is van a programban. Melyek ezek?

17. Mi lesz a kiírás eredménye?

```
void aaa(int n);
int main(){
    aaa(10);
    return 0;
}
void aaa(int n){
    int a[100], i=1;
    for (i;i<=n; i++)
        a[i]=i*(3*i+1);
    printf("i=%d\ts=%d\t%d\t%d\n",i,
a[i-2],*a,*(a+i));
}
```

18. Mi a kiíratás eredménye? Indokolja válaszát!

```
int c1=72, c2=98;
     c1^=c2;
     c2^=c1;
     c1^=c2;
printf("c1= %X\tc2= %X",c1,c2);
```

19. Mi a kiíratás eredménye? Indokolja válaszát!

```
void kiir2(){
    int i, j, k;
    i=-2; j=-3; k=0;
    if(i= j || k) k=-1 * (++i || j);
    else k=-2 * (i && k);
```

```
    printf("%d\t%d\n",i,k);
}
```

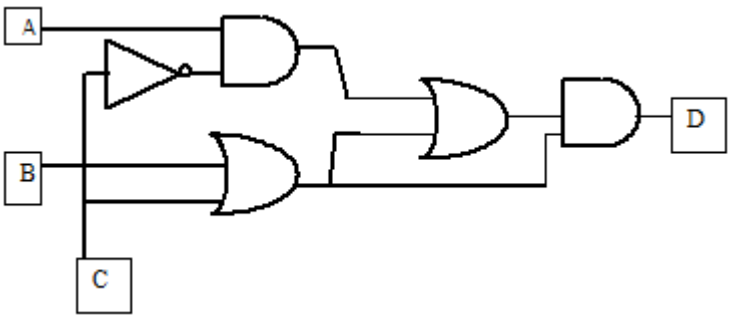
20. Írjon programot, amely bekéri egy forgáskúp magasságát és alapkörének sugarát, majd meghív egy függvényt, amely kiszámítja a kúp térfogatát. A főprogram írassa ki a bekért értékeket, majd a kúp térfogatát.

21. Adja meg a BMP sík megadott karakterének Unicode értékét és UTF-8 ábrázolását tizenhatos számrendszerben!

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x3040		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	お	お	か	が	き	ぎ	く
0x3050	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
0x3060	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
0x3070	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
0x3080	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
0x3090	ゐ	ゑ	を	ん	ゔ					゛	゜	ゝ	ゞ	ゟ		

2.5. I. ZH1 Mintafeladatsor

1. Számoljuk át tízes számrendszerbe az alábbi számokat:
   a. 72.14_8=
   b. 101101.11_2=
   c. 122.64_{16}=
2. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat:
   a. $3BDA_{16}$=
   b. $1110\ 0111\ 1101\ 0111_2$=
   c. $1\ 0111\ 0000\ 1101_2$=
3. Írjuk át 10-es számrendszerbe:
   a. $1320.2,32,32,\ldots_4$=
   b. $4.20,4,4,\ldots_8$=
4. Írjuk le logikai kifejezéssel az alábbi áramkört. Mi a kifejezés értéke, ha A=B=D=1 (igaz)?



5. Végezzük el az alábbi műveleteket a bináris számok körében:
   a. $1011.01_2 + 1101.11_2 =$
   b. $10111.01_2 + 11011.11_2 =$
   c. $1010.10_2 - 1001.01_2 =$
   d. $10000.1010_2 - 11101.1111_2 =$
6. Végezzük el az alábbi műveleteket a hexadecimális számok körében:
   a. $1B0A.0F0_{16} + A111.013_{16} =$
   b. $AA729.2A_{16} - C8B3C.25_{16} =$
7. Melyik IEEE 754 számot ábrázoltuk?

   0 10010011 00110000000000000000000

   1 10001001 11100000000000000000000
8. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint:

   $1401.24_{10} =$

   $-0.08232_{10} =$
9. Ábrázoljuk nyolcas számrendszerben 148_{10}-at: előjelbittel kezdve, a kitevő legyen 1 jegyű (3 bites), 4-többletes, a törtrész 4 jegyű.
10. Ábrázoljuk 16-os számrendszerben -603710.12_{10}-et: előjelbittel kezdve, a kitevő legyen 1 jegyű (4 bites), 8-többletes, a törtrész 4 jegyű.
11. Adjuk meg a 18 abszolútértékes, komplemens ábrázolásait, 1-es, 2-es és 127-többletes.
12. Adjuk meg a −24 abszolútértékes, komplemens ábrázolásait, 1-es, 2-es és 127-többletes.

2.6. II. ZH1 Mintafeladatsor

1. Számoljuk át tízes számrendszerbe az alábbi számokat:
   a. $73.04_8 =$
   b. $10101.01_2 =$
   c. $132.54_{16} =$
2. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat:

a. $1ADF_{16}$=

b. $1110\ 0001\ 1111\ 0111_2$=

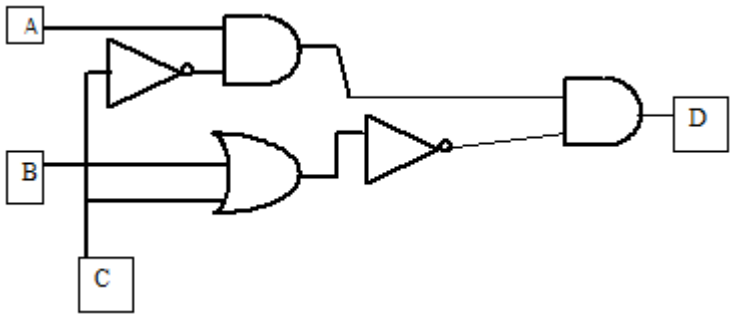
c. $1\ 0101\ 0100\ 1101_2$=

3. Írjuk át 10-es számrendszerbe:

a. $1320.20,131,131,\ldots_4$=

b. $4.24,2,2,\ldots_8$=

4. Írjuk le logikai kifejezéssel az alábbi áramkört. Mi a kifejezés értéke, ha A=B=D=1 (hamis)?



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

a. $1001.01_2 + 1101.11_2$=

b. $10111.01_2 + 1011.11_2$=

c. $1011.10_2 - 1001.01_2$=

d. $11001.1110_2 - 11001.1111_2$=

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

a. $190A.010_{16} + A111.013_{16}$=

b. $1A729.26_{16} - C8B3C.25_{16}$=

7. Melyik IEEE 754 számot ábrázoltuk?

0 10000011 01010000000000000000000

1 10000101 10100000000000000000000

8. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint:

1201.28_{10}=

-0.001232_{10}=

9. Ábrázoljuk nyolcas számrendszerben 124_{10}-at: előjelbittel kezdve, a kitevő legyen 1 jegyű (3 bites), 4-többletes, a törtrész 4 jegyű.

10. Ábrázoljuk 16-os számrendszerben -703710.11_{10}-et: előjelbittel kezdve, a kitevő legyen 1 jegyű (4 bites), 8-többletes, a törtrész 4 jegyű.

11. Adjuk meg a 16 abszolútértékes, komplemens ábrázolásait, 1-es, 2-es és 127-többletes.

12. Adjuk meg a −32 abszolútértékes, komplemens ábrázolásait, 1-es, 2-es és 127-többletes.

2.7. III. ZH1 Mintafeladatsor

1. Adja meg a dokumentum fájl definícióját! Írja le a dokumentum fájlok alapértelmezett használati módját!

2. Mi a funkciója a következő billentyűknek Windows–Word környezetben: Insert, Home, PageUp, PageDown, End, Delete.

3. Bizonyítsa be, hogy a végtelen mértani sorok összegét felhasználva hogyan tudunk szakaszos végtelen p-ados törteket más számrendszerbe átszámolni.

4. Számoljuk át tízes számrendszerbe az alábbi számot (a törtrész maradjon közönséges tört alakban):

   a. 23615.643_{7}=

   b. 24564.321_{8}=

5. Végezzük el az alábbi műveleteket!

   a. $8017345_{9} + 462375_{9}$=

   b. $20E1A57_{16} + CE88F79_{16}$=

   c. $A275B4_{16} - 4CD378_{16}$=

   d. $100110101_{2} - 111010011_{2}$=

6. Számoljuk át az alábbi tízes számrendszerbeli számot hetes számrendszerbe.

   23615.45_{10}=

7. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat!

   a. $BE94A5_{16}$=

   b. $1001110101010011011011_{2}$=

8. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat!

   a. 532674_{8}=

   b. 532674_{2}=

9. Ábrázoljuk oktális számrendszerben a −95.23 számot: előjelbittel kezdve, a kitevő legyen 1 jegyű (3 bites), 4-többletes, a törtrész 4 jegyű.

10. Ábrázoljuk hexadecimális számrendszerben a −34567.67 számot: előjelbittel kezdve, a kitevő legyen 1 jegyű (4 bites), 8-többletes, a törtrész 4 jegyű.

11. Írjuk át tízes számrendszerbe (az eredményt adja meg közönséges tört alakban)!

    $2132.23'231'231'\ldots_{4}$

12. Melyik IEEE 754 számot ábrázoltuk? A végeredményt decimális számrendszerben, egészrész, törtrész alakban adja meg!

    01000110010110110101000000000000

13. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint! A végeredményt hexadecimális számrendszerben adja meg!

    34967.133

14. Adja meg a −104 1-es komplemens, 2-es komplemens, 127-többletes és 128-többletes ábrázolásait 8 biten (a végeredményt hexadecimális számrendszerben adja meg)!

15. Sorolja fel, milyen hibákat tartalmaz az alábbi szövegrészlet! A táblázat első oszlopában nevezze meg hibát, a másodikban adja meg a hiba típusát, a harmadikban pedig a helyes megoldást! Az ábrán jelölje a hibákat az ábécé betűivel!

T·Á·J·É·K·O·Z·T·A·T·Ó¶

z ankét¶

•→Ideje·2007.·június·25-28.¶

•→Helye·Vásárosnamény¶

•→Részvételi·díj·a·rendezvény·teljes·idejére·(4·napra):¶

→ → → → → ·····ELFT·tagoknak → ·····Nem·tagoknak¶

lás·kollégiumban[1]/panzióban·reggelivel:·····8.800·Ft/·18.000·Ft → ··8.800·Ft/·18.000·Ft
zés·kollégiumban[2]/panzióban····· → → 6.200·Ft/···4.600·Ft → ··6.200·Ft/···4.600·Ft
mai·kirándulás[3]·Tisza-parti·vacsorával → 2.000·Ft/···2.000·Ft·····2.000·Ft/···2.000·Ft
adás → → → → → 2.000·Ft/···2.000·Ft·····2.000·Ft/···2.000·Ft
vezési·költség·(minden·részvevőnek): → ·····6.000·Ft/···6.000·Ft·····11.000·Ft/·11.000·F

zes·költség·(kollégium/panzió): ····· 25.000·Ft/·32.600·Ft → ·30.000·Ft/·37.600·F

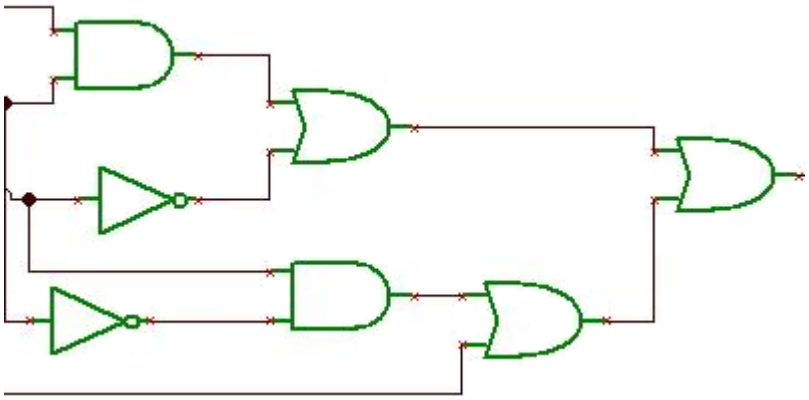
elválasztóval·elkülönített·2x2·ágyas·szobák,·közös·zuhanyozó,·mosdó,·WC·a·folyosóról·nyílik.·Az·épül
yarország·pont:·állandó·Internet·hozzáférési·lehetőség·van.·[2]Bontásban:·reggeli:·400·Ft,·ebéd:·800·Ft,·vac
Ft.·[3]Program:·Tákos,·Csaroda,·Tarpa,·Nagyar,·Szatmárcseke,·Túristvándi,·Fehérgyarmat.·Részletes·informá
orolt·helyiségekről·a·www.szilvasut.hu·és·a·www.vasarosnameny.hu·internetes·oldalakon·olvashatók.·¶

állás·helyei:·-·Babus·Jolán·Középiskolai·Kollégium·4800·Vásárosnamény,·Kossuth·út
- Hotel·Fehér·4800·Vásárosnamény,·Bereg·köz·1.·(www.hotels.hu/feherhot
tkezés·helye:·Babus·Jolán·Középiskolai·Kollégium·4800·Vásárosnamény,·Kossuth·út
loádások,·a·műhelyfoglalkozások·és·az·eszközkiállítás·helye:·Lónyai·Menyhért·Szakköz
és·Szakképző·Iskola,·4800·Vásárosnamény,·Kossuth·út·19.·¶
gisztráció·helye:·Babus·Jolán·Középiskolai·Kollégium·4800·Vásárosnamény,·Kossutl
21.¶

atáridők,·tudnivalók¶

Hiba	Hiba típusa	Helyes megoldás
a)		
b)		
c)		
d)		
e)		
f)		
g)		
h)		
i)		
j)		

16. Írja le kétféle jelölésmódot használva az alábbi áramkört! Mi a kifejezés értéke, ha A=0, B=1, D=0?



17. Adja meg a BMP sík megadott karakterének Unicode értékét és UTF-8 ábrázolását tizenhatos számrendszerben!

ゆ

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x3040		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	お	お	か	が	き	ぎ	く
0x3050	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
0x3060	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
0x3070	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
0x3080	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
0x3090	ゐ	ゑ	を	ん	ゔ					゛	゜	ゝ	ゞ	ヽ	ヾ	

18. Rajzolja fel az AND halmazelméleti megfelelőjét!

2.8. IV. ZH1 Mintafeladatsor

1. Adja meg a helyesen formázott szöveg definícióját, illetve az ehhez kapcsolódó szabályokat!
2. Magyarázza el a numerikus billentyűzet használatát!
3. Mutassa meg egy kettes számrendszerbeli szám és ellentettje közötti összefüggést!
4. Számoljuk át tízes számrendszerbe az alábbi számot (a törtrész maradjon közönséges tört alakban):
   a. 4156.432_{7}=
   b. 34164.526_{8}=
5. Végezzük el az alábbi műveleteket!
   a. $6401241_{7} + 2020232_{7}$=
   b. $5EDD409_{16} + C9BEDA4_{16}$=
   c. $C72BA9_{16} - 4D7AE58_{16}$=

d. $1101101111_2 - 10011011000_2 =$

6. Számoljuk át az alábbi tízes számrendszerbeli számot hetes számrendszerbe.

   $56237.85_{10} =$

7. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat!

   a. $9A3D7C_{16} =$

   b. $11100011101011101011012 =$

8. Írjuk át kettes számrendszerbe a tizenhatos számrendszerbeli, illetve tizenhatos számrendszerbe a kettes számrendszerbeli számokat!

   a. $752346_8 =$

   b. $11001100010110101111_2 =$

9. Ábrázoljuk oktális számrendszerben a −62.32 számot: előjelbittel kezdve, a kitevő legyen 1 jegyű (3 bites), 4-többletes, a törtrész 4 jegyű.

10. Ábrázoljuk hexadecimális számrendszerben a 122444.654 számot: előjelbittel kezdve, a kitevő legyen 1 jegyű (4 bites), 8-többletes, a törtrész 4 jegyű.

11. Írjuk át tízes számrendszerbe (az eredményt adja meg közönséges tört alakban)!

    $1334.14'134'134'\ldots_5$

12. Melyik IEEE 754 számot ábrázoltuk? A végeredményt decimális számrendszerben, egészrész, törtrész alakban adja meg!

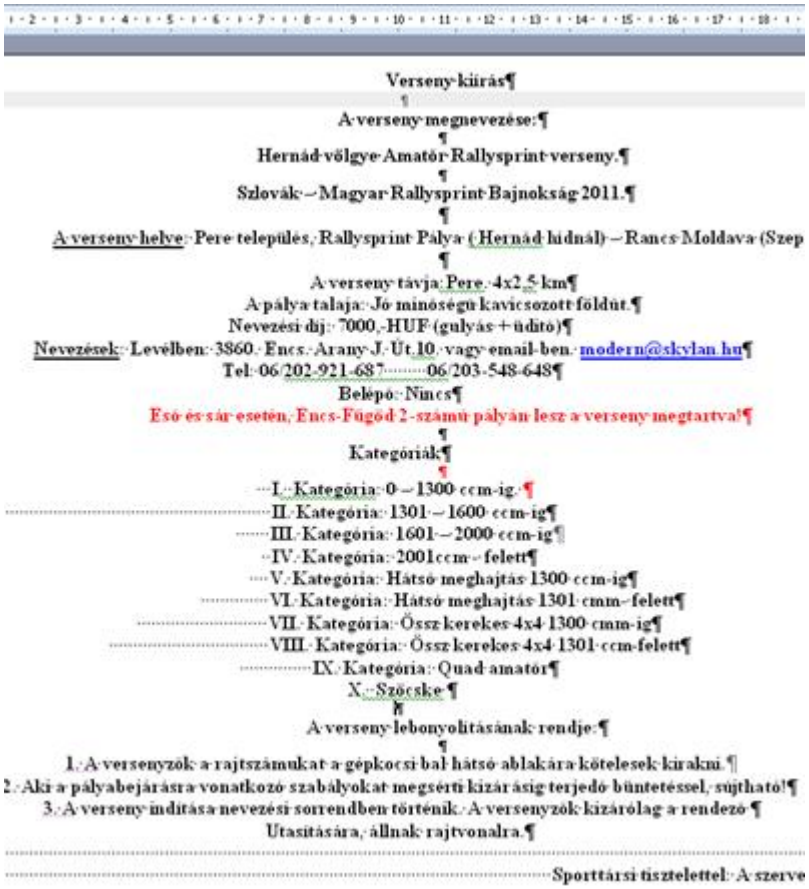
    01001000011010011011000000000000

13. Ábrázoljuk IEEE 754 lebegőpontos szabvány szerint! A végeredményt hexadecimális számrendszerben adja meg!

    −1234567.875

14. Adja meg a 94 1-es komplemens, 2-es komplemens, 127-többletes és 128-többletes ábrázolásait 8 biten (a végeredményt hexadecimális számrendszerben adja meg)!

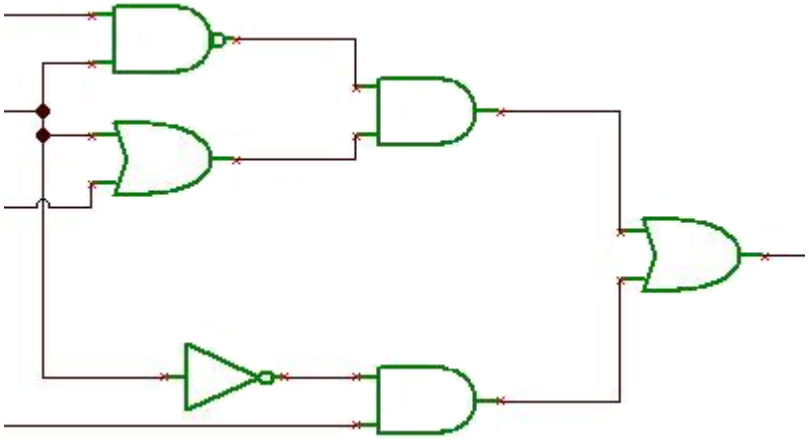
15. Sorolja fel, milyen hibákat tartalmaz az alábbi szövegrészlet! A táblázat első oszlopában nevezze meg hibát, a másodikban adja meg a hiba típusát, a harmadikban pedig a helyes megoldást! Az ábrán jelölje a hibákat az ábécé betűivel!



Verseny·kiírás¶
¶
A·verseny·megnevezése:¶
¶
Hernád·völgye·Amatőr·Rallysprint·verseny.¶
¶
Szlovák·–·Magyar·Rallysprint·Bajnokság·2011.¶
¶
A·verseny·helye:·Pere·település,·Rallysprint·Pálya·(Hernád·hídnál)·–·Rancs·Moldava·(Szep
¶
A·verseny·távja:·Pere,·4x2,5·km¶
A·pálya·talaja:·Jó·minőségű·kavicsozott·földút.¶
Nevezési·díj:·7000,-HUF·(gulyás·+·üdítő)¶
Nevezések:·Levélben:·3860.·Encs,·Arany·J.·Út.10.·vagy·email-ben.·modern@skylan.hu¶
Tel:·06/202-921-687·····06/203-548-648¶
Belépő:·Nincs¶
Eső·és·sár·esetén,·Encs-Fügöd·2-számú·pályán·lesz·a·verseny·megtartva!¶
¶
Kategóriák:¶
¶
···I.·Kategória:·0·–·1300·ccm-ig.·¶
·····II.·Kategória:·1301·–·1600·ccm-ig¶
·····III.·Kategória:·1601·–·2000·ccm-ig¶
···IV.·Kategória:·2001ccm·–·felett¶
····V.·Kategória:·Hátsó·meghajtás·1300·ccm-ig¶
·····VI.·Kategória:·Hátsó·meghajtás·1301·cmm-felett¶
·····VII.·Kategória:·Össz·kerekes·4x4·1300·cmm-ig¶
·····VIII.·Kategória:·Össz·kerekes·4x4·1301·ccm-felett¶
·····IX.·Kategória:·Quad·amatőr¶
X.·Szöcske·¶
¶
A·verseny·lebonyolításának·rendje:¶
¶
1.·A·versenyzők·a·rajtszámukat·a·gépkocsi·bal·hátsó·ablakára·kötelesek·kirakni.¶
2.·Aki·a·pályabejárásra·vonatkozó·szabályokat·megsérti·kizárásig·terjedő·büntetéssel,·sújtható!¶
3.·A·verseny·indítása·nevezési·sorrendben·történik.·A·versenyzők·kizárólag·a·rendező·¶
Utasítására,·állnak·rajtvonalra.¶
·····
·····Sporttársi·tisztelettel:·A·szerve

Hiba	Hiba típusa	Helyes megoldás
a)		
b)		
c)		
d)		
e)		
f)		
g)		
h)		
i)		
j)		

16. Írja le kétféle jelölésmódot használva az alábbi áramkört! Mi a kifejezés értéke, ha A=0, B=1, D=0?



17. Adja meg a BMP sík megadott karakterének Unicode értékét és UTF-8 ábrázolását tizenhatos számrendszerben!

	0x00	0x01	0x02	0x03	0x04	0x05	0x06	0x07	0x08	0x09	0x0A	0x0B	0x0C	0x0D	0x0E	0x0F
0x2d30	ⴰ	ⴱ	ⴲ	ⴳ	ⴴ	ⴵ	ⴶ	ⴷ	ⴸ	ⴹ	ⴺ	ⴻ	ⴼ	ⴽ	ⴾ	ⴿ
0x2d40	ⵀ	ⵁ	ⵂ	ⵃ	ⵄ	ⵅ	ⵆ	ⵇ	ⵈ	ⵉ	ⵊ	ⵋ	ⵌ	ⵍ	ⵎ	ⵏ
0x2d50	ⵐ	ⵑ	ⵒ	ⵓ	ⵔ	ⵕ	ⵖ	ⵗ	ⵘ	ⵙ	ⵚ	ⵛ	ⵜ	ⵝ	ⵞ	ⵟ
0x2d60	ⵠ	ⵡ	ⵢ	ⵣ	ⵤ	ⵥ										ⵯ
0x2d70																⵿

18. Rajzolja fel az XOR halmazelméleti megfelelőjét!

2.9. I. ZH2 Mintafeladatsor

1. Mi lesz a kiírás eredménye?

```
double a=23.23, b=32.32;
     a-=b;
     b+=a;
     a=b-a;
printf("\na= %3.1lf \nb= %f",a,b);
```

2. Melyek a végtelen ciklusok?

   a. i = −1; while(!i);

   b. int i=5; while(i == 10) i=10;

   c. for(i = 1; i = !10; i−−);

   d. for(x = 10; x >= 10; x++);

3. Mi lesz a kiírás eredménye?

```
i=3; j=1; k=2;
k+=-i+++j;
```

```
printf("%d  %d  %d\n%", i,j,k);
```

4. Mi lesz a kiírás eredménye?

```
i=3; j=1; k=2;
k=+i+-j;
printf("%d  %d  %d\n%", i,j,k);
```

5. Mi lesz a kiírás eredménye?

```
i=3; j=1; k=2;
k=++i-j++;
printf("%d  %d  %d\n%", i,j,k);
```

6. Mi lesz az a, b és c változók értéke az egyes kifejezések kiértékelése során?

```
int a, b, c;
a = b = c = 7;
c = a++ * (b%3);
c = a < b ? a+2 : b-3;
b--; a++;
```

7. Mennyi az a változó értéke az alábbi programrészlet végrehajtása során?

```
int a = 4;
int *p;
p=&a;
*p=a+*p+2;
printf("%d\n",a);
```

8. Mennyi az x változó értéke az alábbi programrészlet végrehajtása során?

```
int x = 5;
int *p, **q;
p=&x;
q=&p;
x + =*p + **q;
printf("%d\n",x);
```

9. Mennyi az s változó értéke az alábbi programrészlet végrehajtása során?

```
double a1=1.0, b1=4.0;
double s;
s = a1 + 1/b1 * 3 + 64;
printf("%o\n",(char)s);
printf("%d\n",(char)s);
```

10. Mi lesz az k változó értéke az alábbi programrészlet végrehajtása során?

```
int i, j, k;
i=4;j=3;
if(i>=j && 0)
 k=1;
else
 k=3;
```

```
printf("%d\n",k);
```

11. Mi lesz a kiírás eredménye?

```
char q=5;
int m=2;
double dd,d=1.30;
dd = (int)(q+m*d);
printf("%lf\n",dd);
```

12. Mi lesz a kiírás eredménye?

```
int ax,b;
ax=0x057;
b=0x34;
ax= ax | b;
printf("b=%x, ax=%d\n",b, ax);
```

13. Mi lesz az x változó értéke az alábbi programrészlet végrehajtása során?

```
int x, y;
x = (y = 1 , y + 2);
```

14. Mi lesz az i, j és k változók értéke az egyes kifejezések kiértékelése során?

```
i=1; j=3; k=2;
k = --i-j--;
```

15. Mi lesz a j változó értéke az alábbi programrészlet végrehajtása során?

```
int k=0, j=0;
int i;
for(i=1;i<=4;i++)
 j=k++;
printf("%d\n",j);
```

16. Mi lesz a kiírás eredménye?

```
int ax, a;
ax=0x058;
a=0x033;
ax ^= (1UL << 3);
a &= (1UL << 5);
printf(" ax=%d\n   a=%d\n", ax, a);
```

2.10. II. ZH2 Mintafeladatsor

1. Mi lesz a kiírás eredménye?

```
double a=32.23,b=23.32;
double tmp;
 tmp=a;
 a=b;
 b=tmp;
```

```
printf("\na= %lf \nb= %5.1lf",a,b);
```

2. Melyek a végtelen ciklusok?

   a. i = 1; while(!i);

   b. int i=5; while(i == 5 | i==10) i=10;

   c. for(x = 1;; x++);

   d. for(x = 1; x == 10; x++);

3. Mi lesz a kiírás eredménye?

```
i=3; j=1; k=2;
k+=++i+--j;
printf("%d  %d  %d\n%", i,j,k);
```

4. Mi lesz a kiírás eredménye?

```
i=3; j=1; k=2;
k=-i--+j;
printf("%d  %d  %d\n%", i,j,k);
```

5. Mi lesz a kiírás eredménye?

```
i=3; j=1; k=2;
k=-i+++j;
printf("%d  %d  %d\n%", i,j,k);
```

6. Mi lesz az a, b és c változók értéke az egyes kifejezések kiértékelése során?

```
int a, b, c;
a = b = c = 5;
c = a++ * (b%3);
c = a < b ? a+2 : b-3;
b--; a++;
```

7. Mennyi az a változó értéke az alábbi programrészlet végrehajtása során?

```
int a = 5;
int *p;
p=&a;
*p=a+*p-2;
printf("%d\n",a);
```

8. Mennyi az x változó értéke az alábbi programrészlet végrehajtása során?

```
int x = 7;
int *p, **q;
p=&x;
q=&p;
x-=*p + **q;
printf("%d\n",x);
```

9. Mennyi az s változó értéke az alábbi programrészlet végrehajtása során?

```
double a1=1.0, b1=3.0;
double s;
s = a1 + 1/b1 * 2 + 64;
printf("%x\n",(char)s);
printf("%d\n",(char)s);
```

10. Mi lesz az k változó értéke az alábbi programrészlet végrehajtása során?

```
int i, j, k;
i=7; j=5;
if(i>=j && 0)
 k=3;
else
 k=1;
printf("%d\n",k);
```

11. Mi lesz a kiírás eredménye?

```
char q=3;
int m=2;
double dd,d=1.20;
dd = (int)(q+m*d);
printf("%lf\n",dd);
```

12. Mi lesz a kiírás eredménye?

```
int ax,b;
ax=0x097;
b=0x24; // 16-os számrendszer beli érték
ax= ax | b;
printf("b=%x, ax=%d\n",b, ax);
```

13. Mi lesz az x változó értéke az alábbi programrészlet végrehajtása során?

```
int x, y;
x = (y = 1 , y + 4);
```

14. Mi lesz az i, j és k változók értéke az egyes kifejezések kiértékelése során?

```
i=1; j=3; k=2;
k+= --i-j--;
```

15. Mi lesz a j változó értéke az alábbi programrészlet végrehajtása során?

```
int k=0, j=0;
int i;
for(i=1;i<=3;i++)
 j=k++;
printf("%d\n",j);
```

16. Mi lesz a kiírás eredménye?

```
int ax,a ;
ax=0x057;
a=0x033;
ax |= (1UL << 3);
a &= ~(1UL << 3);
printf(" ax=%d\n   a=%d\n", ax, a);
```

2.11. III. ZH2 Mintafeladatsor

Figyelem! Az Excel feladatok megoldásához – paraméterezésükre vonatkozó megkötések miatt – nem használhatóak a darabteli, szumha, átlagha, szorzatösszeg, átlaghatöbb, darabhatöbb, stb., fkeres, vkeres!

	A	B	C	D	H	I	J
1	Ki locsolt?	Hány éves	Kit locsolt?	Hány éves	Hány percet töltött ott?		Ki locsolt?
2	Ács Máté	20	Belák Diána	21	15		Ács Máté Xxxxxxxx
3	Ács Máté	20	Laki Zsófia	19	15		Ács Máté Yyyyy
4	Ács Máté	20	Pete Diána	20	20		Ács Máté Zzzzz
5	Ács Máté	20	Pittnauer Emma	21	15		Ágoston Róbert Xxxxxxxx
6	Ágoston Róbert	15	Bálint Zsófia	18	5		Ágoston Róbert Yyyyy
7	Ágoston Róbert	15	Hertz Csilla	22	10		Ágoston Róbert Zzzzz
8	Ágoston Róbert	15	Radnóti Andrea	22	5		Alexa Gusztáv Xxxxxxxx
9	Ágoston Róbert	15	Szegedi Zsuzsa	19	5		Alexa Gusztáv Yyyyy
10	Alexa Gusztáv	22	Borbély Csilla	22	5		Alexa Gusztáv Zzzzz
11	Alexa Gusztáv	22	Gáncs Ildikó	22	5		Arany István Xxxxxxxx
12	Alexa Gusztáv	22	Huszár Katalin	16	15		Arany István Yyyyy
13	Alexa Gusztáv	22	Vadász Tamara	21	5		Arany István Zzzzz

1. Mit csinál az alábbi összetett függvény?

   {=HA(MARADÉK(SOR();3)=2;"";HA(MARADÉK(SOR();3)=0;SZUM(HA(BAL(J2;HOSSZ(J2)-6)=A$2:A$661;1));KEREKÍTÉS(ÁTLAG(HA(BAL(J2;HOSSZ(J2)-6)=A$2:A$661;D$2:D$661));0))))}

2. Válaszoljon a következő rövid kérdésekre!

   a. Számolja ki képlet segítségével az átlagos locsolási időt!

   b. Számolja ki képlet segítségével a leghosszabb locsolással töltött időt!

   c. Számolja ki képlet segítségével a locsolással töltött időt!

   d. Írassa ki képlet segítségével a 22 éves fiúk számát!

3. Tároljon egy karaktert az M1 cellában! M2 cellába írassa ki a következő szöveget: „x darab @ karakterrel kezdődő fiú név van a listában.”, ahol x a @ karakterrel kezdődő fiú nevek számát jelöli, a @ karakter az M1 cellában tárolt érték (fenti ábra).

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
1		Ausztrália	Malajzia	Kína	Törökö.	Spanyolo.	Monacó	Kanada	Európa	Anglia	Németo.	Magyaro.	Belgium	Olaszo.	Szingapúr	Japán	Korea	India	Abu-Dhab	Brazília	Total		Place	Point
2	Adrian Sutil	2					6		2		8		6		4								1	25
3	Bruno Senna													2									2	18
4	Daniel Ricciardo																						3	15
5	Felipe Massa	6	10	8				8	10	10	10	8	4	8	2	6	8						4	12
6	Fernando Alonso	12	8	6	15	10	18		18	25	18	15	12	15	12	18	10						5	10
7	Heikki Kovalainen																						6	8
8	Jaime Alguersuari							4	4	1		1		6			6						7	6
9	Jarno Trulli																						8	4
10	Jenson Button	8	18	12	8	15	15	25	8			25	15	18	18	25	12						9	2
11	Jerome d'Ambrosio																						10	1
12	Kamui Kobayashi		4	1	1	1	10	6			2													
13	Karun Chandhok																							

Számolja ki képlettel az egyes versenyzők összpontszámát!

Adja meg képlettel a locsolások számát!

4. Gépeljen egy XXX nevet a Z2 cellába, gépeljen egy YYY helyszínt a Z3 cellába. Írjon egy képletet, amely megadja XXX versenyző YYY helyszínen elért pontszámát!

5. Készítse el az előző feladathoz tartozó szöveget az alábbi formátumban: „XXX versenyző YYY helyszínen elért pontszám:”

6. Mi a kiíratás eredménye?

```
int c1=12, c2=45;
     c1^=c2;
     c2^=c1;
     c1^=c2;
printf("c1= %X\tc2= %X",c1,c2);
```

7. Írjon C programot a következő összeg rekurzív előállítására.

$$\sum_{i=1}^{n} i^2$$

8. Bizonyítsa be indukcióval, hogy:

$$2 \cdot \sum_{i=1}^{n} (i-1) = n \cdot (n-1)$$

9. Rajzolja fel a for ciklus blokkdiagramját!

10. Mit valósít meg az álabbi programrészlet? Írjuk át úgy, hogy ugyanezt valósítsa meg, de do-while ciklussal!

```
int x, k = 0;
scanf("%d", &x);
while(x){
 scanf("%d", &x);
 k++;}
printf("%d", k);
```

11. Mi lesz a kiírás eredménye?

```
int A[45],  i=0;
for ( ; i < 45; ++i )
    A[i] = 2*(i-3);
printf("%d\n%d\n",*(A+25), A[i-3]);
```

12. Írjunk függvényt, mely megcseréli két változó tartalmát segédváltozók használata nélkül. A main függvény írja ki a két változó eredeti, majd a függvény hívása utáni értékeit!

13. Mi lesz az a, b és c változók értéke az egyes kifejezések kiértékelése után? Az értékeket görgetjük.

```
int a, b, c;
a = b = c = 15;
c = (a%2) * b--;
c = a < b ? a+5 : b-7;
b++; --a; c--;
```

14. Mit ír ki az alábbi programrészlet?

```
int a=6; int b=4;
int *p=&a; int *q=&b;
a-= *q;
printf("%d\t%d\n", *q,*p);
p=q;
printf("%d\t%d\n", *q,*p);
```

15. Mi lesz a változók értéke az alábbi programrészlet végrehajtása után? Indokolja meg válaszát!

```
int i, j, k;
i=-2;j=-3, k=10;
if(i=!j || k)
 k=-1 * (--i || j);
else
 k=-2 * (i++ && k);
printf("%d\t%d\t%d\n",i,j,k);
```

16. Mit ír ki az alábbi programrészlet?

```
int m=3;
char b=-9;
double a, d=4.75;
a = (int)(b-m*d);
printf("%.2f\t%d\n",a,(char)d);
```

17. Mi a kiíratás eredménye?

```
double m [4][3]={9,7,0.5,3.14,2,5.0,4,6.5,0,8,5.95,7.2};
printf("%.2f , %.4f \n", m[1][1], m[3][1]);
```

18. Mi a kiíratás eredménye?

```
int i, j;
for(i=1;i<=4;i=i+1) j=j+1;
printf("%d\n",j);
```

19. Mi a kiíratás eredménye? Indokolja meg válaszát!

```
char x;
for (x=10;x >= 10; x++);
printf("%d\n",x);
```

20. Mi a kiíratás eredménye? Indokolja meg válaszát!

```
int i;
for (i=1; i=!10; i--)
printf ("%d\n",i);
```

21. Mi a kiíratás eredménye? Indokolja meg válaszát!

```
char x;
for (x=1; x==5; ++x);
printf("%d\n",x);
```

22. Írjon programot, amely előállít 10 darab véletlen egész számot a [20;50] intervallumon és tárolja ezeket egy tömbben!

2.12. IV. ZH2 Mintafeladatsor

Figyelem! Az Excel feladatok megoldásához – paraméterezésükre vonatkozó megkötések miatt – nem használhatóak a darabteli, szumha, átlagha, szorzatösszeg, átlaghatöbb, darabhatöbb, stb., fkeres, vkeres!

	A	B	C	D	H
1	Ki locsolt?	Hány éves	Kit locsolt?	Hány éves	Hány percet töltött ott?
2	Ács Máté	20	Belák Diána	21	15
3	Ács Máté	20	Laki Zsófia	19	15
4	Ács Máté	20	Pete Diána	20	20
5	Ács Máté	20	Pittnauer Emma	21	15
6	Ágoston Róbert	15	Bálint Zsófia	18	5
7	Ágoston Róbert	15	Hertz Csilla	22	10
8	Ágoston Róbert	15	Radnóti Andrea	22	5
9	Ágoston Róbert	15	Szegedi Zsuzsa	19	5
10	Alexa Gusztáv	22	Borbély Csilla	22	5
11	Alexa Gusztáv	22	Gáncs Ildikó	22	5
12	Alexa Gusztáv	22	Huszár Katalin	16	15
13	Alexa Gusztáv	22	Vadász Tamara	21	5

1. Mit csinál az alábbi összetett függvény?

   {=HA(MARADÉK(SOR();3)=2;"";HA(MARADÉK(SOR();3)=0;SZUM(HA(BAL(J2;HOSSZ(J2)-6)=A$2:A$661;1));KEREKÍTÉS(ÁTLAG(HA(BAL(J2;HOSSZ(J2)-6)=A$2:A$661;D$2:D$661));0))))}

2. Válaszoljon a következő rövid kérdésre!

   a. Adja meg képlettel, hogy átlagosan hány percig tartott a locsolás ezen a húsvéton

3. B100 cellába írassa ki a következő szöveget: „x darab olyan gyümölcs van a listában, amely szénhidrát tartalma egész szám.”, ahol x azoknak a gyümölcsöknek a számát adja meg, amelyek szénhidrát tartalma egész szám.

	A	B	C	D	E	F	G
1	Megnevezés	energia (kj)	kcal	fehérje(g)	sav (g)	szénhidrát (g)	rost (g)
2	Alma	130	31	0,4	0,4	7	1,3
3	Ananász	223	53	0,4	0,7	12	0,4
4	Banán	441	105	1,3	0,1	24,2	0,4
5	Birsalma	176	42	0,6	0,9	9,1	1,9
6	Citrom	113	27	0,4	5,8	2,3	2,2
7	Cukordinnye	168	40	0,3	0,1	9,5	0,7
8	Cseresznye	265	63	0,8	0,7	14	0,4

A táblázat utolsó sora a 23. sorban található.

a. Adja meg képlettel, hogy átlagosan hány percig tartott a locsolás ezen a húsvéton!

b. Adja meg képlettel, hogy hány gyümölcs szerepel a listában!

c. Adja meg képlettel, hogy mennyi a felsorolt gyümölcsök összesített energiatartalma kj-ban kifejezve!

d. Adja meg képlettel a legnagyobb energia tartalmat kj-ban kifejezve!

e. Adja meg képlettel azoknak a gyümölcsöknek a számát, amelyek 0,4 g fehérjét tartalmaznak!

4. Gépeljen egy AAA gyümölcsöt a K1 cellába, gépeljen egy összetevőt a K2 cellába. Írjon egy képletet, amely megadja AAA gyümölcs mennyit tartalmaz a BBB összetevőből !

5. Készítse el az előző feladathoz tartozó szöveget az alábbi formátumban. „AAA gyümölcsben található BBB összetevő mennyisége:”

6. Mi a kiíratás eredménye?

```
double a=25.4782, b=-34.2;
     a+=b;
     b=a-b;
     a-=b;
printf("\na= %lf \nb= %.3f",a,b);
```

7. Írjon C programot n! rekurzív előállítására.

8. Bizonyítsa be indukcióval, hogy:

$$\sum_{i=0}^{n} i^3 = \frac{n^2 \cdot (n+1)^2}{4}$$

9. Rajzolja fel a do while ciklus blokkdiagramját!

10. Mit valósít meg az álabbi programrészlet? Írjuk át úgy, hogy ugyanezt valósítsa meg, de do-while ciklussal!

```
int a, os=0;
    for (a=1;a<10;a++){
        if(!(a%2)){
            printf("%d ",a);
            os=os+a;
```

```
        }
    }
    printf("\n%d\n",os);
```

11. Mi lesz a kiírás eredménye?

```
int B[25],  i=0;
for ( ; i < 20; ++i )
    B[i] = 4*(2*i+1);
printf("%d\n%d\n", *(B+11), B[i-7]);
```

12. Írjunk függvényt, mely megcseréli két változó tartalmát segédváltozók használata nélkül. A main függvény írja ki a két változó eredeti, majd a függvény hívása utáni értékeit!

13. Mi lesz az a, b és c változók értéke az egyes kifejezések kiértékelése után? Az értékeket görgetjük.

```
int a, b, c;
a = b = c = 7;
b = + +b + (a%5);
c = c < a ? a+2 : b/5;
b+=a; a%=2; c++;
```

14. Mit ír ki az alábbi programrészlet?

```
int a=10; int b=20;
int *p=&a; int *q=&b;
*p +=(*q)++-15;
printf("%d\t%d\n", a,*q);
*q+=*p;
printf("%d\t%d\n", a,*q);
```

15. Mi lesz a változók értéke az alábbi programrészlet végrehajtása után? Indokolja meg válaszát!

```
int i, j, k;
i=-2; j=-3; k=0;
if(i= =j || k)
 k=-1 * (++i || j);
else
 k=-2 * (i && k);
printf("%d\t%d\t%d\n",i,j,k);
```

16. Mit ír ki az alábbi programrészlet?

```
int a=0xeb, b=0xac, c=0xeb;
c &= b<<3;
b ^= a>>1;
printf("%X %X \n",c,b);
```

17. Mi a kiíratás eredménye?

```
double m [5][3]={9.5, 7, 5, 6, 4.3, 7.4,5.5,2,1,0.5,3};
printf("%f , %.3f \n", m[2][1], m[1][2] );
```

18. Mi a kiíratás eredménye?

```
int k=1, j=0, i = 0;
while (i=15)
 if (i=1)
  j=k++;
 else
  i++;
printf("j=%d\ti=%d\n",j,i);
```

19. Mi a kiíratás eredménye? Indokolja meg válaszát!

```
int i;
for (i=1; i=!10; i--)
printf ("%d\n",i);
```

20. Mi a kiíratás eredménye? Indokolja meg válaszát!

```
int i=5;
while(i = = 5  |  i= =10) i=10;
printf("%d\n",i);
```

21. Mi a kiíratás eredménye? Indokolja meg válaszát!

```
int i=13;
while (!i) i--;
printf("%d\n",i);
```

22. Írjon programot, amely előállít 12 darab véletlen számot a [−20,70] intervallumon és tárolja ezeket egy tömbben!

Bibliográfia

Alfred V. Aho – Jeffrey D. Ullman: Foundation of Computer Science, 1992, Computer Science Press, New York

Brookshear, J. Glenn: Computer science: an overview 9. ed. (International ed.), cop. 2007, Boston, MA: Pearson; Addison Wesley

Cormen, Leiserson, Rivest, Stein: Introduction to Algorithms, 1990, MIT Press and McGraw-Hill.

Csala Péter – Csetényi Arthur – Tarlós Béla: Informatika alapjai, 2001, ComputerBooks, Budapest

Efraim Turban – R. Kelly Rainer, Jr. –Richard E. Potter: Introduction to Information Technology, 2001, Wiley

Kátai Zoltán, C nyelv és programozás, 2008, Debreceni Egyetem, Debrecen

[H1] *W. Trede-A.Herkelmann-E-Schwarzer: Számítástechnikai alapismeretek*, 1975, Műszaki Könyvkiadó, Budapest