BEVEZETÉS AZ INFORMATIKÁBA 1. rész

TARTALOMJEGYZÉK

Az információ

Az információ fogalma

Az információ létezik, ez tagadhatatlan. Meg lehet-e pontosan határozni, fogalmazni mibenlétét?

Az információ az informatika alapfogalma. Sokféle meghatározás él a mindennapi életben, melyek közösek abban, hogy az információ bizonytalanságot csökkent, és újdonságtartalommal rendelkezik, új ismeretet hordoz. *Az információ tehát olyan ismeret, amely egy jelenséggel vagy folyamattal kapcsolatosan csökkenti a bizonytalanságot, olyan hír, amely újdonsággal szolgál, és hozzájárul egy jelenség megismeréséhez*. [1]

Az információ egyike azoknak a formáknak, amelyekben a külső világ a tudatban megjelenik. Környezetünkből szüntelenül jelek, ingerek, üzenetek érkeznek, melyeket tudatunk feldolgoz, reakciókat váltva ki. Másképpen reagálunk arra a hírre, hogy holnap esni fog, mint arra, hogy ezentúl minden nap esni fog, hiszen ennek a két hírnek más és más a minőségi értéke a számunkra.

Az információ megjelenési formái különbözőek lehetnek. Egy könyvben a betűkből összeálló szavak, illetve az írásjelekkel együtt a mondatok jelentik az információt. Egy feleletre kapott osztályzat szám formában megjelenített információ. Egy térképen a színek, különböző vastagságú vonalak, magyarázatul szolgáló képecskék (piktogramok) is információt hordoznak.

Érdekesség, olvasmány

Információk az élővilágban.
Nem csak az emberek képesek arra, hogy információt szerezzenek, közöljenek egymással, hanem minden élőlény. A **növények** érzékelik a környezetből érkező információkat, mint például a fény, a hőmérséklet, a páratartalom, a csapadék, és ezek változásai. A mimóza például azonnal összecsukja leveleit, ha hozzáérsz, ha egy növény földje kezd kiszáradni, bezárja pórusait, hogy kevesebb vizet párologtasson. Az **állatok** információs rendszere már bonyolultabb. Táplálékuk megszerzéséhez, ellenségeik elkerüléséhez a környezetből információkat szereznek, illat, hő, hang, fény formájában. Ezeket az információkat megjegyezhetik, sőt továbbíthatják is társaiknak. Például a méhek „tánccal” „mesélik el” társaiknak, hol találtak értékes lelőhelyet.[15]

Feladat

1. *Időjárás jelentés:* Eleinte felhős lesz az ég, többfelé várható zápor, zivatar. A csúcshőmérséklet 15 és 20 fok között várható.
   a) Ha egy régi újságban olvasod ezt a hírt, van-e számodra információtartalma? Miért?
   b) Ha ma reggel hallottad ezt az időjárás jelentést a rádióban? Mi okozza a különbséget?
   c) Te Magyarországon élsz, de ez a hír Japánra vonatkozott. Kaptál-e információt? Indokolj!
2. A *közlekedési jelzőlámpán* a piros lámpa világít.
   a) Mit jelent ez? Hordoz információt számodra, ha éppen át akarsz kelni az úttesten?
   b) Annak, aki életében először lát ilyen lámpát, biztosan ugyanazt jelenti? Jelent-e valamit? Indokolj!

Közlemény, hír, adat, információ

Az információ világunk, illetve a tudomány és a technika egyik alapvető fogalma, akárcsak az anyag vagy az energia. Az információ önmagában való előállítása nem lehetséges. Továbbítani, tárolni, feldolgozni csak közleményt lehet, amelynek van információtartalma. Köznapi értelemben **hír**nek nevezzük az újdonságot hordozó üzeneteket. Az informatika ennél precízebb: a jelekké alakított információt **közlemény**nek nevezzük. A közlemény azonban terjengős, azaz hordoz olyan elemeket is, amelyeket elhagyva értelmezése nem változik. Például a rádióban ezt halljuk: „Ma 2002. december 30-a hétfő van, délután 15 óra.” Claude Shannon foglalkozott olyan közlemények előállításával, amelyek minden jelükben információt hordoznak. Az ilyen „szupertömény” közleményt nevezte el hírnek. [3] Előző példánknál maradva a legtömörebb formában előálló közlemény, tehát hír: 2002. 12. 30. 15:00.

Az **adat** tulajdonképpen rögzített ismeret, az információ ábrázolására használt jelsorozat. Mindazokat a jeleket, amelyek a feldolgozáshoz szükségesek, vagy annak folyamán keletkeznek, illetve eredményeképpen megjelennek, adatoknak tekintjük. Nem biztos, hogy van újszerűsége, hiszen ez attól függ, hogy ki kapja. A hír ezzel szemben mozgásban lévő ismeret.

Feladat

1. Egy menetrend adatok tömkelege. Kinek, milyen körülmények között hordoz információt?
2. Gyűjts adatokat menetrendekből!
3. Melyik mondat tekinthető hírnek, ha a jelkészlet a szokásos írásjelekből áll?

Hazánk, Magyarország hét országgal szomszédos.
Kedden történelemből dolgozatot írunk.

4. Tedd a következő mondatokat rövidebbé! Ha tudsz, akkor készíts hírt!

Holnap esni fog az eső.
Azt szeretném elmondani, hogy a hétvégén kirándulni megyünk, ha jó idő lesz.

Az információ mennyisége, egységei

Az információt mindig jelek, jelsorozatok hordozzák. Fontos, hogy a jeleket megértsük, különben nem kapunk információt. Nem minden jel hordoz számunkra információt: vagy azért, mert nem értjük, vagy azért, mert már ismertük a tartalmát.

Lehet-e mérni az információt? Műszerrel nem, de egy üzenet információtartalma számítással meghatározható! A probléma ott kezdődik, hogy az információk nagyon sokfélék lehetnek. Feldolgozandó információ lehet akár egy kép, egy szöveg, egy elektronikus jel is. Ezek jellemzőinek megmérése után az a feladat, hogy a számítógép részére fogadható jellé alakítsuk az információkat.

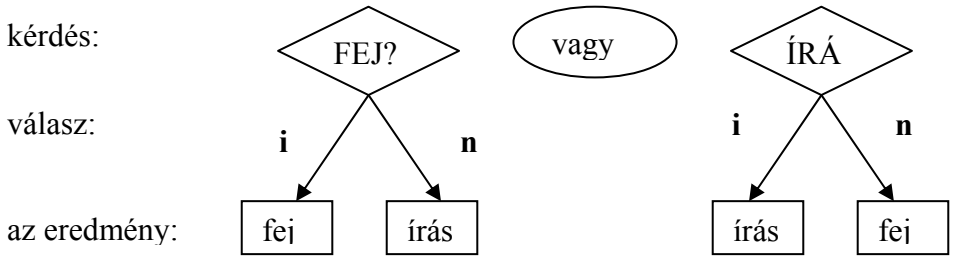
Mindenekelőtt azt kellene tisztázni, mitől lehet nagyobb, értékesebb egy információ egy másiknál. Tekintsük a következő két mondatot:

1. *„Gyurinak januárban van a születésnapja.”*
2. *„Gyurinak január 13-án van a születésnapja.”*

Melyik mondat hordoz számunkra több információt (feltételezve, hogy eddig ezt nem tudtuk)? Az első mondat a 12 hónap közül nevez meg egyet, míg a második az év 365 napjából jelöl ki egyet. Ezért úgy érezzük, hogy a második mondat hordoz több információt számunkra. Azt mondhatjuk tehát, hogy annál nagyobb az információ mennyisége, minél nagyobb az egyformán lehetséges, azonos valószínűségű esetek száma, mielőtt az információt megkapjuk. Annak a közlésnek pedig, amely nem ad új ismeretet, az információtartalma 0. A példánál maradva: a napot nehezebb eltalálni, hiszen 365 féle lehetőség van, míg a lehetséges hónapok száma mindössze 12! [10] Összefoglalva: *minél kisebb egy esemény bekövetkezésének a valószínűsége, annál több információt jelent, ha megtudjuk.*

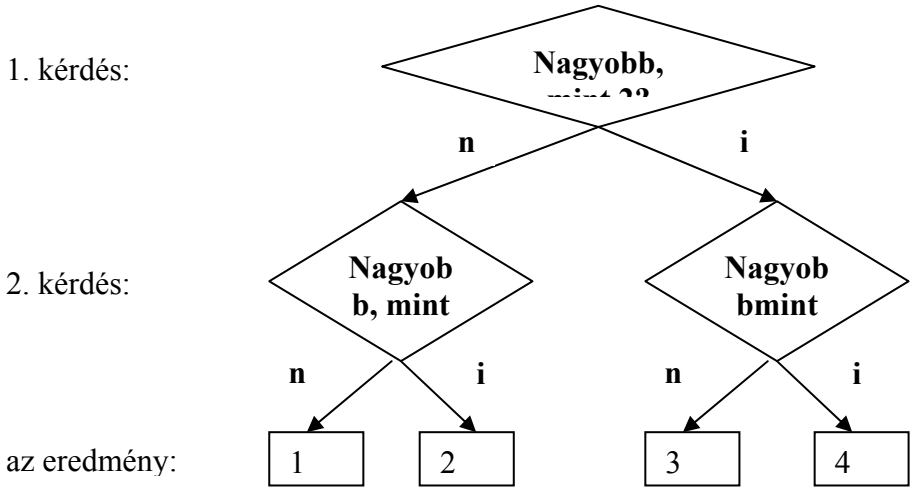
Dobjunk fel sokszor egymás után egy pénzérmét! A pénzfeldobásnak kétféle kimenetele lehet: fej vagy írás. Az eredmények valószínűsége azonos. Amikor az eredményt *kétféle jel*lel tudjuk leírni, *a kettőt együtt bináris jel*nek nevezzük. Például „+” és „–,” „igen” és „nem”, „fej” és „írás”, „0” és „1”. Az információmennyiség mértékegysége a bit, az angol Binary unIT (kettes egység) alapján.

Egy pénzérme feldobásakor egyetlen eldöntendő kérdésre adott válasz után biztosan megmondható a feldobás eredménye:



Egységnyi információnak nevezzük azt az információmennyiséget, melynek kétféle lehet a megvalósulása (egyetlen kérdéssel a megoldáshoz jutunk). Ennek a legkisebb információmennyiségnek az elnevezése tehát a bit. Ha megtudjuk, hogy két azonos valószínűségű lehetőség, esemény közül melyik következett be, akkor 1 bit információmennyiséghez jutunk. A fenti példa eszerint 1 bit információt tartalmaz. [1]

A 4 eset közüli választás már 2 eldöntendő kérdést igényel. Tegyük fel, hogy az 1,2,3,4 számok közül az egyiket kell kitalálnunk. Ez pontosan két eldöntendő kérdéssel tehetjük meg:



Hasonlóképpen kikövetkeztethető, hogy 8 szám közüli kiválasztás 3 kérdést, 16 szám közüli 4 kérdést igényel, azaz 3 bit illetve 4 bit információt hordoz. Ha *i* jelöli a kísérletek (kérdések) számát, *n* pedig az összes lehetőséget (kitalálandó számok), a következő összefüggések állapíthatóak meg [3]:

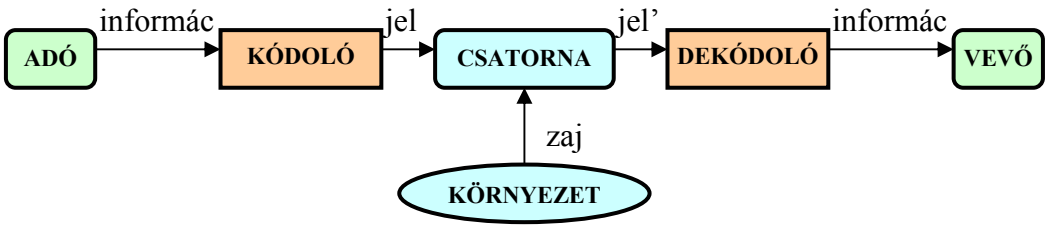
$$n = 2^i, \text{ azaz } \log_2 n = i$$

Az információ útja

Az információ testet öltése kulcsfontosságú folyamat az informatikában. Az információforrás, a feladó valamilyen módszerrel elkészíti a közleményt, azaz a közlendő információt jelekké alakítja. Ezután eljuttatja a címzetthez, ehhez pedig továbbító közeget vesz igénybe. A kódolás során olyan jelsorozattá kell alakítani az információt, amelynek továbbítására ez a közeg alkalmas. A közeget „nem mindig vesszük észre", pedig elengedhetetlen a szerepe. A címzett a közlemény átvétele után megpróbálja a közleményből az információt kinyerni, értelmezni a jeleket.

Az informatikában a feladót nevezzük **adó**nak, a címzettet **vevő**nek. A közlemény előállítása a **kódolás**, az információ kinyerése pedig a **dekódolás**. A továbbító közeg szakszóval: átviteli **csatorna**.

A csatorna feladata, hogy a kódolt közleményt, a jeleket eljutassa az adótól a vevőig. A **környezet** viszont erre a csatornára is hat. Az átvitelt zavarhatják a környezetből érkező hatások, ezért a továbbítandó jelek torzulhatnak, sőt akár el is veszhetnek. Ezeket a zavaró hatásokat nevezzük **zaj**nak.



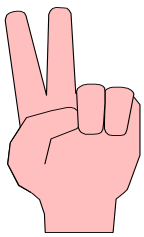
A valóságban nem csak egylépéses kódolás és dekódolás zajlik le. Egy példán végigkövetve az információ útját az adótól a vevőig, tisztázódnak a fogalmak. Egy telefonbeszélgetés során a hívó fél a kezdeményező, ő az adó, a hívott személy pedig a vevő. A kommunikáció – információcsere – során azonban ezek a szerepek folyamatosan cserélődhetnek, attól függően, ki a beszélő. (Feltételezzük, hogy nem egyoldalú ez a beszélgetés.) Az adó elsődleges kódrendszerén, azaz az anyanyelvén mondja el a mondanivalóját. Amint a megfogalmazott gondolatait kimondja, hanggá alakulnak a gondolatok, a hangot pedig a telefon mikrofonja átalakítja elektromos jelekké,

melyek továbbításra kerülnek. Ez a folyamat többlépéses kódolás. A vevő oldalán lévő készülék az elektromos jeleket ismét hanggá alakítja, a vevő érzékeli a hangokat, és értelmezi azokat, megtörténik a többlépcsős dekódolás. Az embereknél alapvető kódolásnak tekinthető a beszéd, az írás, alapvető dekódolásnak pedig a meghallgatás és az olvasás.

A környezetből különböző zavaró hatások érkezhetnek: mások beszéde, „áthallás” a vonalak között, amelyek a csatornán való továbbításkor zavarják a jelet, de zaj már az adó jeléhez is keveredhet. (A zaj minden zavaró tényező hatás összefoglaló neve, így valójában egyáltalán nem biztos, hogy akusztikus jellegű!) Fontos, hogy a jel-zaj arány elég nagy legyen, ellenkező esetben a zaj csökkentése, vagy a jel erősítése szükséges. Egy módszer a zaj ellen, ha ugyanazt az információt többször is közöljük, esetleg más módon. Az ilyen közleményt **redundáns**nak, **terjengős**nek nevezzük. [10] A redundáns közleményben az információ nincs a legtömörebben megfogalmazva, de így nagyobb biztonsággal lehet venni. A redundancia tehát az üzenet információt (*új* ismeretet) nem tartalmazó részarányának mértéke. Mivel a gyakorlatban az átvitel során fellépő zavaró hatások elkerülhetetlenek, a redundanciára szükség van ahhoz, hogy egy üzenet még akkor is értelmezhetővé váljon, ha zajok lépnek föl.

Az ember többféle jel érzékelésére képes érzékszervei közvetítésével. A szem az optikai, a fül az akusztikus, az orr és a nyelv a kémiai, a bőr a mechanikai és termikus jelekké alakított információkat fogja fel.

Jeleket adó szerveink segítenek gondolataink közlésében. Képezhetünk hangokat, szagokat, feromonokat, testünk hőt sugároz. Az emberre igen jellemező jeladási módszer a viselkedés: a mozgás, testtartás, mozdulatok (gesztikuláció), az arc- és szemjáték (mimika). Sokrétű jelrendszer, melyet egyrészt tudatosan használunk, másrészt ösztönös. Ezt a „nem szavakkal történő” kapcsolatteremtést nevezzük metakommunikációnak.



Érdekesség, olvasmány

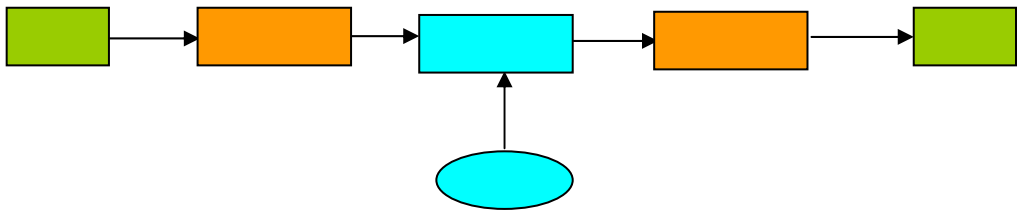
1. **A csatorna fontossága.** A csatorna fontosságának igazolására szolgálhat a következő kísérlet. Egy üvegbura alá metronómot (vagy bármilyen, jól hallható hangot kiadó tárgyat) helyezünk. A metronóm hangja az üvegen keresztül is jól hallatszik. Kezdjük folyamatosan kiszivattyúzni a levegőt a burából! A hang egyre halkabb, sőt, egy idő után nem is hallani, pedig a metronóm láthatóan tovább működik. Az ütések újra hallhatóvá válnak, ha a levegőt

visszaeresztjük. A levegő volt tehát a közeg, amely a hanghullámokat továbbította, tehát az átviteli csatorna. Beszéd közben szintén a levegő a csatorna, amely, ha hiányozna, nem hallanánk egymás szavát sem. [7]

2. **A beszélt nyelvek** általában olyanok, hogy az értelmes mondatok esetében a vevő akkor is ki tudja találni a szöveg értelmét, ha majdnem a felét nem hallotta. Ez a „biztonsági tartalék" jellemző a nyelvekre, ez pótolja az ellenőrzést. Persze, komoly problémákhoz vezethet már egyetlen szó félreértése is! (Nem mindegy, hogy valakinek téli a nadrágja, vagy teli...)

Feladat

1. Fejezd ki arcjátékkal, majd metakommunikációval a következő érzéseket, gondolatokat: támadó, dühös, boldog, érdeklődő, unatkozó, barátsággal közeledő, rosszalló, helyeslő, rémült, értetlenkedő, várakozó, türelmetlen.
2. Mutasd be a következő mondatot különböző érzelmi töltéssel, gesztikulációval, mimikával! „Sanyi elvette a tollamat."
3. Készíts doboztelefont! Hozzávalók: két üres konzervdoboz, 15-20 méter hosszú vékony fém huzal vagy damil. A dobozok alján fúrj lyukat, és a huzallal kösd össze őket! Próbáld ki, milyen távolságig használható!
3. Gyűjts példákat a növény-és állatvilágból információcserére!
4. Álljatok sorba, az első játékos súgja a következő mondatot a mellette álló fülébe: „Holnap megírjuk matekból a témazáró dolgozatot!" Végig ért-e hibátlanul a mondat?
5. Próbáljátok ki ezt a két mondatot is: „Tegnap egy lila kacsával ebédeltem a Zöld Liba étteremben, narancssárga pityókát ettünk." „Öt kettő nulla hat hét hat kilenc kettő." Most is végigértek hibátlanul a mondatok? Mi lehet ennek az oka?
6. Két, bekötött szemű tanulót vezessetek a tanterem két távoli pontjára. A feladat: keressék meg egymást. Nehezíti-e a megoldást, ha valamelyik érzékszervüket nem használhatják? Mikor a legnehezebb a dolguk? Próbáljátok ki!
7. Anna és Bea beszélget az órán, miközben a tanár az új tananyagot magyarázza. Hova írnád a szereplők nevét? Mit írnál a többi dobozba?

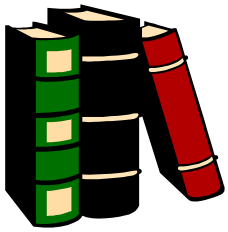


Jelek, jelrendszerek

A jelek nélkülözhetetlen szerepet játszanak az életünkben. Az információ jelekké alakítva jut el az adótól a vevőig, az információt jelek, jelhalmazok hordozzák. Azonban nem minden jel hordoz számunkra információt: vagy azért, mert nem értjük, vagy azért, mert már ismertük tartalmát, nincs újdonságtartalma.

A jelek egy részét érzékszerveinkkel is felfoghatjuk, de nem minden jel ilyen: ezek technikai jelek, mint például a rádióhullámok, jel a CD lemezen, a mágneslemezen, elektromos jelek. A jel tehát érzékszerveinkkel, vagy műszereinkkel felfogható, mérhető jelenség. A jel mindig egy másik dologra, a jelenségre utal. Például a „könyv" szó öt betű egymás után: k, ö, n, y, v. Ez a

jelölő, ez jelöli a könyvet. A jelölt pedig maga a könyv, a tárgy. A jelölő és a jelölt együtt alkotják a jelet. A jelek egy halmaza használati szabályaikkal együtt jelrendszert alkot. A jelrendszer tulajdonképpen egy közlés tartalmát közvetítő fizikai objektum. A jelek gyakran felbonthatók további alkotórészekre, elemi jelekre, melyeknek nem biztos, hogy van önálló jelentésük is, hanem más jelekkel együtt alkotnak értelmes egységet. Így az előző példában a k, ö, n, y, v betűk – elemi jelek – önmagukban nem értelmesek. Ráadásul fontos, hogy a jelek mindig jelrendszert alkotnak, melyekben a használati szabályok nélkül nem sokra megyünk: a fenti betűk más sorrendben nem jelölnek semmit számunkra. [12]



Leggyakrabban kép, beszéd és írás alapján kommunikálunk. Látás útján szerezzük meg információink 70-75%-át, hallás útján kb. 20%-át. Az érzékszerveinkkel felfogható jelek közül tehát kiemelkedő fontosságúak a látható (vizuális) jelek.

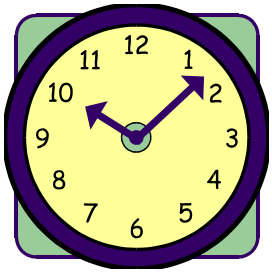
A jelek egy másfajta csoportosítása szerint beszélhetünk *folytonos* és *diszkrét* jelekről. A diszkrét, vagy különálló jel véges sokféle lehet, azaz véges sokféle értéket vehet fel. Ilyen jelek például a betűk, a számjegyek, de a kétféle értékkel bíró logikai jelek is: hamis (0) vagy igaz (1), a ki-be kapcsolható fényjelek, a morzejelek. A diszkrét jelekből hosszabb sorozatok alkothatók: a betűkből szavak, a számjegyekből többjegyű számok. A számjegyekkel leírható jeleket nevezhetjük digitális jeleknek.

A folytonos jelek esetén a jel folyamatos, nincs megszakítás. Ilyen például az óramutató állása, kürtjel, vagy a régi lemezeken a hangbarázda. [12]

A fenti csoportosítást figyelembe véve fontos megkülönböztetnünk két alapvető jelrendszert:

Az *analóg jelrendszerek* esetén a folyamatos, folytonos állapotokon keresztülmenő változás mögött mindig valamilyen fizikai jelenség van. Az analóg információ és jel között mindig kölcsönös és egyértelmű kapcsolat áll fönn. Az információ hordozója mindig folytonosan változtatható és mérhető fizikai jelenség.

A *digitális jelrendszer* diszkrét, egymástól jól megkülönböztethető jelekből épül föl. Minden információ számokat megtestesítő állapotok formájában adódik.



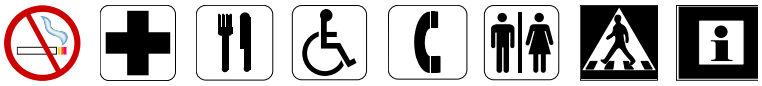
E két jelrendszer elvének megértéséhez figyeljük meg a következő példát: legyen az információ az, hogy eltelt egy óra. Az idő múlása a tartalmi információ, de mi *hordozza* ezt az információt, mi, és hogyan *jeleníti meg* a számunkra? Analóg jelrendszer esetén a számlap, a mutatók helyzete nyújtanak tájékoztatást. A mutató folytonosan változtatja helyzetét, a jel mögött folyamatos fizikai változás áll: az óra rugójának folyamatos alakváltozása. Az óra működése és kijelzője is analóg. Természetesen nem csak „mutatós” órát ismerünk: a számlapon megjelenő számjegyek is jelezhetik az időt, ezek a digitális jelek hordozzák az információt, sőt maga az óra is lehet digitális működésű (kvarcóra).

Érdekesség, olvasmány

1. **A közúti közlekedés.** A közlekedési táblákat tanulni kell. Mivel a világon mindenhol egységesen használják őket, érdemes megismerni őket akkor is, ha csupán gyalogosként, vagy kerékpárral közlekedünk. Segítenek eligazodni, és a szabályok betartásával biztonságosabbá válik a közlekedés. A táblák jelentésének megfejtésében az egységes jelölés sokat segít, például a tiltó táblák mindegyike kör alakú és körben piros szegélyű. Igyekezz minél több táblát megtanulni!
2. **Jelek a zenében.** A zene számsorrá alakításához a hangokat beszámozhatjuk. Egyrészt a hangmagasságot jelezzük, másrészt az időtartamot. A szokásos módszer a skála alsó dó hangja az 1, re a 2, stb., az időtartamnál az 1 az egész, 2 a fél, 4 a negyed, 8 a nyolcad hangot jelzi.
3. **Jelek a háztartásban.** Sok olyan eszköz van a háztartásban, amelyen könnyen érthető jelek vannak. Ismered ezeket? A magnó, videó és a CD lejátszó kezelő gombjainak jelölései eléggé egységesek, a CD gombjainak egy része kétfunkciós, pillanatnyi jelentését az adott szituáció határozza meg. A vasalón, mosógépen, mikrohullámú sütőn, stb. található jelek az egyszerű és balesetmentes kezelést segítik. „Megfejtésükben” útmutatók, jelmagyarázatok is segítenek.
4. **A térképészeti jelek** szintén fontosak az eligazodásban. A kirándulásokon a turista térképek jelei segítenek. Jelentésük ugyan többségében kitalálhatóak, de érdemes utánuk nézni pontosan. A külföldi térképek jelei kissé eltérőek.

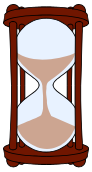
Feladat

1. A magyar ábécé jelkészlete hány karakterből áll?
2. A tízes számrendszer „ábécéje” hány jelből áll? Ezekből a számjegyekből hány szám írható le?
3. Énekórán a kézfej különböző tartásával a szolmizációs hangokat lehet szemléltetni. Ismered ezeknek a kézjeleknek a jelentését? Milyen módszerrel lehet „rögzíteni” a hangjegyeket?
4. Nézz utána, milyen jelek segítségével olvasnak a vakok! (Az írás neve: Braille-írás)
5. Hogyan kommunikálnak a siketek?
6. Mit szimbolizálnak a következő piktogramok?



7. Gyűjts példákat jelekre, jelrendszerekre! Kutakodhatsz a növény- és állatvilágban is!
8. Gyűjts jeleket az írás és a nyomtatás történetéből!
9. Gyűjts technikai jeleket a hírközlés történetéből!

10. Gyűjtsd össze a kedvenc sportágad jelöléseit! (A Sportjátékok könyv segít, vagy az adott sportág szabálykönyve.) Keress újságokban további sportág piktogramokat!
11. Tervezz tantárgyakat szimbolizáló jeleket, piktogramokat!
12. Szervezzetek versenyt: ki tud több piktogramot felismerni közülük, ki tudott több olyat hozni, amit mások nem hoztak? (Ha valaki hozott rossz piktogramot is, beszéljétek meg, mi a hiba benne, hogyan lehetne kijavítani!)
13. Egy homokórán két perc alatt folyik le a homok. Szerinted ez analóg vagy digitális óra?
14. Magyarázd el, hogyan jeleníti meg a mért hőmérsékletet az analóg és a digitális hőmérő.
15. Keress olyan térképet, amelyen szerepel a különböző utak minőségének jelzése! (Autóstérképek, turistatérképek például ilyenek.)
16. Hogyan jelölik a térképeken a folyókat? Milyen módon lehet rajtuk átkelni? Az átkelés módját hogyan jelölik?



Titkosírások

Az üzenetek titkosítása néha nagyon fontos dolog! Különösen akkor, amikor üzleti vagy katonai titkokkal kapcsolatos. A titkosírást az a vágy hozta létre, hogy a leírtakat ne értse meg akárki, csak az, akinek szánták az üzenetet. Az üzenetek titkosítását **kriptográfiának**, vagyis titkosírásnak nevezik. A *kriptos* görög szó, jelentése: rejtett. Az információ titkosítását rejtjelezésnek (sifrírozásnak), megfejtését átírásnak (desifrírozásnak) is nevezik. [6]

Elég egyszerű, viszont kevéssé biztonságos eljárás az, amikor egy-egy betűt mindig ugyanazzal a betűvel helyettesítünk. Betűk helyett érdekes jeleket vagy számjegyeket is használhatunk. Ezeket a betű-jel párokat táblázatba foglalhatjuk. Ezt **kódtáblázat**nak nevezzük, a rejtjelezést **kódolás**nak, a megfejtési folyamatot pedig **dekódolás**nak.

Hogyan lehet megfejteni egy titkosírást, ha nem ismerjük a megoldás kulcsát? Például, ha jól megfigyelsz egy betűhelyettesítéses módszerrel kódolt szöveget, feltűnik, hogy bizonyos jelek vagy jelcsoportok ismétlődnek. A leggyakrabban használt betű az *e*, a magában álló *a, e, s* betű, egy kétbetűs szó gyakran *és*, vagy *az*, stb. Néhány betűt és szót megfejtve, a többi is kikövetkeztethető. Ezért is mondtuk a betűhelyettesítéses módszerre, hogy nem túl biztonságos.

A kézi titkosítás és megfejtés hosszadalmas művelet. A titkosírások megfejtésére gyakran használnak számítógépet. A gép ugyanis alkalmas arra, hogy nagyon sok lehetőséget gyorsan végigpróbáljon, és közölje az eredményt.

Érdekesség, olvasmány

Titkosírások

- **Ókori titkosírás:** Az ókori titkosírások egyik formája az volt, hogy egy rabszolgát kopaszra nyírtak, fejbőrére pedig ráírták az üzenetet. Haja megnövéséig zárt helyen tartották, azután teljesítette feladatát: elvitte a titkos üzenetet. Akinek az üzenet szólt, lenyíratta a rabszolga haját, elolvasta a szöveget. (Vajon hogyan akadályozták meg, hogy más is megláthassa ezt az üzenetet?)
- **A spártaiak titkosírása:** A spártaiak egy bizonyos vastagságú hengeres botra bőrszíjat tekertek fel úgy, hogy a menetek szorosan egymás mellé kerültek. A szöveget egymást követő sorokban a szíjra írták úgy, hogy minden menetre egy-egy betű került. A letekert szíjon értelmetlen betűhalmaz sorakozott.
- **Julius Caesar titkosírása:** Nevezik betűeltolásos titkosírásnak is. Ennek lényege, hogy az ábécét néhány betűvel eltoljuk a másik alatt, a felsőben olvassuk a szöveget, és az alsó ábécé szerint leírjuk. Tulajdonképpen minden betű helyett az ábécében pl. 3-mal utána következőt kell leírni: az a helyett c-t, b helyett e-t, c helyett é-t, stb. Julius Caesarnál a „kulcs" az a=d volt.
- **Rácsos rejtjelezés:** Napóleon egyik generálisa találta ki ezt a titkosírást. A rácsból kivágott négyzetek helyére írták a titkos szöveget, sőt, a rácsot még néha el is forgatták. A címzett hasonló ráccsal olvasta el az üzenetet.

Az üzenetet a címzett akkor tudja megfejteni, ha van egy ugyanolyan rácsa, mint amivel Te elkészítetted a titkos üzenetet. Ha jogtalan kézbe kerül a rejtjelezett szöveg, az illetéktelenek rács híján nem tudják elolvasni a levelet!

- **Könyvkódolásos módszer** Ennél a titkosírásnál nem kell elküldeni az üzenetet! Csupán a megfejtés kulcsát, azaz a betűk kiválogatásának módszerét kell a megfejtőnek ismernie. A feladó és a címzett kiválaszt egy olyan könyvet, amelyikkel mindegyikük rendelkezik. Megállapodnak abban, hogy a könyv melyik oldalát veszik alapul, ezt később lehet változtatni. A feladó a titkosítandó szöveg betűit megkeresi a kiválasztott oldalon. A betű helyett azonban két számot ír le: az első szám azt jelenti, hogy a betű hányadik sorban van, a második szám pedig azt, hogy abban a sorban hányadik betű. Egy-egy betűhöz így más és más számpárok is tartozhatnak, ami nagyban megnehezíti az illetéktelen megfejtők helyzetét!

A következő két titkosítási módszer szerint elsőként számokká alakítjuk az írásjeleket, majd azt továbbalakítjuk a második esetben. A számítógép esetében is majd valami hasonló történik: a közlendő jeleit számokká alakítjuk, majd a számokat vissza írásjelekké.

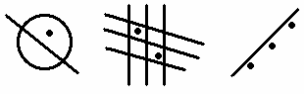
- **Kopogós titkosírás:** A titkos üzenetet betűnként kell továbbítani a betűnégyzet segítségével. Először a betű oszlopának, majd sorának számát kopogtatjuk ki. Írásban is használható ez a módszer: kétjegyű számok képviselik a betűket, mégpedig a tízesek helyén álló szám jelenti az oszlopot, az egyesek helyén álló pedig a sort. (például a M betűt először 4 kopogás, majd 3 kopogás jelöli, írásban pedig a 43 szám.)

	1	2	3	4	5	6
1	A	Á	B	C	D	E
2	É	F	G	H	I	Í
3	J	K	L	**M**	N	O
4	Ó	Ö	Ő	P	Q	R
5	S	T	U	Ú	Ü	Ű
6	V	W	X	Y	Z	

- **Kínai ábécé:** A titkosítás során első lépésben minden betűhöz egy számot rendelünk az alábbi táblázat szerint.

A	Á	B	C	D	E	É	F	G	H	I	Í	J	K	L	M	N	O
1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
Ó	Ö	Ő	P	Q	R	S	T	U	Ú	Ü	Ű	V	W	X	Y	Z	
19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	

A lényeg azonban csak most következik: a számokat jelekké alakítjuk a következő szabály szerint: 3 alkotóelemet használhatunk, amelyeknek értéke is van. A vonal egyet ér, a pont ötöt, a kör pedig tízet. Kis rajzokat készítünk ezekből a jelekből úgy, hogy az összérték megfeleljen a betű számértékének. A kapott rajzocskák emlékeztetnek a kínai írásjelekre, innen kapta a titkosírás a nevét. Mivel egy-egy betűhöz igen sokféle rajz készíthető, a megfejtés nagyon nehézzé válhat! Néhány példa az M betű lehetséges rajzaira:



Feladat

Még egy egyszerű titkosítási módszer is munkaigényes, kitartás és nagy figyelem kell a használatához. Ez idő alatt lemérheted azt is, hogy benned mekkora a kitartás.

1. Készítsd el a saját rácsos titkosírásodat! Jelölj meg egy négyzethálós, téglalap alakú kartonon egyes négyzeteket X-szel, és vágd ki a jelölt négyzeteket! Tedd ezt a rácsos papírt egy másik lapra, és a kivágott helyeken írd be a titkosítandó szöveget! Ezután vedd le a rácsot, és a maradék helyeket véletlenszerűen töltsd ki betűkkel! (Akkor igazán jó ez a rács, ha legalább 35-40 négyzetből áll!)
2. Régen a könyvkódolásos titkosíráshoz gyakran a Bibliát használták. Szerinted miért?
3. Mit gondolsz, a spártai titkosírás megfejtésénél fontos a hengeres bot vastagsága?
4. A kopogós és a kínai titkosírásban nincsenek írásjelek. Hogyan lehetne pótolni ezt a hiányt?
5. Írja le mindenki a nevét kínai titkosírással, dobjátok a neveket egy kalapba, és mindenki húzzon egy titkosított nevet. Fejtsétek meg a neveket!
6. Kártyákra írjatok fel szavakat! Egyikőtök húzzon egy kártyát, és kopogja le a többieknek!
7. Gyűjts további titkosírásokat, vagy találj ki újabbakat!
8. Dekódold a titkosírással írt szöveget a kódtáblázat segítségével!

A	Á	B	C	D	E	É	F	G	H	I	Í	J	K	L	M	N	
O	Ó	Ö	Ő	P	Q	R	S	T	U	Ú	Ü	Ű	V	W	X	Y	Z

Kódolás, dekódolás

Kódolásnak nevezzük azt a folyamatot, amikor *a jeleket meghatározott szabályok szerint egy másik jelrendszerbeli jelekké alakítjuk*. A visszaalakítást dekódolásnak mondjuk. [6] Miért van szükség erre? A többféle ok közül az egyikre, a titkosításra éppen az előzőekben láttunk példákat: tulajdonképpen titkosírás készítésekor is kódolunk, a megfejtéskor pedig dekódolunk! A másik esetről is volt már szó: közleménnyé alakításkor olyan jelekké kell alakítanunk az információt, amelyet az átviteli csatornán keresztül továbbítani tudunk. Ha a jeleket tárolni vagy továbbítani akarjuk, általában kódolni kell őket. A számítógépek és az adatátviteli rendszerek karakterkészletének kódolására szolgáló táblázat az ASCII-kódrendszer (American National Standard Code for Information Interchange, melynek jelentése: amerikai nemzeti szabványos kód információátvitelre). Minden karakternek megfelel egy szám. A számítógépen az ALT billentyűt lenyomva tartva a numerikus billentyűzetről begépelt kódszám hatására a monitoron megjelenik a karakter. Az ASCII-kódtábláról még lesz szó a *Karakterek ábrázolása* című fejezetben.

Érdekesség, olvasmány

A Morze-jelek

Samuel Morse (1791-1872) eredetileg festő volt de foglalkoztatta őt a távjelzés is. 1844-ben elkészítette a Washington – Baltimore közötti villamos távíróvonalat. Morse készüléke pontokat és vonásokat írt papírszalagra, és egy kódtáblázat segítségével lehetett a jeleket betűkké visszaalakítani. A készülék működése röviden: pillanatkapcsolóval történik az adás. Ha lenyomjuk a kart, amely tulajdonképpen kapcsolóként működik, zár az áramkör, ha elengedjük, megszakad. A vevő oldalán egy írókészülék található, melynek fő része egy elektromágnes: ha áram halad át a vezetéken, az elektromágnes magához húz egy kart, ennek a végén van az írószerkezet. Ez húz rövid vagy hosszú vonalat attól függően, hogy mennyi ideig volt nyomva tartva a kapcsoló. Az írószerkezet egy egyenletesen mozgó papírcsíkhoz érintkezve hagy nyomot. [12]

A Morze-féle kódtáblázat:

betű	kód	betű	kód	betű	kód	szám	kód	írásjel	kód
A	●-	I	●●	R	●-●	0	-----	pont	●-●-●-
Á	●--●-	J	●---	S	●●●	1	●----	vessző	--●●--
B	-●●●	K	-●-	T	-	2	●●---	kettőspont	---●●●
C	-●-●	L	●-●●	U	●●-	3	●●●--	kérdőjel	●●--●●
D	-●●	M	--	Ü	●●--	4	●●●●-	aposztróf	●----●
E	●	N	-●	V	●●●-	5	●●●●●	kötőjel	-●●●●-
É	●●-●●	O	---	W	●--	6	-●●●●	egyenlőségjel	-●●●-
F	●●-●	Ö	---●	X	-●●-	7	--●●●	törtjel	-●●-●
G	--●	P	●--●	Y	-●--	8	---●●	zárójel	-●--●-
H	●●●●	Q	--●-	Z	--●●	9	----●	idézőjel	●-●●-●

Hibajel: folyamatosan leadott legalább 6 pont

Feladat

1. Mindenkiről tartanak nyilván adatokat, ezek közül több is számkód, például a naplóban lévő sorszám is ilyen számkód. Gyűjts hasonlókat!
2. A leveleken, képeslapokon, postai küldeményeken a település irányítószámát is fel kell tüntetni.
   a) Mit gondolsz, miért?
   b) Mi a településetek irányítószáma?
   c) A postai irányítószámok listája alapján kódold át számkódokra a következő települések neveit: Pilisvörösvár, Gyula, Szeged, Dunaújváros.
3. Hány bitre lenne szükség az angol ábécé 26 betűjének titkosításához, ha azonos hosszú bitsorokat használunk?

Számítógépek matematikája

Számrendszerek

Számok írása

A számokat a különböző számrendszerekben számjegyekkel ábrázoljuk. A napjainkban használatos számrendszerek helyiértékes rendszerűek, de nem mindig volt ez így: gondoljunk csak a római számírásra! Helyiértékes számrendszer esetén a szám értéke a számon belül elfoglalt helyétől is függ. A számrendszer azoknak a jeleknek és elveknek az összessége, amely alapján a számot felírjuk, elolvassuk.

A mindennapi életben általános a tízes számrendszer használata, természetes, hogy tízféle számjegyet használunk, és a tíz, a száz, az ezer „kerek" szám. Ezekkel a számokkal könnyű számolni. Ennek történeti oka valószínűleg abban rejlik, hogy éppen tíz ujj van a kezünkön, és a számoláshoz az ujjak mindig „kéznél vannak".

Azonban nem csak tízes számrendszer létezik. A lehetséges számrendszerek száma tulajdonképpen végtelen, de csak néhány használata vált szükségszerűvé. A számítástechnikában más (bináris, oktális, hexadecimális) számrendszereket is alkalmaznak. A számrendszer alapszáma (alapja, bázisa) a rendelkezésre álló számjegyek száma, a helyiértékes számrendszer elnevezése tehát az alapjának megfelelően történik.

Egy *A* alapú számrendszerben *A* darab számjegy létezik, 0-tól (*A*-1)-ig. Például a tízes (decimális) számrendszer alapja 10, mert a számok ábrázolására legfeljebb 10 számjegyet használunk, 0-9-ig. A helyiértékes számrendszerben a következő alakban írható fel egy szám:

$sz = m * b^n$, ahol sz: a szám
m: a mantissza
b: az alap
n: a kitevő

A szám melletti alsó indexben jelöljük a számrendszer alapját. Kivételt képeznek a tízes számrendszerbeli számok, ezek esetén a jelölés elmaradhat, ha

ez nem okoz félreértést. (A hétköznapi életben például nem használjuk a jelölést.)

Tízes (decimális) számrendszer

A számrendszer alapja 10. Egy helyiértéken 10 különböző számjegy írható, ezek: 0, 1, 2, 3, 4, 5, 6, 7, 8, 9.

Egy decimális szám normálalakja általánosan: $sz = m \cdot 10^n$

Példa egy tízes számrendszerbeli számra: 567, melyet így olvasunk: „ötszázhatvanhét”.

Minden helyiérték tíz különböző hatványainak felel meg. Jobbról balra haladva: egyesek, tízesek, százasok (ezresek, tízezresek, …)

Kettes (bináris) számrendszer

A számrendszer alapja 2, tehát egy helyiértéken 2 különböző számjegy fordulhat elő, ezek a 0 és az 1.

Egy bináris szám normálalakja általánosan: $sz = m \cdot 2^n$

Példa egy kettes számrendszerbeli számra: 11001_2, melyet így olvasunk: „egy, egy, nulla, nulla, egy, kettes számrendszerben”.

Itt a helyiértékek 2 hatványainak felelnek meg, jobbról balra haladva: $2^0 = 1$, $2^1 = 2$, $2^2 = 4$, $2^3 = 8$, $2^4 = 16$, $2^5 = 32$, $2^6 = 64$, $2^7 = 128$, $2^8 = 256$, ...

Nyolcas (oktális) számrendszer

A számrendszer alapja 8, így egy helyiértéken 8 különböző számjegy fordulhat elő, ezek: 0, 1, 2, 3, 4, 5, 6, 7.

Egy oktális szám normálalakja általánosan: $sz = m \cdot 8^n$

Példa egy nyolcas számrendszerbeli számra: 3075_8, melyet így olvasunk: „három, nulla, hét, öt, nyolcas számrendszerben”.

A helyiértékek 8 hatványainak felelnek meg, tehát jobbról balra haladva $8^0 = 1$, $8^1 = 8$, $8^2 = 64$, $8^3 = 512$, $8^4 = 4096$, $8^5 = 32768$…

Tizenhatos (hexadecimális) számrendszer

A számrendszer alapja 16, azaz egy helyiértéken 16 különböző számjegy fordulhat elő. Eszerint 16 különböző jelre van szükségünk. Mivel a 10 féle decimális számjegy nem elegendő, kellenek kiegészítő számjegyek is, ezek az

ábécé betűi lesznek. Így a hexadecimális számrendszerben előforduló számjegyek: 0, 1, 2 , 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

Egy hexadecimális szám normálalakja általánosan: $sz = m \cdot 16^n$

Példa egy tizenhatos számrendszerbeli számra:

$2B5C_{16}$, melyet így olvasunk: „kettő, bé, öt, cé tizenhatos számrendszerben”. Hexadecimális számrendszer esetén használhatjuk a $ jelzést a szám előtt, esetleg H vagy h betűt is: $2B5C = H2B5C = h2B5C.

A helyiértékek 16 hatványainak felelnek meg, jobbról balra haladva: $16^0 = 1$, $16^1 = 16$, $16^2 = 256$, $16^3 = 4096$, $16^4 = 65536$…

A táblázat 2, 8, 10, 16 első tíz hatványát mutatja:

n	2^n	8^n	10^n	16^n
0	1	1	1	1
1	2	8	10	16
2	4	64	100	256
3	8	512	1 000	4 096
4	16	4 096	10 000	65 536
5	32	32 768	100 000	1 048 576
6	64	262 144	1 000 000	16 777 216
7	128	2 097 152	10 000 000	268 435 456
8	256	16 777 216	100 000 000	4 294 967 296
9	512	134 217 728	1 000 000 000	68 719 476 736
10	1 024	1 073 741 824	10 000 000 000	1 099 511 627 776

A legkisebb értelmezhető számrendszer a bináris számrendszer, hiszen az egyes számrendszer gyakorlati haszna megkérdőjelezhető, ugyanis 1-nek minden hatványa is saját maga lesz.

A számrendszerek közti átváltás

Konvertálás decimális számmá

Az átalakítás módszerének alapelve az, hogy az átalakítandó szám számjegyeit megszorozzuk a megfelelő helyiértékek tízes számrendszerbeli alakjával, és az így kapott számokat összeadjuk.

a) átalakítás bináris számrendszerbeli alakról

A bináris szám számjegyeit megszorozzuk a megfelelő helyiértékek tízes számrendszerbeli alakjával, és az így kapott számokat összeadjuk.

Példa: $11001_2 = 1 \cdot 2^4 + 1 \cdot 2^3 + 0 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = 16 + 8 + 0 + 0 + 1 = 25$

b) átalakítás oktális számrendszerbeli alakról

Az oktális szám számjegyeit megszorozzuk a megfelelő helyiértékek tízes számrendszerbeli alakjával, és az így kapott számokat összeadjuk.

Példa: $3075_8 = 3 \cdot 8^3 + 0 \cdot 8^2 + 7 \cdot 8^1 + 5 \cdot 8^0 = 3 \cdot 512 + 0 + 7 \cdot 8 + 5 \cdot 1 =$

$= 1536 + 0 + 56 + 5 = 1597$

c) átalakítás hexadecimális számrendszerbeli alakról

A hexadecimális szám számjegyeit megszorozzuk a megfelelő helyiértékek tízes számrendszerbeli alakjával, és az így kapott számokat összeadjuk.

Példa: $2B5C_{16} = 2 \cdot 16^3 + B \cdot 16^2 + 5 \cdot 16^1 + C \cdot 16^0 =$

$= 2 \cdot 4096 + 11 \cdot 256 + 5 \cdot 16 + 12 \cdot 1 = 8192 + 2816 + 80 + 12 = 11100$

d) átalakítás tetszőleges A alapú számrendszerbeli alakról

A szám számjegyeit jelöljük *B*-vel, mint tömbbel, melynek elemeire így hivatkozhatunk: *B[N] B[N-1] ... B[2] B[1] B[0]*, ahol *N* a legmagasabb fokszámú elem helyiérték sorszáma, másképpen fogalmazva: a legmagasabb fokszámú tag együtthatójának az indexe. (*N* értéke eggyel kevesebb, mint ahány számjegye a számnak van.) A szám konvertálása tízes számrendszerbe *A* alapú számrendszerből tehát a következő képletnek megfelelően történhet:

$$SZ = \sum_{i=0}^{n} B[i] \cdot A^i$$

Például $417_8 = B[0] \cdot 8^0 + B[1] \cdot 8^1 + B[2] \cdot 8^2 = 7 \cdot 1 + 1 \cdot 8 + 4 \cdot 64 = 271_{10}$

Ahhoz, hogy algoritmizálhassuk a számítást, az ún. Horner-elrendezés nyújt segítséget: a szám első számjegyét megszorozzuk a számrendszer alapjával, a következő jegyet hozzáadjuk, majd újra az alapszámmal szorozzuk az eredményt. Ezt addig folytatjuk, amíg a számjegyek el nem fogynak. Az utolsó számjegy hozzáadása után már nem kell szoroznunk.

Lássuk, mi a magyarázata annak, hogy „működik” ez a módszer. Horner egy polinom helyettesítési értékének kiszámítására a következő összefüggést szerkesztette meg:

$$p = \sum_{i=0}^{n} a_i \cdot x^i = a_0 \cdot x^0 + a_1 \cdot x^1 + a_2 \cdot x^2 + \ldots + a_{n-1} \cdot x^{n-1} + a_n \cdot x^n$$

Ez éppen megfelel az általunk fogalmazottaknak, ilyen formában:

$$SZ = \sum_{i=0}^{N} B[i] \cdot A^{i} = B[0] \cdot A^{0} + B[1] \cdot A^{1} + ... + B[N-1] \cdot A^{N-1} + B[N] \cdot A^{N},$$

mely alakot céljainknak megfelelően átrendezve a következőt kapjuk.

$$SZ = B[N] \cdot A^{n} + B[N-1] \cdot A^{n-1} + ... + B[1] \cdot A^{1} + B[0] \cdot A^{0}$$

Az utolsó tag kivételével emeljünk ki minden tagból A-t:

$$SZ = \left(B[N] \cdot A^{n-1} + B[N-1] \cdot A^{n-2} + ... + B[2] \cdot A^{1} + B[1]\right) \cdot A + B[0] \cdot A^{0}$$

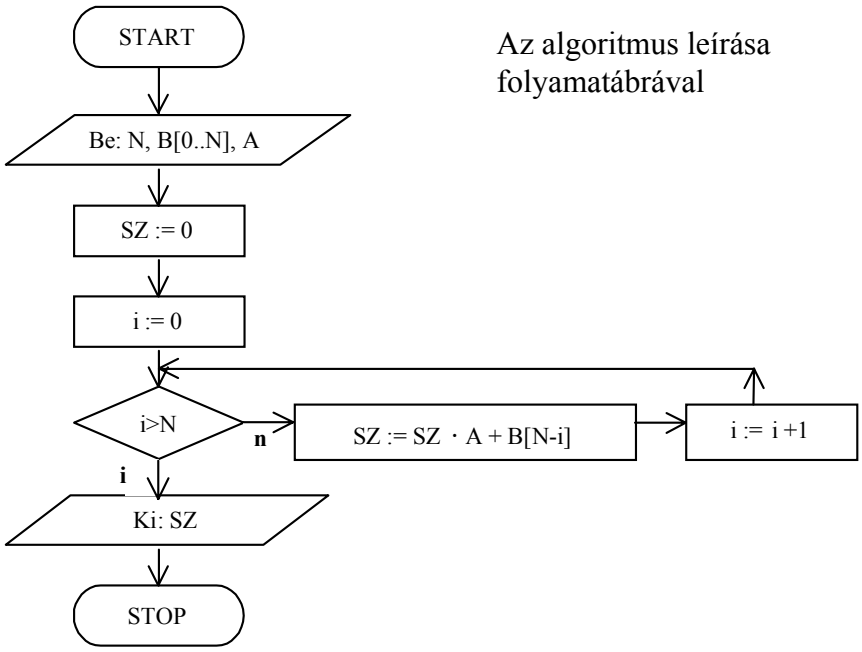
Így eggyel kisebb fokszámú polinomot kaptunk. Ezt a módszert folytathatjuk addig, amíg ilyen alakot nem kapjuk:

$$SZ = \left(\left(B[N]\right) \cdot A + B[N-1]\right) \cdot A + ...$$

Az algoritmus elkészítéséhez ismernünk kell

a kezdőértéket: SZ = 0

bármely tag kiszámításának szabályát: SZ = SZ · A + B[N-i], ahol i = 1..N



Az algoritmus leírása folyamatábrával

Decimális számrendszerbeli szám konvertálása

a) bináris számrendszerbe

Egy tízes számrendszerbeli egész szám kettes számrendszerbe való átváltása tehát a következő lépésekben történik: a számot elosztjuk 2-vel, a maradék vagy 1, vagy 0 lesz. A 2-vel való osztást addig végezzük – miközben a maradékokat felírjuk –, amíg a hányados nulla nem lesz. A keletkező maradékokat fordított sorrendbe (alulról felfele, balról jobbra) leírva kapjuk a bináris alakot.

Például a 35 decimális szám átalakítása kettes számrendszerbeli számmá:

35 : 2 = 17,maradék: **1**
17 : 2 = 8,maradék: **1**
8 : 2 = 4,maradék: **0**
4 : 2 = 2,maradék: **0**
2 : 2 = 1,maradék: **0**
1 : 2 = 0,maradék: **1**

$35_{10} = 100011_2$

b) oktális számrendszerbe

A számot ebben az esetben 8-cal osztjuk, mert a számrendszer alapja nyolc. A maradék 0, 1, 2, 3, 4, 5, 6, 7 lehet. Az osztást ismét addig végezzük, amíg a hányados nulla nem lesz, közben a maradékokat feljegyezzük. A keletkező maradékokat most is fordított sorrendben (alulról felfele, balról jobbra) kell leírnunk ahhoz, hogy megkapjuk az oktális alakot.

Például a 2003 decimális szám átalakítása nyolcas számrendszerbeli számmá:

2003 : 8 = 250,maradék: **3**
250 : 8 = 31,maradék: **2**
31 : 8 = 3,maradék: **7**
3 : 8 = 0,maradék: **3**

$2003_{10} = 3723_8$

c) hexadecimális számrendszerbe

A tízes számrendszerbeli egész számot elosztjuk 16-tal, hiszen a hexadecimális számrendszer alapja 16. A maradék lehet 0, 1, 2 , 3, 4, 5, 6, 7, 8, 9, A(=10), B(=11), C(=12), D(=13), E(=14), F(=15). A 16-tal való osztást addig végezzük, amíg a hányados nulla nem lesz, közben a maradékokat felírjuk. Az hexadecimális alakot most is fordított sorrendbe (alulról felfele, balról jobbra) leírva kapjuk.

Például a 4779 decimális szám átalakítása tizenhatos számrendszerbeli számmá:

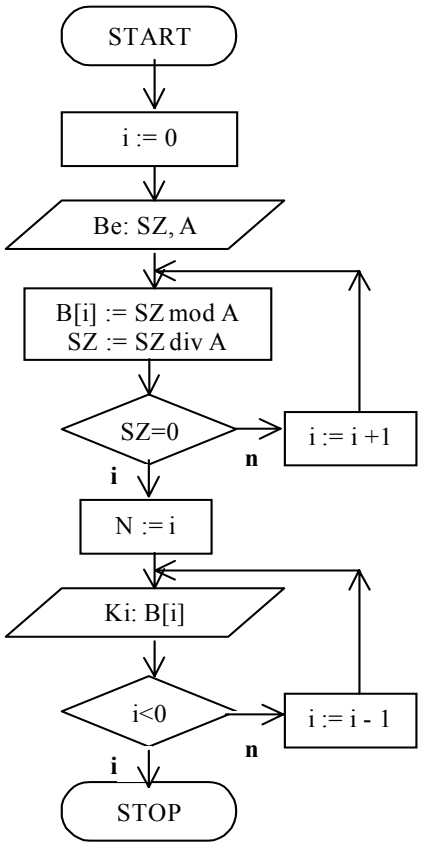
4779 : 16 = 298,maradék: **B** (11)
298 : 16 = 18,maradék: **A** (10)
18 : 16 = 1,maradék: **2**
1 : 16 = 0,maradék: **1**

$4779_{10} = 12AB_{16}$

d) tetszőleges A alapú számrendszerbe

A tízes számrendszerben felírt egész szám átalakítása (átváltása) egy *A* alapú formára úgy történik, hogy a decimális számot *A*-val ismételten osztjuk mindaddig, amíg a hányados 1-nél kisebbé nem válik. Az osztások során kapott maradékok az *A* alapú számrendszerben egy-egy számjegyet jelentenek. Ügyelnünk kell azonban arra, hogy a kapott új számjegyeket fordított sorrendben kell olvasni.

Az algoritmus folyamatábrával:



Az első 16 természetes szám alakja a különböző számrendszerekben:

decimális	**bináris**	**oktális**	**hexadecimális**
0	0	0	0
1	1	1	1
2	10	2	2
3	11	3	3
4	100	4	4
5	101	5	5
6	110	6	6
7	111	7	7
8	1000	10	8
9	1001	11	9
10	1010	12	A
11	1011	13	B
12	1100	14	C
13	1101	15	D
14	1110	16	E
15	1111	17	F

Minél nagyobb a számrendszer alapja, annál tömörebben leírható a szám [4]:

számrendszer	alap	szám	számjegyek száma
bináris	2	11 000 000 111 001	14
oktális	8	361 100	6
decimális	10	123 456	6
hexadecimális	16	1E 240	5

Felmerülhet egy igen gyakorlatias kérdés: milyen arány állapítható meg a bináris és a decimális számrendszerbeli helyiérték-szükséglet esetén?

Kiindulásként fontoljuk meg a következőt: N bináris jeggyel ki tudjuk fejezni mindazokat a számokat, amelyek a $0 - 2^N-1$ zárt intervallumba esnek. Hasonlóképpen n decimális jeggyel mindazokat, amelyek a $0 - 10^n-1$ zárt intervallumban találhatók. Ha az N bináris jeggyel leírható legnagyobb szám ugyanakkora, mint az n decimális jeggyel leírható legnagyobb, akkor felírhatjuk az egyenlőséget: $2^N-1 = 10^n-1$

Ebből $N \cdot \lg 2 = n$

kifejezve N-et: $N = \frac{n}{\lg 2} = \frac{n}{0{,}301} \approx 3{,}3n$

Ez azt jelenti a számunkra, hogy egy adott szám leírására a bináris számrendszerben megközelítőleg 3,3-szer több jegy (helyiérték) szükséges, mint a decimális számrendszerben. [2]

További átalakítások

a) $A \rightarrow A^k$, *ahol* $k \in N^+$

Az átalakítás alapelve, hogy *k* hosszúságú számjegycsoportokat képezünk az átalakítandó számból, és ezeket a csoportokat egyenként számítjuk át az új alapra.

- Bináris számrendszerben megadott szám átalakítása oktális számrendszerbeli számmá

  Az átalakítás hármas számjegycsoportok (bit-hármasok, vagy triádok) képzésével történik Ennek magyarázata a számrendszerek alapjai közötti összefüggésben rejlik, mely szerint az oktális számrendszer alapszáma éppen egyenlő a bináris számrendszer alapszámának köbével: $8 = 2^3$. A csoportosítást jobbról, a legkisebb helyiértéktől kezdjük, az utolsó csoportban szükség szerint nullával pótolva:

  Példa: $\mathbf{10110011_2} = (010_2)(110_2)(011_2) = (2_{10})(6_{10})(3_{10}) = \mathbf{263_8}$

- Bináris számrendszerben adott szám átváltása hexadecimális számrendszerbeli számmá

  Az elv hasonló, most azonban négyes számjegycsoportok (bit-négyesek vagy tetrádok) képzésével végezzük az átalakítást, hiszen a hexadecimális számrendszer alapszáma a bináris számrendszer alapszámának negyedik hatványával egyenlő: $16 = 2^4$. A csoportosításra vonatkozó szabályok ugyanazok.

  Példa: $\mathbf{10110011_2} = (1011_2)(0011_2) = (11_{10})(3_{10}) = (B_{16})(3_{16}) = \mathbf{B3_{16}}$

b) $A^k \rightarrow A$, *ahol* $k \in N^+$

Az átalakítás lényege, hogy minden számjegyet külön-külön át kell számítani az új alapra, *k* pozícióra.

- Oktális számrendszerben megadott szám átváltása bináris számrendszerbeli számmá

  Minden számjegyet egyenként 3 pozícióra írunk át.

  Példa: $\mathbf{567_8} = (101_2)(110_2)(111_2) = \mathbf{101110111_2}$

- Hexadecimális számrendszerben megadott szám átváltása bináris számrendszerbeli számmá

  A számjegyek átírása ebben az esetben 4 pozícióra történik. Mivel a hexadecimális számrendszerben tíztől a számjegyeket az ábécé betűivel

jelöljük, segíthet, ha legelőször decimális számmá alakítjuk a hexadecimális számot.

Példa: $\mathbf{A7_{16}} = (A_{16})(7_{16}) = (10_{10})(7_{10}) = (1010_2)(0111_2) = \mathbf{10100111_2}$

A következő táblázat a 431_{10} szám alakját mutatja a különböző számrendszerekben. Jól látható, hogy a bináris szám jegyeit hármasával csoportosítva ($2^3 = 8$!) egyszerűen előállíthatjuk az oktális számrendszerbeli alakot, míg a négyes csoportosítás ($2^4 = 16$!) a hexadecimális átváltást könnyíti meg.

oktális	0			6			5			7		
bináris	0	0	0	1	1	0	1	0	1	1	1	1
hexadecimális	1			A			F					

Az átalakítások egyéb esetekben a tízes számrendszeren keresztül történhetnek.

Feladat

1. Rakd sorba a kettes számrendszerbe való váltás lépéseit!
   a) a hányados átveszi a szám szerepét;
   b) leírjuk a kapott maradékokat jobbról balra haladva;
   c) a maradékot leírjuk a szám mellé;
   d) ha a szám nagyobb nullánál, akkor vissza az első pontba;
   e) a számot osztjuk kettővel;
   f) a hányadost leírjuk a szám alá;
2. Végezd el a 108 decimális szám átváltását kettes, nyolcas és tizenhatos számrendszerbe!
3. Váltsd át a születési évedet bináris, oktális és hexadecimális számrendszerbe!
4. A Naprendszerben a bolygók számának megadásához hány számjegy szükséges
   a) decimális számrendszerben?
   b) oktális számrendszerben?
   c) bináris számrendszerben?
5. Írd át kettes számrendszerből tízesbe, nyolcasba, tizenhatosba a következő számokat!
   a) 111101
   b) 11111111
   c) 1010010101
   d) 11001101101
6. Az alábbi táblázat kettes, tízes és tizenhatos számrendszerben megadott számokat tartalmaz. Egészítsd ki a táblázatot úgy, hogy az egy sorba írt számok azonos mennyiségeket jelöljenek!

Kettes	Tízes	Tizenhatos
	18	
1001		
	22	
		2F
1101		
		14

7. Alakítsd át a következő számot bináris, oktális és decimális számmá: $A0F_{16}$
8. Figyeld meg a következő számot: 3018
   a) Milyen számrendszerbeli szám lehet ez a szám?
   b) Milyen számrendszerbeli szám *nem* lehet ez a szám?

Tört számok konvertálása

Nem esett még szó a törtszámokról. Törtszámok csak tízes számrendszerben léteznek? Természetesen nem. A nem egész számok ábrázolásakor a számjegyek sorát egy elválasztójel (vessző vagy pont) két részre osztja. Az elválasztójeltől balra lévő számjegyek az egészek, ezek alkotják a szám egészrészét, míg a tőle jobbra lévő számjegyek a törtrészt jelentik.

Ahogyan eddig a helyiértékek a számrendszer alapszámának megfelelő hatványok szerint alakultak, ugyanúgy igaz ez a törtrész esetén is. A következő táblázat ezt szemlélteti:

A alapú számrendszer	…	$\mathbf{A^2}$	$\mathbf{A^1}$	$\mathbf{A^0}$	**.**	$\mathbf{A^{-1}}$	$\mathbf{A^{-2}}$	$\mathbf{A^{-3}}$	…
Decimális számrendszer	…	10^2	10^1	10^0	**.**	10^{-1}	10^{-2}	10^{-3}	…
Bináris számrendszer	…	2^2	2^1	2^0	**.**	2^{-1}	2^{-2}	2^{-3}	…

Egy **vegyes szám** (olyan tört, amelynek 0-tól különböző egész része is van) átírása tízes alapról *A* alapra két alapvető feladatrészre bontható: külön történik az egész rész és külön a törtrész átírása.

$128.32_{10} = ?_A$

Az előzőekben már láttuk, hogyan történik egy egész szám átalakítása, így a feladat leegyszerűsödik, és a kérdést így fogalmazhatjuk meg: hogyan kell **valódi tört**et (olyan tört, amelynek az egészrésze 0) tízes alapról *A* alapra átírni?

$0.32_{10} = ?_A$ $\quad 0 < T < 1$, ahol T a valódi törtet jelöli

Felvetődnek az alábbi kérdések az átalakítás megkezdése előtt:

– A valódi tört az *A* alapú számrendszerben is valódi tört lesz-e?

A válasz: igen, mert a szám értéke nem változik attól, hogy a konverzió megtörtént!

– Előfordulhat-e, hogy az elválasztójel után végtelen sok számjegynek kellene következnie?

A válasz: igen. Lehet, hogy véges törtet kapunk, de előfordulhat, hogy végtelen szakaszos törtet kapunk. Ezért is fontos, hogy az átalakítás előtt tisztázott legyen, hogy hány jegyig kell kiszámítani a törtet (*N*). Az *A* alapra alakított tört azonban racionális szám lesz.

A valódi tört felírható a következő formában:

$$T_{10} = B[1]\cdot A^{-1} + B[2]\cdot A^{-2} + B[3]\cdot A^{-3} + ...$$

Szorozzuk meg az egyenlőség mindkét oldalát *A* –val!

$$T_{10}\cdot A = B[1] + B[2]\cdot A^{-1} + B[3]\cdot A^{-2} + ...$$

egészrész törtrész a továbbiakban

A továbbiakban újra szorozzuk *A* –val a törtrészt, és ezt folytatjuk, amíg a „végére” nem értünk, illetve el nem értük a kívánt *N* értéket.

Példa: 0.3125_{10} átalakítása bináris számmá:

törtrész

0.3125	·2 =	0.625	egészrésze:	**0**
0.625	·2 =	1.25	egészrésze:	**1**
0.25	·2 =	0.5	egészrésze:	**0**
0.5	·2 =	1.0	egészrésze:	**1**
0				

A számot megszorozzuk az új számrendszer alapjával, kettővel. Az eredmény egészrészét leírjuk. Az újabb szorzásban csak az eredmény törtrésze vesz részt. Ebben az esetben véges törtet kaptunk az átalakítás során. Az átalakítást addig tudtuk folytatni, amíg a törtrész 0 nem lett.

Fontos, hogy az olvasás iránya ebben az esetben eltérő az eddigiektől: fentről le, jobbról balra, valamint az, hogy ez a számsor a törtrészt jelenti, az egészrész pedig nulla. $0.3125_{10} = 0.0101_2$

Példa: a 0.29 decimális valódi tört átalakítása kettes számrendszerbe, 8 jegy pontossággal:

törtrész

0.29	·2 =	0.58	egészrésze:	**0**
.58	·2 =	1.16	egészrésze:	**1**
.16	·2 =	0.32	egészrésze:	**0**
.32	·2 =	0.64	egészrésze:	**0**
.64	·2 =	1.28	egészrésze:	**1**
.28	·2 =	0.56	egészrésze:	**0**
.56	·2 =	1.12	egészrésze:	**1**
.12	·2 =	0.24	egészrésze:	**0**
.24				

Látható, hogy még nem jutottunk az átírás „végére”, 8 jegy után befejeztük az átalakítást. $0.29_{10} = 0.01001010_2$

Feladat

1. Váltsd át a következő tízes számrendszerbeli valódi törteket bináris számrendszerbe!
   a) 0,75
   b) 0,375
   c) 0,285
2. Váltsd át a következő tízes számrendszerbeli törteket bináris számrendszerbe!
   a) 5,5
   b) 25,125
   c) 89,67

Adatok ábrázolása a számítógépben

A számítógép minden műveletet számokkal végez. Felvetődik a kérdés: hogyan történik az adatok és a programok tárolása – azaz ábrázolása – a számítógépben?

A feldolgozás ideje alatt az adatok a gépben tárolódnak, ezért a gépben való tárolás belső adatábrázolással történik. [2] A témánk tehát az, hogyan történik a számítógépben a belső adatábrázolás.

Az adatok belső ábrázolása többféle lehet. Másképpen ábrázoljuk a szövegeket (alfanumerikus jelsorozatokat) és másképp a számokat, sőt, a számértékek is többféle módon jelenhetnek meg a számítógépben.

A számítógép az információkat **kétállapotú elemek** sorozatával képes tárolni. Olyan kódrendszerre van tehát szükség, amelyben az alkalmazott jelek száma éppen kettő. Adódik tehát a következtetés: a kódrendszer alapját a kettes számrendszer biztosítja majd. A kettes számrendszerben való ábrázolás azért fontos, mert jól fel lehet használni az adatok számítógépben való ábrázolásának fizikai megvalósításához. A kettes számrendszernek viszont van hátránya is: a tízes számrendszerhez képest egy szám bináris alakja igen hosszú jelsorozattal írható le. Emiatt használatosak még a nyolcas (oktális) és a tizenhatos (hexadecimális) számrendszerek is.

Bit, byte, szó

A kettes számrendszerben ábrázolt adatok alapegysége a **bit**, az angol Binary digIT kifejezés rövidítéseként. A kettes számrendszer egy-egy számjegye 1 bitet jelent, tehát a bit a bináris alakban megadott adatok legkisebb egysége. Egy bit a 0 vagy az 1 bináris értéket tárolhatja. Ez meglehetősen kicsi mennyiség, ezért ennek többszörösét célszerű egy egységként kezelni: egy **byte** (ejtsd: bájt) 8 bit együttesének felel meg. (A

kettes számrendszer használata miatt a bitek helyi értékének figyelembevételével lehetséges nagyobb egész számot ábrázolni.)

Az adatok tárolására alkalmas legkisebb elérhető (címezhető) tárolóelem a tárolócella. A **szó** több ilyen tárolócellának egy új egységgé való összefogását jelenti. A címezhető egységek lehetnek byte-ok, de vannak 16 és 32 bites számítógépek is.

1 bájt = 8 bit — 2 bájt = 16 bit (félszó)

4 bájt = 32 bit (szó) — 8 bájt = 64 bit (dupla szó)

A byte mértékegység többszöröseinek jelölésére használható a kilo-, mega-, giga-, tera- előtag, de mást jelent, mint általában: a számítástechnikában 1024 bájtot nevezünk 1 kilobájtnak, mégpedig azért, mert a 2 hatványai közül a 2^{10}=1024 áll legközelebb az 1000-hez. A különbséget jelöljük írásban is a kilo-előtag esetén, ahol K (és nem k) a rövidítés. [4]

2^{10} byte = 1 Kbyte (kilobájt) — 2^{20} byte = 1 Mbyte (megabájt)

2^{30} byte = 1 Gbyte (gigabájt) — 2^{40} byte = 1 Tbyte (terabájt)

Előjel nélküli egész számok ábrázolása

Az egész számok ábrázolása a kódolás különleges esetét jelenti. A számítógép az egész számot először az általa alkalmazott számrendszerbe alakítja át, majd ezt követi a bináris jelekké alakítás. Műszaki okokból a legtöbb számítógép meghatározott számú bitet (állandó szóhosszúságot) használ az ábrázoláshoz. (Ha a szám ennél rövidebb, nullákkal egészíti ki balról a gép az adott szóhosszúság eléréséhez.) Az ábrázolás egyenes kódolással történik, azaz az értékek kettes számrendszerbeli alakja felel meg a biteknek.

Ábrázolás 1 bájton

A legkisebb ábrázolható szám: $0000\ 0000_2 = 0_{10}$

0	0	0	0	0	0	0	0

A legnagyobb ábrázolható szám: $1111\ 1111_2 = 255_{10}$ (2^8-1 = 256 – 1 = 255)

1	1	1	1	1	1	1	1

Összesen 2^8, azaz 256 különböző szám ábrázolható 1 bájton.

Ábrázolás 2 bájton

A legkisebb ábrázolható szám: $0000\ 0000\ 0000\ 0000_2 = 0_{10}$

A legnagyobb ábrázolható szám: $1111\ 1111\ 1111\ 1111_2 = 65535_{10}$

(2^{16}-1 = 65536 – 1 = 65535)

Összesen 2^{16}, azaz 65536 különböző szám ábrázolható 1 bájton.

Példa: az 1993 decimális szám ábrázolása 2 bájton

$1993_{10} = 111\ 1100\ 1001_2$, ábrázoláskor pótolni kell balról nullákkal, így:

0	0	0	0	0	1	1	1	1	1	0	0	1	0	0	1

(A helyközök egyébként csak az olvasás megkönnyítésére használatosak, a számítógépnek erre nincs szüksége, ezért a tárolóban a jelek helyköz nélkül követik egymást.)

Előjeles egész számok ábrázolása

A pozitív számokat a bináris alakjuk ábrázolja. Hogyan lehetne a negatív számokat megkülönböztetni, ábrázolni?

A megoldást az **előjelbit** alkalmazása jelenti, mely 0, ha pozitív számot, 1, ha negatív számot ábrázolunk. A negatív számok ábrázolásához azonban ez még nem elég: ábrázolásukra dolgozták ki a **kettes komplemens** módszerét. (Szokás más néven a nullára történő kiegészítés módszerének nevezni, hiszen egy szám és kettes komplemensének összege éppen 0!)

Egy bináris szám kettes komplemensének kialakítása a következő lépésekben történik:

1. a szám minden bitjét az ellenkezőjére változtatjuk, vagyis minden 0 számjegyet 1-gyel, és minden 1 számjegyet 0-val helyettesítünk. Így megkapjuk az *egyes komplemens*t.
2. az egyes komplemenshez +1-et hozzáadunk. Így a *szám kettes komplemens*éhez jutunk. (Az összeadás műveletének pontos leírását a megfelelő fejezet tartalmazza!)

Példa: az 10111_2 szám kettes komplemensének kiszámítása

az alapszám:	10111
egyes komplemens:	01000
+1 hozzáadása:	1
a kettes komplemens:	01001

(Egy „gyors” módszer a kettes komplemens előállítására: jobbról balra haladva amíg 0-t találunk, leírjuk, az elsőnek megtalált 1-t is változatlanul leírjuk, majd innentől fogva minden számjegyet „átbillentünk” az ellenkezőjére. Érdemes kipróbálni!)

Ábrázolás 1 bájton

A legkisebb ábrázolható szám: $1000\ 0000_2 = -128_{10}$

1	0	0	0	0	0	0	0

Gondoljuk meg: az első bitnek muszáj 1-nek lennie, hiszen negatív szám esetén az előjel-bit értéke 1. A továbbiakban pedig akkor kapunk minimális értéket, ha minden helyiértékre 0-t írunk.

A legnagyobb ábrázolható szám: $0111\ 1111_2 = 127_{10}$

0	1	1	1	1	1	1	1

Hasonlóképpen gondolkodhatunk: az első bitnek 0-nak kell lennie, mert pozitív szám esetén az előjel-bit értéke 0. A továbbiakban pedig akkor kapunk maximális értéket, ha minden helyiértékre 1-et írunk.

Most is igaz, hogy összesen 2^8, azaz 256 különböző szám ábrázolható 1 bájton.

Példa: -5 decimális szám ábrázolása 1 bájton

1. lépés: +5 ábrázolása: $+5_{10} = 00000101_2$

0	0	0	0	0	1	0	1

2. lépés: kettes komplemens képzése:

az alapszám:	00000101
egyes komplemens:	11111010
+1 hozzáadása:	1
a kettes komplemens:	11111011

tehát $-5_{10} = 1111\ 1011_2$

Hogyan ellenőrizhető ez az eredmény? Tudjuk azt, hogy a szám és kettes komplemensének összege éppen 0. Ezt az összefüggést használjuk fel az ellenőrzéshez:

az alapszám	00000101
+kettes komplemens	+11111011
az eredmény	(1)00000000

A 9. helyiértéken kapott szám az ún. túlcsordulás. Ez a módszer az előjelet kódoló helyen megjelent átvitelt nem veszi figyelembe, így valóban 0-t kaptunk az összeadás eredményeként.

Példa: Melyik számot ábrázoltuk?

1	0	0	0	0	0	1	1

Tudjuk, hogy negatív szám, ezt az előjelbitből állapítható meg. A kérdés úgy fogalmazható meg, hogy melyik számnak a kettes komplemens ez? Ha egy $+A$ szám kettes komplemense $-A$, akkor $-A$ szám komplemense $-(-A)$! Lássuk tehát ennek a számnak a kettes komplemensét:

az alapszám:	10000011
egyes komplemens:	01111100
+1 hozzáadása:	1
a kettes komplemens:	01111101

és mivel $01111101_2 = 125_{10}$, a keresett szám -125.

Ellenőrzés:

az alapszám (-125)	10000011
+kettes komplemens (+125)	01111101
az eredmény	(1)00000000

Valóban 0-t kaptunk az összeadás eredményeként (az előjelet kódoló helyen megjelent átvitelt nem véve figyelembe).

Ábrázolás 2 bájton

A legkisebb ábrázolható szám: $1000\ 0000\ 0000\ 0000_2 = -32768_{10}$

A legnagyobb ábrázolható szám: $0111\ 1111\ 1111\ 1111_2 = 32767_{10}$

Összesen tehát 2^{16}, azaz 65536 különböző szám ábrázolható 1 bájton.

Törtszámok ábrázolása – lebegőpontos ábrázolás

A nem egész számok (valós számok) ábrázolására alkalmazott módszer a lebegőpontos számábrázolás. Mivel a számítógépek csak véges hosszúságú számjegysorozatokkal képesek dolgozni, a valós számokat egy közelítő értékkel helyettesítve racionális számokká kell alakítani.

A bináris lebegőpontos ábrázolás minden számot szorzat alakban ad meg, így egy lebegőpontos szám (floating point number) a következő alakban írható fel:

$z = m \cdot A^k$, ahol z: a lebegőpontos szám,

m: a mantissza

A: az alkalmazott számrendszer alapja

k: a kitevő (karakterisztika)

bináris lebegőpontos szám esetén tehát: $z = m \cdot 2^k$ [4]

A mantissza és a kitevő is lehet negatív szám.

A számoknak ez a leképzése gyakran közelítő eredményt ad, mert előre rögzített, hogy a mantissza hány számjegyet tartalmazhat.

Példa: $12.43_{10} = 1010.01010111\ldots_2$. Ha azonban a mantissza csak 8 jegy hosszúságú lehet, akkor ebből csak a következő vehető figyelembe: 1010.0101_2, pedig ennek a számnak decimális számrendszerben 12.3125 az értéke.

Normáltnak vagy normalizáltnak nevezzük mantisszát, ha a következő előírásnak eleget tesz: $1/A \le m < 1$ és $m \neq 0$

tízes számrendszer esetén: $1/10 \le m < 1$ és $m \neq 0$

példa: $12.34 = 0.1234 \cdot 10^2$

bináris mantissza esetén tehát: $1/2 \le m < 1$ és $m \neq 0$

példa: $1010.0101 = 0.10100101 \cdot 2^4$

A legkisebb mantissza értéke tehát ezek után 0.1_{10} illetve 0.1_2.

Az ábrázolás során a számítógépnek nem kell tárolnia a számrendszer alapját, hiszen minden számítás azonos alapú számrendszerben ábrázolt számokkal történik. A számítógép meghatározott számú tárolóhelyet biztosít mind a mantissza, mind pedig a kitevő számára, ezeket már tárolni kell.

A mantissza egy valódi tört, melynek **ábrázolása** történhet kettes komplemens alapján.

A kitevő ábrázolása azonban legtöbbször **feszített** módban (többletes kód segítségével) történik. Ekkor k-t a lebegőpontos szám **karakterisztiká**jának nevezzük. A karakterisztika adja meg, hogy a mantissza törtpontját hány helyiértékkel kell áthelyezni. A feszített módban történő ábrázolás azt jelenti, hogy a számhoz hozzáadunk egy többletet, amelyet a következő módon határozunk meg: többlet = 2^{n-1}, ahol n azt jelenti, hogy hány bit áll

rendelkezésre a karakterisztika ábrázolására. Például, ha n = 8, akkor $2^7 = 128$-cal kell a számot növelni. Erre azért van szükség, mert a kitevő lehet negatív szám is, és így a negatív kitevő is ábrázolhatóvá válik anélkül, hogy előjelét külön kellene ábrázolni.

A lebegőpontos ábrázoláshoz gyakran 4 bájtot használnak, de léteznek 6, 8, 10 bájtos lebegőpontos számábrázolások is. A mantissza és a karakterisztika sorrendje nincs megkötve a bájton. Az ábrázolható értéktartomány függ attól, hogy hány biten történik a mantissza, illetve a karakterisztika ábrázolása.

A lebegőpontos számot ábrázolhatjuk 4 bájton, azaz 32 biten. Ekkor 1 bit az előjelbit, amely a mantissza előjelét jelenti, 8 bit a kitevő, 23 bit pedig a mantissza tárolására szolgál.

Példa: 11.8125 decimális szám ábrázolása bináris lebegőpontos számábrázolással

1. lépés: a szám átalakítása kettes számrendszerbeli számmá

   $11.8125_{10} = 1011.1101_2$

2. lépés: binárisan normált alakra kell hozni a számot, vagyis meg kell állapítani

   a) a mantisszát: figyelembe véve, hogy $1/2 \leq m < 1$ és $m \neq 0$

      m = 0.10111101

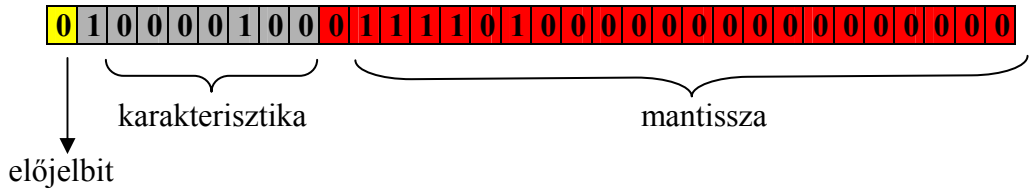
      Valójában az ábrázolás során szükségtelen ábrázolni a törtpont előtti és utáni számjegyet, hiszen a normalizált alak minden esetben ugyanígy kezdődik.

   b) a kitevőt: mivel a törtpontot négy helyiértékkel kellett áthelyezni balra, ezért a kitevő $4_{10} = 100_2$

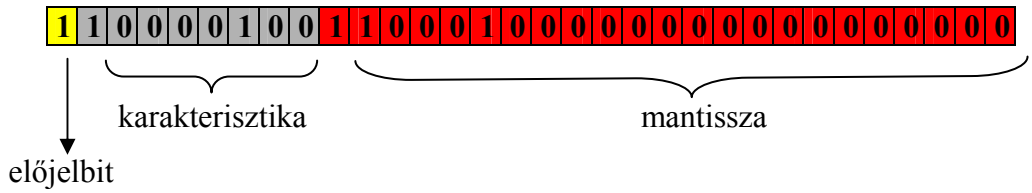
   c) a karakterisztikát: feszített módban a karakterisztika úgy alakul, hogy – mivel 8 biten ábrázoljuk a karakterisztikát – a kitevőhöz hozzáadunk 2^7-t, azaz 128-at, ami 10000000_2-t:

kitevő	100
többlet	+ 10000000
karakterisztika	= 10000100

A szám binárisan normált alakban tehát 32 biten ábrázolva:



Példa: $-12.25_{10} = 1100.01_2$ ábrázolása:



A számábrázolás pontossága a mantissza hosszúságával növekszik. [4]

Például a – 0.16022 szám esetén:

mantissza hosszúsága		
5	00010	$= -0{,}12500_{10}$
6	000101	$= -0{,}15625_{10}$
7	0001010	$= -0{,}15625_{10}$
8	00010100	$= -0{,}15625_{10}$
9	000101001	$= -0{,}16016_{10}$
10	0001010010	$= -0{,}16016_{10}$
…	…	…
16	0001010010000010	$= -0{,}16019_{10}$

Feladat

1. Váltsd át a következő decimális számokat bináris számrendszerbeli számmá, majd ábrázold előjel nélküli egész számként, 1 bájton.
   a) 58 b) 147 c) 254
2. Váltsd át a következő decimális számokat bináris számrendszerbeli számmá, majd ábrázold előjel nélküli egész számként, 2 bájton.
   a) 32558 b) 9874 c) 648211
3. Váltsd át a következő decimális számokat bináris számrendszerbeli számmá, majd ábrázold előjeles egész számként, 1 bájton.
   a) – 101 b) 25 c) - 99
4. Váltsd át a következő decimális számokat bináris számrendszerbeli számmá, majd ábrázold előjeles egész számként, 2 bájton.
   a) – 3556 b) 8245 c) - 999
5. Váltsd át kettes normál alakra a következő tízes számrendszerbeli törteket:
   a) 9.25 b) –17.5625
6. Váltsd át a következő tízes számrendszerbeli törteket bináris számrendszerbe, majd ábrázold 32 biten lebegőpontos számként!
   a) 5.5 b) 158.32

Fixpontos ábrázolás

A fixpontos szám (fixed point number) helyiértékes számrendszerben ábrázolt számjegysorozat. [4]

A fixpontos ábrázolásnál a bináris pont fix helyen van. Ez általában az utolsó pozíció utáni helyet jelenti. A számítógép a fixpontos számokat egyszerű számjegysorozatként tárolja. A bináris pont helye a deklarációból derül ki, és mivel mindig ugyanazon a helyen van, ezért ezt külön tárolni nem szükséges. Legtöbb esetben az egész számok ábrázolására használják, de nem egész számok ábrázolására is alkalmas. Előjeles fixpontos számok esetén az első bitet (balról az elsőt, azaz a legmagasabb helyiértékűt) előjelbitnek tekintjük.

A fixpontos ábrázolás előnye, hogy a műveletvégzés sebessége nagyobb, mint lebegőpontos számok esetén, hátránya, hogy az ábrázolási tartomány kisebb.

Karakterek ábrázolása

Mindaddig, amíg a számítógépekkel kizárólag számítási feladatokat végeztek, az adatok számok, számjegyek voltak. Ezek ábrázolására a fixpontos és a lebegőpontos ábrázolás elegendő volt. A számítógépek azonban olyan adatfeldolgozási feladatok elvégzésére is alkalmasak, amelyeknél az adatok karakterek, így kívánatossá vált olyan ábrázolási módszerek kialakítása, melyekkel a karakterek ábrázolásának problémája megoldható.

A számítógép által használt teljes jelkészletet alfanumerikus jeleknek, röviden **karakter**eknek nevezzük. *Az adatok tehát nem csupán számok, számjegyek lehetnek, hanem betűk, műveleti és írásjelek, speciális jelek is.* (Megállapodás szerint az ún. vezérlőkaraktereket, mint például a lapemelés (FF, Form Feed), nem számítjuk az alfanumerikus jelek közé.) [4]

Szám vagy karakter?

Fontos, hogy egy számjegyet, vagy egy számjegyekből álló jelsorozatot mikor tekintünk valóban számnak, és mikor karaktersorozatnak. A számot az különbözteti meg a számjegyekből álló karaktersorozattól, hogy a számokkal számítási (aritmetikai) műveleteket végzünk, míg a karaktereket, még ha számjegyek is, nem kívánjuk számítási műveletekben felhasználni.

Például a 2085 számjegyek számot alkotnak, vagy karaktersorozatot? Ha ez egy termék egységára, amivel további számításokat szeretnénk végezni, akkor számként kell kezelünk. Ha ez egy település irányítószáma, valószínűleg számítási műveletben nem fog szerepelni, így karaktersorozatként fogjuk kezelni. Mindez befolyásolja az ábrázolás módját. A következő ábrázolási módok során *a számjegyekre is karakterekként tekintünk*!

Kódrendszerek

A jelkészlet kódolására több féle módszer is született az idők folyamán. Az alapelv az, hogy az egyes karakterekhez (betűkhöz, speciális jelekhez, számjegyekhez) hozzárendelnek egy meghatározott bitkombinációt.

Azt, hogy hány bitből áll egy ilyen kombináció, az határozza meg, hogy a jelkészlet hány karakterből áll, vagyis hány eltérő karaktert kell az ábrázolás során megkülönböztetnünk.

Ha például 4 bit áll rendelkezésre, akkor mindössze $2^4 = 16$ különböző jelet ábrázolhatunk, 5 bit esetén már $2^5 = 32$ féle, tehát kétszer annyi eltérő jelet.

A táblázat a lehetséges kombinációk számát mutatja be a bitszám függvényében [2]:

bitek száma	lehetséges bitkombinációk száma
1	2
2	4
3	8
4	16
5	32
6	64
7	128
8	256
…	…
k	2^k

Ezek szerint a tíz számjegy ábrázolására elegendő lenne 4 bit, mert $2^4 = 16$ féle jel ábrázolására ennyi bőven elegendő. (3 bit viszont kevés volna, hiszen $2^3 = 8$ féle jelet lehetne ábrázolni.) Az angol ábécé 26 betűjéhez, ha csak a nagybetűket vesszük figyelembe, $2^5 = 32$ bit szükséges. Ugyanennyi kell a kisbetűkhöz. Ekkor még nem is esett szó a speciális jelekről, és vegyük észre azt is, hogy így el is tékozoltunk néhány bitet…

A BCD kód

A jelkészlet kódolására régen igen elterjedt módszer volt a BCD kód (Binary Coded Decimal Code = binárisan kódolt decimális kód). Ez egy hatbites kódrendszer, tehát $2^6 = 64$ különféle karaktert képes ábrázolni.

A számjegyek esetén éppen a bináris alak felel meg a kódolt alaknak. Ezt követi az angol ábécé nagybetűi, majd a speciális karakterek 111111-ig. [2]

kódolandó karakter	BCD kód	kódolandó karakter	BCD kód
0	000000	A	001010
1	000001	B	001011
2	000010	C	001100
3	000011	D	001101
4	000100	E	001110
5	000101	F	001111
6	000110	…	…
7	000111		
8	001000		
9	001001		

Ezt a kódrendszert olyan számítógépek alkalmazták, melyben a szóhossz 24 bit volt, egy szóban 4 karakter fért el.

Például a 2371 szám karakteres alakja BCD kódrendszerben a következő:

2	3	7	1
000010	000011	000111	000001

A BCD kódban ábrázolt számokkal azonban műveletet végezni nem lehet!

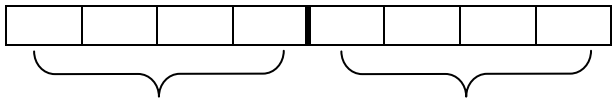
Érezhető, hogy ez a kódrendszer igen hamar „kinőhető”, hiszen szükséges lehet számos egyéb karakter ábrázolására is.

Az EBCDIC kód

A BCD kód kiterjesztett változata, elnevezése (Extended Binary Coded Decimal Interchange Code = kiterjesztett BCD kód) is erre utal. Az EBCDIC

egy olyan kódkészlet, amely szöveg, grafika, és vezérlőkarakterek ábrázolását teszi lehetővé a számítógépeken. Főleg az IBM nagyszámítógépei alkalmazzák. Nyolcbites kód, ezért $2^8 = 256$ különböző bitkombinációt tesz lehetővé. Úgy tűnhet, hogy ez igen bő tartomány, erre nincs is szükség a jelkészlet kódolásához. A kódképzés szabálya azonban merőben más, mint a BCD kódrendszer esetén: itt a kódrendszer megalkotói nem egyszerű hozzárendelést választottak, nem egymás után, a sorszámuknak megfelelően kódoltak a karakterek.

Tulajdonképpen egy 16 oszlopból és 16 sorból álló táblázatról van szó. A nyolc bitet felosztották két négybites hosszúságú részre. Az első 4 bit az ún. zónarész. a második 4 bit pedig a számrész. Vegyük észre, hogy 4 biten éppen egy tizenhatos számrendszerbeli szám ábrázolható.



zóna vagy oszlop száma 0..15 (0..F)

sor száma 0..15 (0..F)

A következő táblázat a 16x16-os táblázat egy részét mutatja be [2]:

SZÁM \ ZÓNA	0	6	7	8	9	A	B	C	D	E	F
0		-									0
1		/		a	j			A	J		1
2				b	k	s		B	K	S	2
3				c	l	t		C	L	T	3
4				d	m	u		D	M	U	4
5				e	n	v		E	N	V	5
6				f	o	w		F	O	W	6
7				g	p	x		G	P	X	7
8				h	q	y		H	Q	Y	8
9				i	r	z		I	R	Z	9
A		^	:								
B		’	#								
C		%	@								
D		–	|								
E			=								
F		?	„								

A táblázatból kiolvasható, hogy az egyes karaktereknek mely zóna- illetve számrész felel meg. Az oszlopszámot és a sorszámot is tizenhatos számrendszerbeli számként olvassuk le. A táblázatban:

– a kisbetűk a 8.-A. oszlopokban vannak, az 1. sorral kezdődően;
– a nagybetűk a C.-E. oszlopokban vannak, szintén az 1. sorral kezdődően;
– a számjegyek az F. oszlopban vannak az 0. sorral kezdődően;
– vannak „hézagok” is.

Példa: A „BIT” szó karaktereinek ábrázolása során a bitsorozat így alakul:

| C2 | C9 | E3 |

míg a „bit” karaktersorozaté:

| 82 | 89 | A3 |

az „ADAT” karaktersorozaté:

| C1 | C4 | C1 | E3 |

a „45”, (azaz négy – öt) karaktersorozat esetében:

| F4 | F5 |

Egy 32 bites szóban éppen 4 bájt tárolható – hosszabb szövegek esetén több szóban kell tárolni a szöveget. [2]

Zónázott vagy zónás decimális ábrázolás – EBCDIC kódrendszer esetén

Észrevehető, hogy minden számjegy ábrázolásakor a zónarészbe F (1111). Így minden számjegy ábrázolásához 1 byte, vagyis 8 bit szükséges. Ilyen módon az előjel nélküli számokat ábrázoljuk – vagy úgy is fogalmazhatunk, hogy a pozitív számokat (bár ne feledjük, hogy ez karaktersorozat, nem pedig számítási műveletben részt vevő szám!). Hogyan lehetne megkülönböztetni a negatív számokat? A megoldás: az előjeleknek megfelelő négybites kombinációkat alkalmazunk az ábrázolandó szám legalacsonyabb helyiértékű számjegyének megfelelő zónarészben. [2]

bináris alak	hexadecimális alak	előjel
1010	A	+
1011	B	–
1100	C	+
1101	D	–
1110	E	+
1111	F	+

Az 1100 és az 1101 előjelkódokat használva:

Példa: + 357 ábrázolása:

bináris:	1111	0011	1111	0101	**1100**	0111
hexadecimális:	F	3	F	5	**C**	7

– 941 ábrázolása:

bináris:	1111	1001	1111	0100	**1101**	0001
hexadecimális:	F	9	F	4	**D**	1

Tömörített (pakolt) decimális ábrázolás – EBCDIC kódrendszer esetén

Már az előzőekben is észrevettük, hogy minden számjegy ábrázolásakor a zónarészbe F (1111) kerül, és minden számjegy ábrázolásához 1 byte, vagyis 8 bit szükséges. Ha azonban csak számokat akarunk ábrázolni, ez a zónarész akár el is hagyható, hiszen minden számnál ugyanaz. Ezzel jelentősen spóroltunk, hiszen 4 bitet minden számjegy esetén megtakarítunk: bizonyos esetekben 8 biten így két számjegyet ábrázolhatunk egy helyett! Ez tehát egyfajta tömörítés, közel a felére csökkenthető a lefoglalt bitek száma.

Annak az oka, hogy nem éppen a felére csökken a lefoglalt bitek száma, az előjelek ábrázolása miatt van, hiszen ezeket nem hagyhatjuk el „csak úgy”, a az utolsó 4 biten jelölni kell az előjelet.

Példák:

a) 834 ábrázolása:

zónázott forma:	F8	F3	F4	= 3 bájt
tömörített alak:	83	4F		= 2 bájt

A tömörítendő szám páratlan számú számjegyet tartalmaz, így éppen egész számú bájtot foglal el a tömörített szám. Előjel nélküli ábrázolás.

b) 5179 ábrázolása:

zónázott forma:	F5	F1	F7	F9	= 4 bájt
tömörített alak:	05	17	5F		= 3 bájt

Az 5 előtti 0: feltölti az első 4 bitet nullával, ezzel egészítve ki egész számú bájtra. Előjel nélküli ábrázolás.

c) – 628 ábrázolása:

zónázott forma:	F6	F2	D8	= 3 bájt
tömörített alak:	62	8**D**		= 2 bájt

A leghátsó 4 biten megjelent az előjel. Páratlan a számjegyek száma, nincs szükség üres bitekre.

d) + 3809 ábrázolása:

zónázott forma:	F3	F8	F0	F9	= 4 bájt
tömörített alak:	03	80	9**C**		= 3 bájt

A leghátsó 4 biten megjelent az előjel. Páros a számjegyek száma, szükség van a legfelső 4 üres bitre [2].

Az ASCII kódrendszer

Egységesített kódrendszer, amelyben a különféle karakterekhez (betűk, számok, vezérlőkarakterek, írásjelek) bináris kódokat rendelnek. Elsősorban mikroszámítógépek esetén használják. Az ASCII egy mozaik szó: American Standard Code for Information Interchange (amerikai szabványos kód az információ kölcsönös cseréjére). A kódot az American Standard Institute dolgozta ki. Az ASCII kódrendszert 1977-ben az Amerikai Szabványügyi Hivatal megerősítése, és jóváhagyása után a Nemzetközi Szabványügy Hivatal (ISO) is átvette és ISO646 néven regisztrálta. Az ASCII 7 bites kód volt eleinte, így $2^7 = 128$ különféle bitsorozat létezett 0-tól 127-ig sorszámozva. Ez ún. alap karakterkészlet, vagy standard (az angol ábécé kis- és nagybetűi, számjegyek, írásjelek). A PC-k megjelenésekor az IBM által hozzáadott 1 bites kiterjesztéssel újabb 128 karakter használatát szabványosította, amely kódrendszer Latin1 néven ismert. A nyolcbites kóddá való kiegészítés tette lehetővé a nemzeti sajátosságokat is figyelembe vevő karakterkészlet kialakítását: 128-255 között az ún. kiegészítő karakterkészlet (számos európai nyelv – pl. francia, német spanyol, stb. – speciális nemzeti karakterei, ékezetes betűk, az angolban nem létező egyéb betűtípusok, vonalrajzoló, a görög ABC betűi, táblázatrajzoló karakterek, stb.) elemei találhatók. Ezeket nevezik kódlapoknak is, pl. Latin1, Latin2, 852-es kódlap, stb. Ez utóbbi, a 852-es a Magyar Szabványügyi Hivatal által is elfogadott kódlap, amely a teljes magyar karakterkészletet tartalmazza, így minden magyar nyelvű programnak ismernie „illik”. [7]

A standard felállítása óta ezek a karakterek az ASCII segítségével lehetővé teszik a számítógépek közötti kommunikációt.

A szabvány az ASCII karakterkészlet definiálásakor a kódokat két fő csoportba osztotta: grafikus karakterek és vezérlő karakterek csoportjába.

Grafikus karakterek alatt a megjeleníthető, látható, nyomtatható karaktereket értjük, míg a *vezérlő karakterek*, a megjelenítés vezérlésére, formájának kialakítására, valamint az információcsere vezérlésére szolgálnak. [21]

Az ASCII kódtábla

Dec	Hex	Kar	Dec	Hex	Kar	Dec	Hex	Kar	Dec	Hex	Kar
0	00	NUL	32	20	**SP**	64	40	@	96	60	Œ
1	01	SOH	33	21	!	65	41	A	97	61	a
2	02	STX	34	22	"	66	42	B	98	62	b
3	03	ETX	35	23	#	67	43	C	99	63	c
4	04	EOT	36	24	$	68	44	D	100	64	d
5	05	ENQ	37	25	%	69	45	E	101	65	e
6	06	ACK	38	26	&	70	46	F	102	66	f
7	07	BEL	39	27	Œ	71	47	G	103	67	g
8	08	BS	40	28	(	72	48	H	104	68	h
9	09	HT	41	29	)	73	49	I	105	69	i
10	0A	LF	42	2A	*	74	4A	J	106	6A	j
11	0B	VT	43	2B	+	75	4B	K	107	6B	k
12	0C	FF	44	2C	,	76	4C	L	108	6C	l
13	0D	CR	45	2D	-	77	4D	M	109	6D	m
14	0E	SO	46	2E	.	78	4E	N	110	6E	n
15	0F	SI	47	2F	/	79	4F	O	111	6F	o
16	10	DLE	48	30	0	80	50	P	112	70	p
17	11	DC1	49	31	1	81	51	Q	113	71	q
18	12	DC2	50	32	2	82	52	R	114	72	r
19	13	DC3	51	33	3	83	53	S	115	73	s
20	14	DC4	52	34	4	84	54	T	116	74	t
21	15	NAK	53	35	5	85	55	U	117	75	u
22	16	SYN	54	36	6	86	56	V	118	76	v
23	17	ETB	55	37	7	87	57	W	119	77	w
24	18	CAN	56	38	8	88	58	X	120	78	x
25	19	EM	57	39	9	89	59	Y	121	79	y
26	1A	SUB	58	3A	:	90	5A	Z	122	7A	z
27	1B	ESC	59	3B	;	91	5B	[	123	7B	{
28	1C	FS	60	3C	<	92	5C		124	7C	|
29	1D	GS	61	3D	=	93	5D	]	125	7D	}
30	1E	RS	62	3E	>	94	5E	^	126	7E	~
31	1F	US	63	3F	?	95	5F	-	127	7F	DEL

(Dec = decimális sorszám, Hex = sorszám hexadecimálisan, Kar = karakter)

A vezérlőkaraktereket három kategóriába soroljuk: információcsere vezérlők, formátumot befolyásolók és információ elkülönítők. Az első 32 karakter, és az utolsó DEL karakter tartozik ezekbe a kategóriákba.

Érdekesség, olvasmány

Információcsere vezérlő karakterre példa a 04H kódú EOT karakter, amit annak a jelzésére használnak, hogy a karakterek átvitele befejeződött és ez a kód jelöli, hogy nincs több átviendő karakter.
Formátum befolyásoló karakterekkel lehet a karaktersorozat megjelenési formáját befolyásolni. Például az LF (0AH) Line Feed (soremelés) karakter hatására a karakterek megjelenítése az adott pozícióban, új sorban folytatódik.
Az információ elkülönítő karakterek az információ logikai értelemben való elkülönítésére szolgálnak. Ilyen módon lehetséges különböző hosszúságú karaktersorozatok rekordok átvitele. Ha például három különböző hosszúságú rekordot akarunk átvinni, akkor a rekordokat a Rekord Separator (RS) (1EH) karakterrel lehet egymástól elválasztani. A vezérlőkarakterek némelyike a fentiek egyikébe sem sorolható be, ezeket általános vezérlőkaraktereknek nevezzük. A szabvány minden ASCII karaktert részletesen meghatároz. [21]

Minden ASCII vezérlőkarakter speciális feladat megvalósítására szolgál.

Vezérlőkódok jelentése [21]

NUL	NULL Character	Nulla (helykitöltő)	DC1	Device Control 1	általános vezérlőjel
SOH	Start of Heading	fejléc kezdete	DC2	Device Control 2	általános vezérlőjel
STX	Start of Text	szöveg kezdete	DC3	Device Control 3	általán vezérlőjel
ETX	End of Text	szöveg vége	DC4	Device Control 4	általános vezérlőjel
EOT	End of Transmission	adás vége	NAK	Negative Acknowledge	negatív nyugtázás
ENQ	Enquiry	kérés	SYN	Synchronous Idle	szinkronizáló jel
ACK	Acknowledge	elfogadás, nyugtázás	ETB	End of Transm. Block	egy blokk adás vége
BEL	Bell	hangjelzés	CAN	Cancel	érvény-telenítés
BS	Backspace	vissza-léptetés	EM	End of Medium	információ hordozó vége
HT	Horizontal Tabulation	vízszintes tabuláció	SUB	Substitute	helyettesítés
LF	Line Feed	soremelés	ESC	Escape	átkapcsolás
VT	Vertical Tabulation	függőleges tabuláció	FS	File Separator	fájlelválasztó
FF	Form Feed	lapdobás	GS	Group Separator	csoport elválasztó
CR	Carriage Return	kocsi-vissza	RS	Record Separator	rekord elválasztó
SO	Shift Out	kódváltás	US	Unit Separator	egység elválasztó
SI	Shift In	kód visszaváltás	SP	Space	szóköz
DLE	Data Link Escape	adat átkapcsolás	DEL	Delete	törlés

ASCII kódrendszer tehát minden karakterhez egy 0-255 intervallumban lévő számot rendel. Ezeket a sorszámokat nevezzük a karakterek ASCII kódjainak, melyek 1 bájtot foglalnak el a memóriában. Ezen bájt értéke az ASCII kód, képernyőn való megjelenítési formája pedig a karakter képe. [14]

Zónázott vagy zónás decimális ábrázolás – ASCII kódrendszer esetén

Figyeljük meg a számjegyeket az ASCII kódrendszerben:

számjegy:	0	1	2	3	4	5	6	7	8	9
decimális belső kód:	48	49	50	51	52	53	55	56	57	58
hexadecimális kód:	30	31	32	33	34	35	36	37	38	39

A számítógép a kódot hexadecimális formában rögzíti. Például az 164 szám (karaktersorozat, tehát egy-hat-négy) ASCII kódja alapján a következő módon kerül ábrázolásra:

ábrázolandó szám:	1	6	4
hexadecimálisan:	31	36	34

Zónás ábrázolásról beszélhetünk, ahol 1 bájton ábrázolunk egy számjegyet, az első 4 bit pedig a zóna. Az előjel nélküli – „pozitív” – számok ábrázolása nem jelent gondot, a probléma – akárcsak az EBCDIC kódrendszer esetén – a „negatív” számok megkülönböztetésével van. A megoldás is hasonló: a legalacsonyabb helyiértékű számjegy zónarészében jelöljük az előjelet, ami ebben az esetben **7** lesz.

Példa: – 543 ábrázolása:

ábrázolandó szám:	– 5	4	3
hexadecimálisan:	35	34	**7**3

Tömörített (pakolt) decimális ábrázolás – ASCII kódrendszer esetén

Hasonló megfontolások vezethetnek bennünket, mint az EBCDIC kódrendszer esetén. A különbség annyi, hogy az előjelbitet a legalacsonyabb számjegy zónarészében megtartjuk.

Példák:

a) + 543 ábrázolása:

zónázott forma:	35	34	33	= 3 bájt
tömörített alak:	54	**C**3		= 2 bájt

A tömörítendő szám páratlan számú számjegyet tartalmaz, így éppen egész számú bájtot foglal el a tömörített szám. Az utolsó számjegy zónarészében C a pozitív előjelet jelenti.

b) – 7102 ábrázolása:

zónázott forma:	37	31	30	32	= 4 bájt
tömörített alak:	07	10	**D**2		= 3 bájt

A legutolsó számjegy előtti 4 biten megjelent az előjel. Páros a számjegyek száma, szükség van a legfelső 4 üres bitre a 7 előtt.

A számítógépbe a kívülről bekerülő adatok mindig zónázott formájúak. [2] Azok az adatok, amelyekkel nem kívánunk számítási műveleteket végezni a továbbiakban, ilyen kódban maradhatnak. Azok a számadatok azonban, amelyekkel a továbbiakban különböző aritmetikai műveleteket szeretnénk végezni, átalakításra kell, hogy kerüljenek. A programozási nyelvektől függően lehetőség van bináris (fixpontos vagy lebegőpontos) átalakításra.

Feladat

1. Hány különböző karaktert képes ábrázolni legfeljebb
   a) egy kétbites kódrendszer
   b) egy hatbites kódrendszer
   c) egy hétbites kódrendszer
2. Hány különböző jel ábrázolható 1 bájton?
3. Írd fel a következő számok BCD kódbeli megfelelőjét!
   a) 23
   b) 144
   c) 369
4. Írd fel a következő számok EBCDIC kódbeli megfelelőjét!
   a) 52
   b) + 48
   c) – 36
5. Írd fel a következő számok tömörített ábrázolású megfelelőjét!
   a) 358
   b) – 254
   c) 99
6. A számítógépen az Alt billentyűt lenyomva tartva a numerikus billentyűzetről begépelt kódszám hatására a monitoron megjelenik a karakter. A táblázat segítségével próbáld megjeleníteni a következő magyar ékezetes betűket!

Á	**É**	**Í**	**Ó**	**Ö**	**Ő**	**Ú**	**Ü**	**Ű**
181	144	214	224	153	138	233	154	235
á	**é**	**í**	**ó**	**ö**	**ő**	**ú**	**ü**	**ű**
160	130	161	162	148	139	163	129	251

7. Írd le a neved ASCII számkóddal!

Műveletek bináris számokkal

Az adatok feldolgozása során különböző műveleteket, operációkat hajtunk végre. Azokat a mennyiségeket, amelyekre az operáció vonatkozik operandusoknak nevezzük. Az operandusokat műveleti jelek, operátorok kapcsolják össze.

Aritmetikai műveletek

Az aritmetikai operátorok operandusai számok, a műveletek eredménye is mindig szám lesz.

Egy operandusú (unáris) operátorok

1. **+ „plusz”** Olyan operáció, amely az utána következő operandust változatlanul hagyja. Az eredmény maga az operandus.
   Például: $+A$ művelet után A értéke változatlan.
2. **– „mínusz”** Ha az ábrázolás előjel nélküli, akkor az eredménye az operandus kettes komplemense, ha előjeles az ábrázolás, akkor az eredmény az operandus (–1) -szerese.
   (Ha el kell döntenünk, hogy $+a$ vagy $-a$ szám a nagyobb, addig semmit nem tudunk mondani, amíg nem ismerjük a értékét – hiszen nem értékről, hanem műveletről van szó!)
3. **bitenkénti negáció**: Olyan operáció, amely csak egész típusú operandus esetén alkalmazható. Minden bitet „átbillentünk” az ellenkezőjére.

Értéktáblázata:

NOT	**0**	**1**
	1	**0**

Példa: NOT 5, 1 bájton ábrázolva, előjel nélküli ábrázolásban:

5 =	0	0	0	0	0	1	0	1	
NOT 5 =	1	1	1	1	1	0	1	0	= 250

Vegyük észre, hogy éppen az egyes komplemenst kaptuk eredményül!

Két operandusú (bináris) operátorok

A bináris szó arra utal, hogy 2 operandus vesz részt ezekben a műveletekben. Az operátorok különböző típusúak is lehetnek, azonban ha az egyik operandus lebegőpontos, akkor az eredmény is az lesz.

1. **multiplikatív műveletek**

   a) *** szorzás**: az eredmény akkor és csakis akkor egész típusú, ha mindkét operandus egész típusú.

   Műveleti táblája:

*	0	1
0	**0**	**0**
1	**0**	**1**

   Példa: Számítsuk ki 5 · 6 értékét! A számokat ábrázoljuk 1 bájton!

```
0 0 0 0 0 1 0 1 * 0 0 0 0 0 1 1 0
  0 0 0 0 0 1 0 1
    0 0 0 0 0 1 0 1
    0 0 0 0 1 1 1 1 0                (= 30)
```

   A szorzás során a szorzandó jegyeit csak akkor írtuk le, ha a szorzó jegye 1. Ügyelni kell azonban a szorzó utolsó jegyére, ami ebben az esetben 0 volt, és bár a szorzásban külön nem jelöltük, az eredmény végére oda kellett tenni a legkisebb helyiértékre! (Hasonló ez ahhoz, mint amikor tízes számrendszerben a 35 · 20 szorzást úgy végezzük el, hogy 35-öt megszorozzuk 2-vel, majd a legkisebb helyiértékre a 0-t „teszünk”.)

   b) **/ osztás**: az eredmény mindig lebegőpontosan ábrázolt lesz az operandusoktól függetlenül. A nullával való osztás nem értelmezett!

   A most következő operációk csak egész operandusok esetén értelmezettek, eredményük is egész szám.

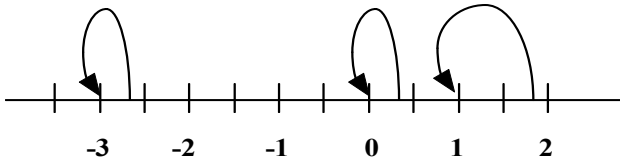
   c) **DIV**: szokás **egész osztás**nak is nevezni. Megértéséhez azonban szükséges néhány – gyakran félreértelmezett – kérdés, fogalom tisztázása.

   - *Mit értünk egy szám egész részén?*

   Egy *A* szám egész része az a legnagyobb egész szám, ami még nem nagyobb, mint *A*. Jelölés: [*A*]

A definíciót pontos megértését segíti a következő ábra:

[–2,7] = –3 [0,3] = 0 [1,8] = 1



- *Mit értünk egy szám törtrészén?*

Jelölje $\{A\}$ a szám törtrészét. Ekkor $\{A\} = A - [A]$

Példa: $\{-2,8\} = -2,8 - [-3] = 0,2$

- *Az előjelfüggvény*

$$\text{sign}(x) = \begin{cases} +1, \text{ ha } x > 0 \\ 0, \text{ ha } x = 0 \\ -1, \text{ ha } x < 0 \end{cases}$$

Mindezen ismeretek után a DIV műveletet a következő módon definiálhatjuk:

$$\text{A DIV B} = \text{sign}\left(\frac{\text{A}}{\text{B}}\right) \cdot \left[\frac{|\text{A}|}{|\text{B}|}\right]$$

Példák:

8 DIV 3 = (+1) · 2 = 2
–8 DIV 3 = (–1) · 2 = –2
8 DIV (–3) = (–1) · 2 = –2
–8 DIV (–3) = (+1) · 2 = 2

d) **MOD**: **maradékképzés**. Két egész operandus osztási maradékát képzi. Az eredmény is egész. (A rövidítés a modulus, maradék szóból ered)

Definíció szerint:

$$\text{A MOD B} = \text{A} - \text{A DIV B} \cdot \text{B}$$

Példák:

8 MOD 3 = 8 – 8 DIV 3 · 3 = 8 – 2 · 3 = 2
–8 MOD 3 = –8 – (–8) DIV 3 · 3 = –8 – (–2) · 3 = –2
8 MOD (–3) = 8 – 8 DIV (–3) · (–3) = 8 – (–2) · (–3) = 2
–8 MOD (–3) = –8 – (–8) DIV (–3) · (–3) = –8 – 2 · (–3) = –2

e) **AND: bitenkénti és**. Az két operandus és az eredmény is egész. Bitenként összehasonlítja az operandusokat a műveleti tábla szerint:

Értéktáblázata:

AND	0	1
0	0	0
1	0	1

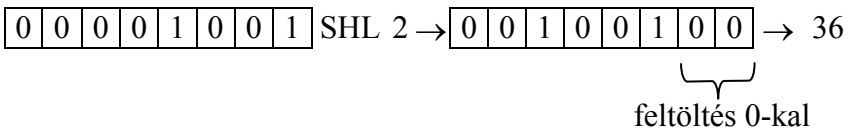
Példa: Mennyi az 5 AND 8 művelet eredménye? (Az operandusokat 1 bájton ábrázoljuk, negatív érték esetén kettes komplemens módban)

	0	0	0	0	0	1	0	1	= 5
AND	0	0	0	0	1	0	0	0	= 8
	0	0	0	0	0	0	0	0	= 0

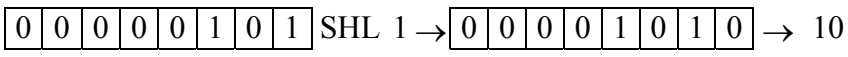
f) **SHL: bitenkénti eltolás balra** (shift left): A két operandus és az eredmény egész. Az első operandus bitjeit a második operandus értékével balra tolja, az üres helyeket 0-val feltölti.

Példák:

9 SHL 2



5 SHL 1

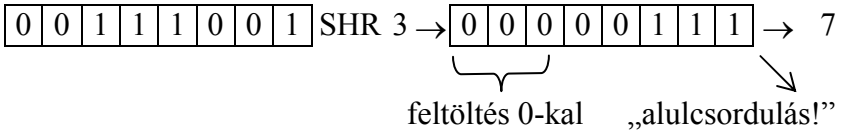


Vegyük észre, hogy éppen a szám kétszeresét kaptuk. Tehát *A* SHL 1 művelet az *A* szám 2-vel való szorzását jelenti. (Az állítás azonban túlcsordulás esetén nem igaz: balról „elveszítünk” számjegyet)

g) **SHR: bitenkénti eltolás jobbra** (shift right): A két operandus és az eredmény egész. Az első operandus bitjeit a második operandus értékével jobbra tolja, az üres helyeket 0-val feltölti.

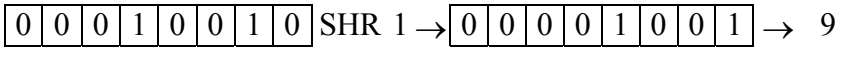
Példák:

57 SHR 3



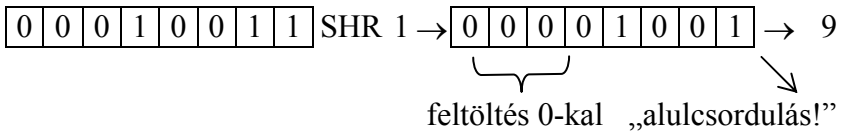
Jobbról „elveszítünk” számjegyet: ezt nevezik „alulcsordulásnak” [4].

18 SHR 1



Vegyük észre, hogy éppen a szám felét kaptuk. Tehát *A* SHR 1 művelet az *A* szám 2-vel való osztását jelenti. Ez az állítás azonban nem igaz alulcsordulás esetén, mert jobbról számjegyet „veszítünk”:

19 SHR 1



2. **Additív műveletek**

a) **+ összeadás**: az eredmény akkor és csakis akkor egész típusú, ha mindkét operandus egész típusú, egyébként lebegőpontos.

Műveleti táblája:

+	**0**	**1**
0	**0**	**1**
1	**1**	**0***

Az utolsó cellában a * azt jelzi, hogy az adott helyiértéken a leírandó 0 mellett a következő helyiértékre átviendő jegy is keletkezett, vagyis az átvitel 1.

Példa: 11 + 13 = ? A számokat 1 bájton ábrázoljuk előjel nélküli számként.

	0	0	0	0	1	0	1	1	= 11
+	0	0	0	0	1	1	0	1	= 13
	0	0	0	1	1	0	0	0	= 24

Az összeadás lépései tehát a következők:

- leírjuk a műveleti tábla alapján a két bit összegét
- az átvitel két 1-es bit esetén 1, egyébként 0

b) **– kivonás**: az eredmény akkor és csakis akkor egész típusú, ha mindkét operandus egész típusú, egyébként lebegőpontos.

Kivonás helyett képezzük a kivonandó kettes komplemensét, és ezt a kisebbítendőhöz hozzáadjuk. A módszer részletes ismertetése az *Előjeles egész számok ábrázolása* című fejezetben található.

Példa: 115 –109 = ?

A kisebbítendő:	0	1	1	1	0	0	1	1	= 115
A kivonandó:	0	1	1	0	1	1	0	1	= 109
A kivonandó kettes komplemense:	1	0	0	1	0	0	1	1	

A művelet:

	0	1	1	1	0	0	1	1	
+	1	0	0	1	0	0	1	1	
1	0	0	0	0	0	1	1	0	= 6

Az eredményben a túlcsordulást nem vesszük figyelembe.

c) **OR: bitenkénti vagy**. A két operandus és az eredmény is egész típusú. A két operandus bitjeit hasonlítja az értéktáblázat szerint:

Értéktáblázata:

OR	**0**	**1**
0	**0**	**1**
1	**1**	**1**

Példa: Mennyi a 21 OR 25 művelet eredménye? (Az operandusokat 1 bájton ábrázoljuk, negatív érték esetén kettes komplemens módban)

	0	0	0	1	0	1	0	1	= 21
OR	0	0	0	1	1	0	0	1	= 25
	0	0	0	1	1	1	0	1	= 29

d) **XOR: bitenkénti kizáró vagy**. A két operandus és az eredmény is egész típusú. A két operandus bitjeit hasonlítja a értéktáblázat szerint:

Értéktáblázata

XOR	**0**	**1**
0	**0**	**1**
1	**1**	**0**

Példa: Mennyi az 5 XOR 9 művelet eredménye? (Az operandusokat 1 bájton ábrázoljuk, negatív érték esetén kettes komplemens módban)

	0	0	0	0	0	1	0	1	= 5
XOR	0	0	0	0	1	0	0	1	= 9
	0	0	0	0	1	1	0	0	= 12

Aritmetikai kifejezés kiértékelése

Az aritmetikai kifejezés számértékkel rendelkező operandusok, aritmetikai operátorok és zárójelek összekapcsolásával állíthatók elő. [4] Az így keletkező kifejezések kiértékelésére a következő szabályok vonatkoznak:

I. szabály: Műveleti sorrend vagy precedencia: a műveletek nem egyenrangúak. A sorrend a következő:

1. unáris műveletek (+, –, NOT)
2. multiplikatív műveletek (*, /, DIV, MOD, AND, SHL, SHR)
3. additív műveletek (+, –, OR, XOR)

II. szabály: Azonos rangú operátorok esetén **balról jobbra** haladva végezzük el a műveleteket.

III. szabály: A **zárójelek** a műveletek szokásos sorrendjét módosíthatják, és eltérő sorrendet írhatnak elő. Ekkor előbb a zárójelben lévő kifejezést értékeljük ki.

Példa: Az operandusokat byte típuson ábrázoljuk, negatív érték esetén kettes komplemens módban

NOT 7 DIV 3 OR 4 DIV 2 = ?

1. 3. 2. 4.

1. lépés: NOT 7 = 248

7 =	0	0	0	0	0	1	1	1	
NOT 7 =	1	1	1	1	1	0	0	0	= 248

2. lépés: 248 DIV 3 = +1 · 82 = 82
3. lépés: 4 DIV 2 = +1 · 2 = 2
4. lépés: 82 OR 2

	0	1	0	1	0	0	1	0	= 82
OR	0	0	0	0	0	0	1	0	= 2
	0	1	0	1	0	0	1	0	= 82

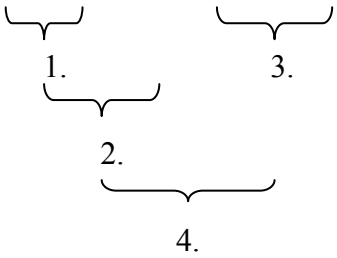
NOT 7 DIV 3 OR 4 DIV 2 = 82

Összehasonlító műveletek (relációk): $<, >, =, \leq, \geq, \neq$

Az összehasonlító műveletek NEM aritmetikai műveletek, hiszen eredményük nem szám, hanem logikai érték: false (hamis) vagy true (igaz).

Az ok, amiért mégis itt szerepelnek az, hogy gyakran előforduló művelet két olyan kifejezés összehasonlítása, melyek értéke szám. Az összehasonlítási műveletek a legutolsóként elvégzendők!

Példa: NOT 5 DIV 2 < 9 MOD 4 művelet eredményét keressük byte típuson:



1. lépés: NOT 5 = 250

5 =	0	0	0	0	0	1	0	1	
NOT 5 =	1	1	1	1	1	0	1	0	= 250

2. lépés: 250 DIV 2 = +1 · 125 = 125
3. lépés: 9 MOD 4 = 9 – 9 DIV 4 · 4 = 9 – 2 · 4 = 1
4. lépés: 125 < 1 → FALSE

NOT 5 DIV 2 < 9 MOD 4 → FALSE

Logikai műveletek

A logikai operátorok operandusai logikai értékek, a műveletek eredménye is mindig logikai érték lesz.

Kétfajta logikai értéket különböztetünk meg: FALSE (F, hamis) és TRUE (T, igaz.) A logikai értékek között igaz a következő reláció: F < T

1. **unáris operátor: NOT**: negáció. Ha az operandus értéke TRUE volt, akkor az eredmény FALSE lesz és fordítva.

igazságtáblája:

NOT	**F**	**T**
	T	**F**

2. **multiplikatív operátor: AND**: logikai és. Ha mindkét operandus TRUE, akkor az eredmény TRUE, minden más esetben FALSE.

igazságtáblája:

AND	**F**	**T**
F	**F**	**F**
T	**F**	**T**

3. **additív operátorok**
   a) **OR:** logikai vagy. Ha a két operandus közül legalább az egyik TRUE, akkor az eredmény TRUE, egyébként FALSE.

igazságtáblája:

OR	**F**	**T**
F	**F**	**T**
T	**T**	**T**

b) XOR: logikai kizáró vagy. Az eredmény TRUE, ha az operandusok értéke különböző, FALSE, ha azonos.

igazságtáblája:

XOR	F	T
F	F	T
T	T	F

A logikai műveletek kiértékelésére ugyanolyan a szabályok vonatkoznak, mint az aritmetikai kifejezésekre.

A következő példa azt mutatja be, hogy egy meggondolatlanul felírt feladat hogyan vezethet szemantikai hibához. (A szemantikai hiba értelmezési hibát jelent, szemben a szintaktikai hibával, amely egy adott nyelv szabályainak megsértésekor következik be.).

Példa: x < 3 OR x > 5 művelet eredménye mi lesz, ha x = 2?

2 < 3 OR 2 > 5

A feladat megoldásakor a már megismert kiértékelési szabályokat követjük: zárójelezés nem módosítja a műveletek szokásos sorrendjét. Unáris és multiplikatív művelet nincs. Az additív és összehasonlító műveletek közül (az OR most aritmetikai művelet, hiszen operandusai számok, nem pedig logikai értékek!), az additív műveletet kell először elvégeznünk: 3 OR 2, melynek eredménye 3, mert bitenként elvégezve az aritmetikai műveletet:

	0	0	0	0	0	0	1	1	= 3
OR	0	0	0	0	0	0	1	0	= 2
	0	0	0	0	0	0	1	1	= 3

Most már következhetnek az összehasonlító műveletek:

2< 3 > 5 ???? → olyan relációhoz jutottunk, amelynek három oldala van – ez hibás műveletsor!

El kell tehát gondolkodnunk, mi vezetett a problémához. Megvizsgálva a feladatot, hamar rábukkanhatunk a hiba okára: az OR művelet ez esetben ugyanis mint összehasonlítás művelet szerepel, így operandusainak logikai értékeknek kellene lenniük, nem pedig számoknak!! Zárójelezéssel a következő alakban már elvégezhető a művelet:

$$\underbrace{\underbrace{(2 < 3)}_{1.}\ \text{OR}\ \underbrace{(2 > 5)}_{2.}}_{3.}$$

Így logikai kifejezéseket kapcsolunk össze az OR logikai-vagy operátorral.

1. lépés: 2 < 3 reláció eredménye: TRUE

2. lépés: 2 > 5 reláció eredménye: FALSE

3. lépés: TRUE OR FALSE művelet eredménye: TRUE

A művelet eredménye tehát TRUE!

Feladat

1. Végezd el 1 bájtos, *előjeles* egészekkel, a következő műveleteket bináris számokkal! A végeredményt alakítsd át tízes számrendszerbeli számmá!
   0111 0111 + 0011 1001 = ? 0111 0000 – 0110 1100 = ? 1001 1111 + 0101 0101 = ?
2. Mit ad eredményül a következő kifejezés? NOT (12 OR 5) AND 11
3. Mi lesz az eredménye X-nek, ha A = 5, B = 13, C = 28?
   X = (A OR B) OR (NOT A AND C)

Érdekesség, olvasmány

A számítógép különleges, felépítésének megfelelő matematikai formalizmust használ, amely az algebra egyik részterülete, az ún. Boole-algebra. **George Boole** (1815-1864) **és Auguste de Morgan** (1806-1871) fejlesztette ki a matematikának ezt az ágát.[4] A Boole-algebrából levezetett logikai műveleteken kívül ismerkedjünk meg a következő alapvető összefüggésekkel:

Logikai azonosságok

Jelöljön *A* és *B* logikai típusú változókat, melyek értéke TRUE vagy FALSE lehet. Ebben az esetben a következő logikai azonosságok érvényesek:

Általános azonosságok:

A **AND** (**NOT** *A*)	→	eredménye mindig FALSE
A **OR** (**NOT** *A*)	→	eredménye mindig TRUE
A **AND** TRUE	→	eredménye *A*-tól függ: megegyezik *A* -val
A **AND** FALSE	→	eredménye mindig FALSE
A **OR** TRUE	→	eredménye mindig TRUE
A **OR** FALSE	→	eredménye *A*-tól függ: megegyezik *A* -val
NOT(**NOT** *A*)	→	a kétszeres tagadás eredménye maga az érték, vagyis *A*

De Morgan azonosságok:

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

Ezek szerint egy egyszerű, (azaz csak AND vagy csak OR operátort tartalmazó) logikai művelet komplementere megkapható, ha az AND/OR utasításokat OR/AND utasítással, a műveletben szereplő elemeket pedig komplementerükkel cseréljük fel. [4]

Irodalomjegyzék

[1] Dr Koncz József:
TECHNIKA – INFORMATIKA
Agria Kiadó Kft. 1990,
és Dr. Koncz József előadásai

[2] Csépai János:
A számítástechnika alapjai
Műszaki Könyvkiadó, Budapest, 1985.

[3] G. Cullmann, M. Denis-Papin, A, Kaufmann:
A hír tudománya – Az információelmélet alapjai
Gondolat, Budapest, 1973.

[4] Hans Breuer
SH atlasz – Informatika
Springer Hungarica Kiadó Kft., 1995

[5] Rozgonyi-Borus Ferenc
RAM-ba zárt világ (Számítástechnikai segédkönyv)
Mozaik Oktatási Stúdió, Szeged, 1997.

[6] Rozgonyi-Borus Ferenc
Számítástechnika 5-6
Mozaik Oktatási Stúdió, Szeged, 1998.

[7] Rozgonyi-Borus Ferenc
Számítástechnika 7
Mozaik Kiadó, Szeged, 2000.

[8] Rozgonyi-Borus Ferenc
Számítástechnika – Összefoglaló feladatgyűjtemény
Mozaik Kiadó, Szeged, 1997.

[9] Dr. Kovács Tivadar, Dr. Kovácsné Cohner Judit, Ozsváth Miklós, G. Nagy János:
Mit kell tudni a PC-ről?
ComputerBooks, Budapest, 1998.

[10] Dancsó Tünde – Végh András:
Számítástechnika 10-11 éveseknek

Műszaki Könyvkiadó, Budapest, 1997.

[11] Dancsó Tünde – Végh András:
Számítástechnika 11-12 éveseknek
Műszaki Könyvkiadó, Budapest, 1998.

[12] Végh András:
Számítástechnika 12-13 éveseknek
Műszaki Könyvkiadó, Budapest, 1997.

[13] Végh András:
Számítástechnika 13-14 éveseknek
Műszaki Könyvkiadó, Budapest, 2001.

[14] Dr. Koncz József:
A PASCAL programozási nyelv elmélete és gyakorlata
EKTF Líceum Kiadó, Eger, 1999.

[15] Schulcz Róbert
Számítástechnika – Munkatankönyv 6. osztályosoknak
Comenius Kiadó Kft., Nágocs, 2003.

[16] Schulcz Róbert
Számítástechnika – Munkatankönyv 7. osztályosoknak
Comenius Kiadó Kft., Nágocs, 2003.

[17] Schulcz Róbert
Számítástechnika – Munkatankönyv 8. osztályosoknak
Comenius Kiadó Kft., Nágocs, 2003.

[18] Pitrik József:
Tanári kézikönyv az Informatika, Könyvtárhasználat, Számítástechnika tankönyvsorozathoz
Apáczai Kiadó, Celldömölk, 1999.

[19] Pitrik József:
INFORMATIKA 5-6. – Könyvtárhasználat, számítástechnika (Tankönyv kezdők részére)
Apáczai Kiadó, Celldömölk, 1999.

[20] Pitrik József:
INFORMATIKA 7-8. – Könyvtárhasználat, számítástechnika (Tankönyv haladók részére)
Apáczai Kiadó, Celldömölk, 1999.

[21] *http://www.szabilinux.hu/konya/konyv/2fejezet/2fdigi2.htm*

[22] Bodnár István Olivér, Kiss Csaba, Krnács András:
Számítástechnikai alapismeretek A
Műszaki Könyvkiadó, Budapest, 2000.

[23] *http://www.neumann-centenarium.hu*

[24] *http://www.zsido.hu/kozosseg/Neumann.htm*

[25] *http://www.ffg.sulinet.hu/oktinfo98/oppe/index.htm*

[26] *http://www.scitech.mtesz.hu/10kiraly/kiraly_5.htm*

[27] *http://www.mta.hu/aktualis/kivonat/hirek_2003_02.html*

[28] Bonifert Zsolt:
Informatika
Műszaki Könyvkiadó, 2001.

[29] Turcsányiné Szabó Márta, Zsakó László:
Comenius Logo gyakorlatok
Kossuth Kiadó, 1997.

[30] *http://www.cic.klte.hu/iszk30/oktatas.htm*

[31] *http://www.cic.klte.hu/ttk50/inf.htm*

[32] *http://www.inflab.bme.hu/hist/oldies.html*

[33] *http://ttk.unideb.hu/ttk25/mtcs/szk.html*

[34] *http://www.sulinet.hu/eletestudomany/archiv/2001/0121/kronika/kronika.html*

[35] *http://www.hpo.hu/ipsz/200104/ evnaptar.htm*

[36] *http://www.mek.iif.hu/porta/szint/muszaki/szamtech/pc*

[37] *http://www.mek.iif.hu/porta/szint/tarsad/konyvtar/informat/jegyzet/html/index.html*

[38] *http://www.inlap.jate.u-szeged.hu/modtan/nyito.htm*