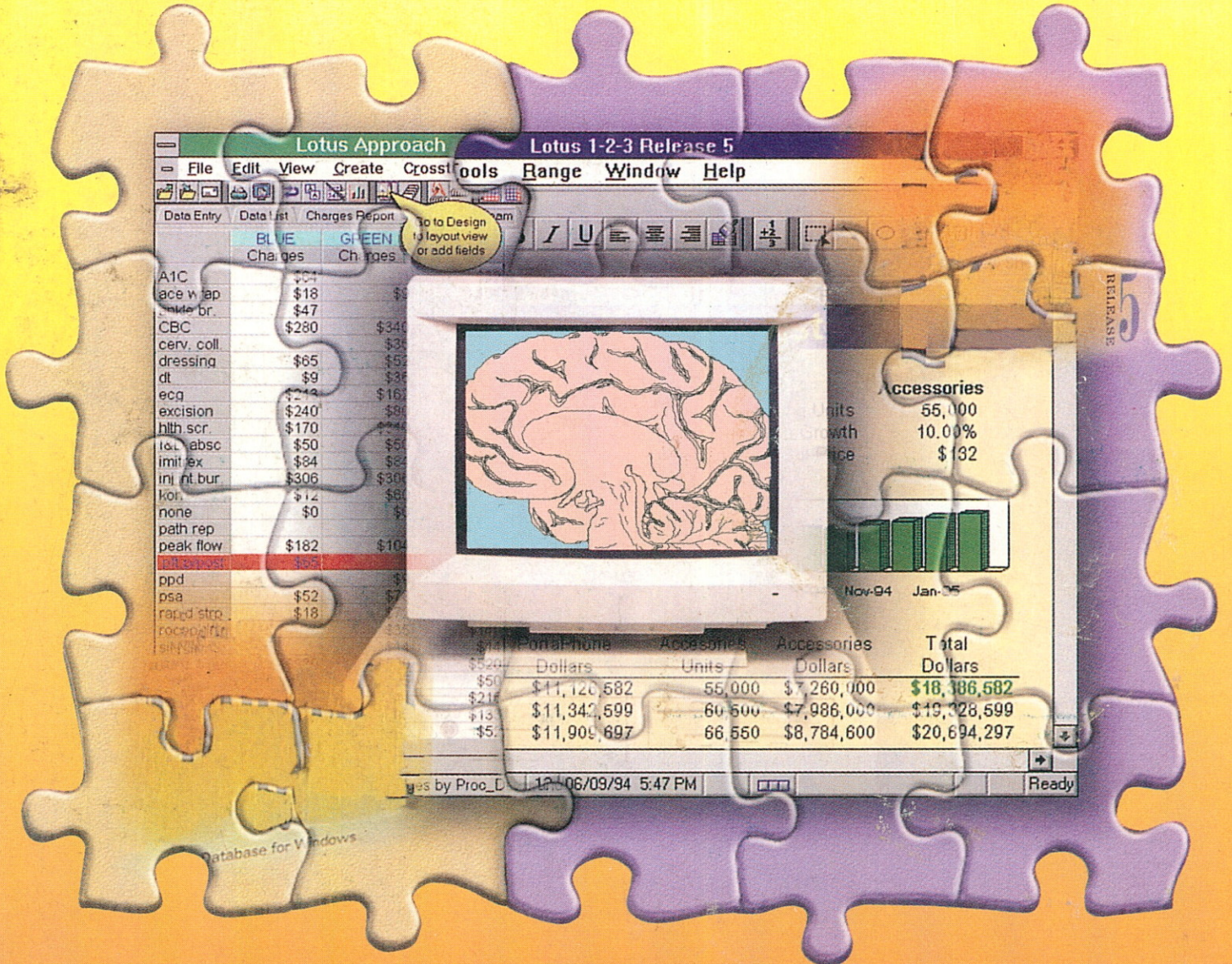
PROGRAMOZÁSI FELADATOK II.



KOSSUTH KIADÓ

PROGRAMOZÁSI FELADATOK II.

A KÖTETET ÖSSZEÁLLÍTOTTA

ZSAKÓ LÁSZLÓ

KOSSUTH KIADÓ 1997

A KÖTET SZERZŐI

BERCELINÉ SZTYEHLIK ZSUZSA, GÁLFFY ISTVÁNNÉ
GUDENUS LÁSZLÓ, GULYÁSNÉ LŐRINCZ ILONA
HORVÁTH ATTILA, HUBERT TIBOR
KECZER ZOLTÁN, LŐRENTEI TAMÁS
MARTON ANIKÓ, ORBÁN ANNA
SZABADHEGYI CSABA, SZLÁVI PÉTER
TÖRÖK TURUL, VARGA BALÁZS, ZSAKÓ LÁSZLÓ

ISBN 963 09 3977 0

TARTALOM

ELŐSZÓ

1987-ben nagy feladatra vállalkozott egy számítástechnika tanárokból álló, 25 fős társaság (volt közöttük általános iskolában, gimnáziumban, szakközépiskolában, főiskolán, illetve egyetemen tanító is). Összegyűjtötték mindazon programozási feladatokat, amelyek programnyelvtől függetlenül megfogalmazhatók, s a megoldásuk a programnyelvek széles körében elkészíthető.

Így készült el a Számítástechnikai feladatok 2000-ig című feladatgyűjtemény, 2000 feladattal, s ezek egy részének megoldási javaslatával.

E nagy munka elkészítői, a feladatok összegyűjtői: Achs Ágnes, Bányai Antal, Berceliné Sztyehlik Zsuzsa, Botlik Benedek, Gálffy Istvánné, Gudenus László, Gulyásné Lőrincz Ilona, Hanák D. Péter, Horváth Attila, Hubert Tibor, Keczer Zoltán, Koltai Márta, Kőhegyi János, Környei László, Lőrentei Tamás, Németh Piroska, Orbán Anna, Papné Horváth Katalin, Pethő József, Sárkány Ernő, Szabadhegyi Csaba, Szlávi Péter, Török Turul, Varga Balázs, Zsakó László.

Tíz év elteltével a korábbi gyűjtők egy része úgy gondolta, hogy az elmúlt évtized számítástechnikai tankönyvkiadása egyik legnagyobb volumenű termékét újra ki kellene adni. Ennek érdekében az arra vállalkozók átnézték a feladatokat, az elavultakat kihagyták, a módosítani valókat átfogalmazták, s a feladatgyűjteményt új feladatokkal bővítették.

Kikerült e feladatgyűjteményből a Versenyfeladatok fejezet, mivel ez megjelent önálló könyvként a Neumann János Számítógéptudományi Társaság gondozásában, Versenyfeladatok tára címmel könyvben, illetve CD-n.

A feladatgyűjteményt az eredetihez hasonlóan, mérete miatt most is két kötetben jelentetjük meg.

A könyv mindenkinek szól, aki számítástechnikát tanul a Nemzeti Alaptanterv szerint, de elsősorban azok forgathatják sikeresen, akik az átlagosnál többet foglalkoznak programozással. Így alapműve azoknak, akik részt vesznek számítástechnikai versenyeken (pl. a Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi Versenyen), vagy programozásból szeretnének felvételizni egyetemekre, illetve főiskolákra. Ugyanúgy jó feladatokat tartalmaz programozást tanuló egyetemi és főiskolai hallgatóknak.

A feladatgyűjtemény fejezetei olyan témakörökről szólnak, amelyek az általános és a középiskolai számítástechnika tananyaghoz kapcsolódnak, de akad közöttük több e szint feletti tudást igénylő probléma is. Minden fejezet néhány, a tartalomra utaló kulcsszóval, egy rövid bevezetővel és irodalomjegyzékkel kezdődik. A bevezető nem tartalmaz részletes magyarázatot, csupán felsorolja a fejezet fel- adatainak megoldásá-

hoz szükséges ismeretek körét, amely a felsorolt művekben részletesen is elolvasható. Ha a fejezet olyan témát dolgoz fel, amelyet tovább csoportosítottunk, akkor alfejezeteket talál az Olvasó. Ezek elején ugyancsak tömör bevezetővel próbáltuk segíteni a feladatmegoldást.

A feladatok sorszámozása fejezetenként 1-től kezdődik és folyamatos. Háromféle nehézségi szintet jelképező ábra található a sorszámok mellett:

 a legkönnyebb feladatok mellett egy, a dobogó harmadik fokán ülő emberke látható,

 a közepes nehézségűeknél az emberke a dobogó második fokán ül,

 a legnehezebbeknél már legfelül található az emberke, jelképezve ezzel azt, hogy aki ezeket megoldja, méltán tekinthető győztesnek.

Az azonos nehézségi szinttel jelölt problémák értelemszerűen lehetnek fejezetenként, sőt alfejezetenként is különböző súlyúak. Például az Elemi grafika rész legnehezebb feladata is könnyebb, mint a Rekurzív ábrák, vagy a Térbeli ábrázolás legegyszerűbbje. Emiatt is kérjük az Olvasót, hogy egy feladat megoldás, kiadása előtt olvassa el az ide tartozó bevezetést is, mert abból kiderülhet a fejezet átlagos nehézségi szintje.

A matematikával, fizikával kapcsolatos hasonló jellegű feladatgyűjteményekben általában a feladatok megoldását is megtalálhatjuk. Ebben a könyvben a feladatok többségéhez csak útmutatót nyújtunk, mert a nagybetűs MEGOLDÁST senki sem tudná leírni, hiszen mindenki mást tartana annak. (A dolog ahhoz hasonlít, mintha irodalom órán verset kellene írni valamilyen témából, s a következő órán a tanár elmondaná a helyes megoldást – verset.) Néhányszor a feladatok megfogalmazása nem teljesen részletezett, ezzel az volt a célunk, hogy itt nagyobb szerepet adjunk az Olvasó fantáziájának.

Ez egy általános célú programozási feladatgyűjtemény, ezért nem szerettünk volna kötődni egyetlen programozási nyelvhez sem. A feladataink között azonban elkerülhetetlenül vannak bizonyos programozási nyelvekkel kapcsolatosak is. Mivel ma a közoktatásban a Pascal a legelterjedtebb általános célú programozási nyelv, ezért nem meglepő, hogy a nyelvorientált feladatok nagyrészt ehhez a nyelvhez kapcsolódik. Megtalálható még ezen kívül néhány BASIC-hez és C-hez kapcsolódó feladat is.

E könyv előzményében még úgy gondoltuk, hogy a gépközeli feladatok megoldása alapvetően assembly programozást igényel, mára azonban már túllépett ezen a számítástechnika. Így ajánlhatjuk mindenkinek, hogy e feladatok megoldására is Pascal vagy C nyelvű programokat készítsen.

7. FEJEZET

SZAKTÁRGYI FELADATOK

MATEMATIKA
FIZIKA
BIOLÓGIA
KÉMIA
TECHNIKA
EGYÉB

7. SZAKTÁRGYI FELADATOK

Itt olyan feladatokkal foglalkozunk, amelyek megoldásában – a legtöbb esetben – fontosabb a szaktárgyi ismeret, mint a számítástechnikai. Ezeket a feladatokat természetesen besorolhattuk volna számítástechnikai szempontból más fejezetekbe, de így éppen az összetartozásukat kifejező jellemző, a szaktárgy szorult volna háttérbe.

Fontosnak tartjuk az erőteljesen szaktárgyakhoz kötődő feladatok egy helyen szerepeltetését amiatt is, mert a számítástechnika-oktatás területén az egyik továbblépési irány éppen a szakorientált számítástechnikai képzés.

A fejezetben a számítástechnika alkalmazásának szempontjából fontosabb tárgyaknak külön alfejezetet szenteltünk, a többieket pedig egy alfejezetbe foglaltuk. (A számítógépes méréssel, vezérléssel kapcsolatos feladatokkal külön fejezetben foglalkozunk.)

Így figyelembe vettük a következő tantárgyakat:

- matematika,
- fizika,
- biológia,
- kémia,
- technika
- egyéb tárgyak.

A feladatok kitűzése, megoldása általában kevésbé jelent problémát egy matematikai vagy fizikai jellegű példánál, viszont nehézségek merülhetnek fel pl. egy biológiához kapcsolódó feladatnál: a megoldás számítástechnikai oldalát ismerő tanár rendszerint nem rendelkezik a szükséges biológiai ismeretekkel, míg az ezek birtokában lévő biológiatanárnak nincs kellő gyakorlata a számítástechnika terén. Ilyen esetben feltétlenül kettejük együttműködésére van szükség.

Javasolt irodalom:

1. M. Eigen - R. Winkler: A játék.
   Gondolat, 1981
2. Marx Gy.: A természet játékai.
   Ifjúsági Lap- és Könyvkiadó, 1984

7.1 Matematika

Ez a rész zömmel olyan matematikai tárgyú számítástechnikai feladatokat tartalmaz, amely nem haladja meg a középiskolai matematikaanyagot. Ahol mégis, ott igyekeztünk a feladat szövegében, illetve a függelékben útmutatást vagy javasolt irodalmat adni.

7.1.1 Egész számok és polinomok aritmetikája

7.1 

Írjunk programot, amely tetszőleges természetes számról eldönti, hogy prím-e!

7.2 

Írjunk programot, amely 2-vel kezdődően kiírja a 100-ig található prímszámokat!

7.3 

Írjunk programot, amely 2-vel kezdődően kiírja az első 100 prímszámot!

7.4 

Határozzuk meg programmal azokat a prímeket, amelyek egy adott természetes számnál kisebbek!

7.5 

Gyorsítsuk az előző programot azáltal, hogy csak a már megtalált prímekkel való oszthatóságot vizsgáljuk!

7.6 

Határozzuk meg programmal egy természetes szám osztóinak számát!

7.7 

Határozzuk meg programmal egy természetes szám osztóit!

7.8 

Határozzuk meg programmal egy természetes szám páratlan osztóit!

7.9 

Határozzuk meg programmal egy természetes szám négyzetszám osztóit!

7.10 

Határozzuk meg programmal egy természetes szám prímosztóit!

7.11 

Állapítsuk meg egy természetes szám prímosztóinak multiplicitását! (A szám prímhatványok szorzataként történő előállításában a hatványkitevőket.)

7.12 

Határozzuk meg programmal egy természetes szám legnagyobb multiplicitású prímosztóit!

7.13 

Határozzuk meg programmal egy természetes szám egyszeres multiplicitású prímosztóit!

7.14 

Határozzuk meg programmal egy természetes szám osztóinak összegét!

7.15 

Döntsük el egy természetes számról, hogy tökéletes szám-e! (Tökéletes az a szám, amely egyenlő a nála kisebb osztóinak összegével.)

7.16 

Barátságosnak mondunk két természetes számot, ha az egyik (nála kisebb) osztóinak összege megegyezik a másik számmal és viszont. Döntsük el két természetes számról, hogy barátságosak-e!

7.17 

Keressünk barátságos számokat!

7.18 

Állapítsuk meg euklideszi algoritmussal két szám legnagyobb közös osztóját és legkisebb közös többszörösét!

7.19 

Írjuk meg az előző feladat megoldását az egészrész függvény és az osztás használata nélkül!

7.20 

Állapítsuk meg programmal, hogy N és M természetes számok relatív prímek-e!

7.21 

Adott az N természetes szám. Határozzuk meg programmal az N-nél kisebb, N-hez relatív prím számokat!

7.22 

Ikerprímeknek nevezzük az egymástól kettővel különböző prímeket. Keressünk programmal ikerprímeket!

7.23 

Írjunk programot, amelyik egy tetszőleges pénzösszeget a szokásos forint címletekre bont (címletező program)!

7.24 

Minden olyan esetben, amikor többeknek fizetünk ki különböző pénzösszeget, nem csupán annak fedezetéről, hanem a megfelelő címletekről is gondoskodnunk kell. Egészítsük ki a címletező programot azzal, hogy az egyes címletekből kifizetendő darabszámokat mindaddig összegezze, míg 0-t nem akarunk címletezni! Ilyenkor az eddigi tételek összesítését írja ki!

7.25 

Készítsünk programot véges tizedes tört alakú racionális számok p/q alakra hozásához (p és q relatív prímek)!

7.26 

Készítsünk programot, amelyik adott számig meghatározza a pitagoraszi számhármasokat!

7.27 

Írjunk programot egy polinom értékeinek kiszámítására a Horner-elrendezéssel!

7.28 

Írjunk programot együtthatóikkal adott polinomok szorzatának kiszámítására!

7.1.2 Algebra

7.29 

Írjunk programot, amely Gauss módszerével old meg lineáris egyenletrendszereket!

7.30 

Írjunk programot, amely az egyenletrendszer determinánsa segítségével old meg lineáris egyenletrendszereket!

7.31 

Egy függvényből N db rendezett párt ismerünk. Lagrange módszerével készítsük el a legfeljebb N-1-edfokú interpolációs polinomot, és ezzel számítsuk ki tetszőleges helyen a függvényértéket!

7.1.3 Kombinatorika

7.32 

Szemléltessük programmal az alapvető kombinatorikai fogalmakat! A program készítse el egy legfeljebb N elemű halmaz összes permutációját! Keressünk olyan algoritmust, amely nem követeli meg az összes eddig előállított sorozat tárolását!

7.33 

Az előző feladathoz hasonlóan készítsük el egy legfeljebb N elemű halmaz összes K elemű ismétléses kombinációját! Végezzük el ugyanezt az ismétlés nélküli kombinációkra is!

7.34 

Készítsük el egy legfeljebb N elemű halmaz összes K elemű ismétléses variációját! Végezzük el ugyanezt az ismétlés nélküli variációkra is!

7.35 

Készítsük el egy legfeljebb N elemű halmaz összes ciklikus permutációját

7.36 

Készítsünk programot faktoriálisok kiszámítására! Ha van rá mód, használjuk ki a rekurzió lehetőségét!

7.37 

Készítsünk programot a binomiális együtthatók kiszámítására faktoriálisok segítségével!

7.38 

Készítsünk programot a binomiális együtthatók kiszámítására a Pascal-háromszög alapján!

7.39 

Írjunk programot, amely előállítja a Pascal-háromszög tetszőleges sorát!

7.40 

Nyomtassuk ki a Pascal-háromszöget adott soráig!

7.1.4 Geometria

7.41 

Készítsünk programot a háromszög területének kiszámítására, ha adott

- a három oldal,
- két oldal és egy szög (vigyázat: melyik szög!),
- egy oldal és két szög!

7.42 

Közelítsük a π értékét úgy, hogy a kör kerülete helyett egyre növekvő oldalszámú szabályos érintősokszögek kerületével számolunk! Meddig van értelme az oldalszám növelésének?

7.43 

Közelítsük a π értékét úgy, hogy a kör kerülete helyett egyre növekvő oldalszámú szabályos húrsokszögek kerületével számolunk! Meddig van értelme az oldalszám növelésének?

7.44 

A síkban egy pontnak és két párhuzamos egyenesnek ötféle kölcsönös helyzete lehet: a pont a két párhuzamos egyenes fölött, alatt, között lehet, valamint illeszkedhet az egyikre vagy a másikra. Írjunk programot, amely tetszőleges síkbeli párhuzamos egyenespár és a P(x, y) pont esetén megadja ezek kölcsönös helyzetét! Hogy célszerű a bemenő adatokat megadni? Milyen elfajult esetek ellen kell védeni a programot?

7.45 

Egy négyszöget négy csúcsának koordinátáival adunk meg. Készítsünk programot, amely alkalmas egy így megadott négyszög osztályba sorolására (tehát megállapítja, hogy az adott négyszög húrnégyszög, érintőnégyszög, trapéz, deltoid, paralelogramma, téglalap, rombusz stb.-e)!

7.46 

Adott egy táblázatban csúcsai koordinátáival N négyszög. Készítsünk programot, amely bármely (az előző feladatban körvonalazott) négyszögosztályhoz megadja a kérdéses osztályba tartozó négyszög(ek) csúcsait!

7.47 

Írjunk programot, amely három egész koordinátájú pont koordinátáiból meghatároz egy negyedik pontot úgy, hogy a négy pont egy paralelogramma négy csúcsa legyen! Adjon értelmes hibaüzenetet abban az esetben, ha nincs ilyen pont! Vizsgálja meg a program a megoldások számát is!

7.48 

Írjunk programot csúcsaival adott sokszög felrajzolására!

7.49 

Adott a síkon N pont a koordinátáival. Adjuk meg csúcsaival azt a legszűkebb konvex sokszöget, amely belsejében vagy határán tartalmazza az adott pontokat (szokás ezt a pontok konvex burkának is nevezni)!

7.50 

Készítsünk programot egy koordináta-rendszerben lévő egyenes és kör kölcsönös helyzetének meghatározására!

7.51 

Készítsünk programot egyenletük együtthatóival adott egyenesek és körök metszéspontjainak meghatározására!

7.52 

Egy ellipszissereg minden tagja kanonikus helyzetű (azaz középpontja az origó, továbbá tengelyeik párhuzamosak a koordinátatengelyekkel), és mindnyájukra igaz, hogy a*b=100. Írjunk olyan programot, amelyik megrajzolja az ellipszissereg néhány elemét! (Megjegyezzük, hogy az ellipszisek mind azonos területűek.)

7.53 

Írjunk olyan programot, amely egy adott ponthoz mint fókuszponthoz, s egy adott egyenessel párhuzamos vezéregyenesekhez tartozó parabolasereg néhány elemét rajzolja ki!

7.54 

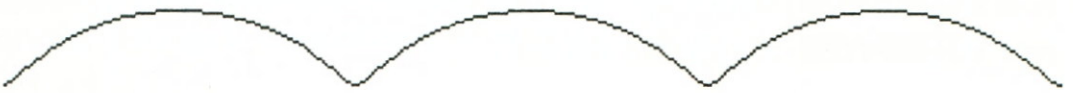
Írjunk programot amely kirajzolja a következőkben megadott görbéket:

1. Ciklois: Egy r sugarú kör egy egyenesen csúszás nélkül gördül. A körhöz rögzített, annak középpontjából kiinduló félegyenes egy pontjának mozgását figyeljük. Ha t-vel jelöljük a kör szögelfordulását, a-val pedig a kérdéses pontnak a kör középpontjától mért távolsága és a sugár arányát, a leírt görbe paraméteres egyenletrendszere

$$x = r*(t-a*\sin(t))$$
$$y = r*(1-a*\cos(t)).$$

Ebből a=1 esetén a körvonal egy pontjának pályáját kapjuk, ezt csak egyszerűen cikloisnak nevezzük,

0<a<1 esetén a kör egy belső pontjának pályáját, az ún. nyújtott cikloist kapjuk,



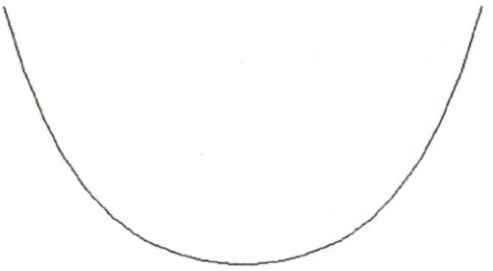
a>1 esetén pedig egy (meghosszabbított sugarán rögzített) külső pontjának pályáját kapjuk, ennek a neve: hurkolt ciklois.

2. Láncgörbe:

$y = \mathrm{ch}\frac{x}{a}$, ahol ch a koszinusz-hiperbolikusz függvény,

$\mathrm{ch}(t) = \frac{e^{t} + e^{-t}}{2}$.

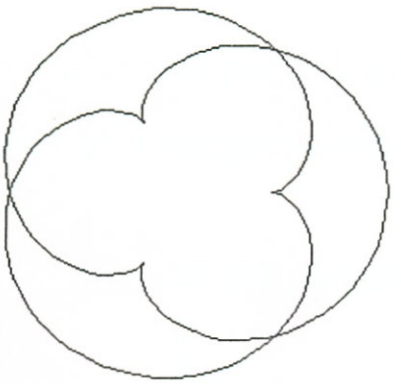
(A görbe nevét onnan kapta hogy bármely, két pontjában rögzített, tökéletesen hajlékony lánc saját súlya hatására ilyen alakot vesz fel.)



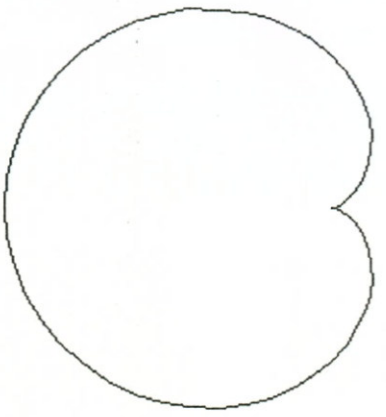
7.55

Írjunk programot, amely kirajzolja a következő görbéket:

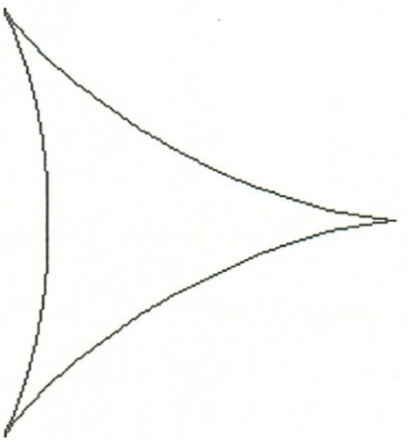
1. Epiciklois: Egy r sugarú kör egy R sugarú körön kívül csúszás nélkül gördül. A cikloishoz hasonlóan itt is három különböző esetet érdemes megnézni (a pont a körön, azon belül, kívül van, ld. az előző feladatot!). Az ábrán egy, a kör kerületén rögzített pont pályáját láthatjuk, R/r= 3/2 esetén.



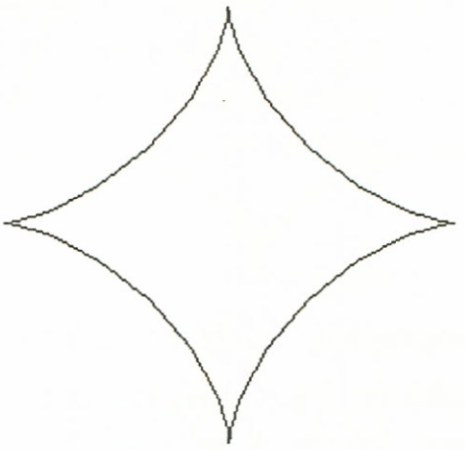
2. Kardioid: Olyan epiciklois, ahol R=r.



3. Hipociklois: Egy r sugarú kör egy R sugarún belül csúszás nélkül gördül. Az ábrán egy, a kör kerületén rögzített pont pályáját láthatjuk, R/r=3/2 esetén.



4. Asztroid: A hipociklois speciális esete, amikor a gördülő kör sugara r = R/4.



7.56

Határozzuk meg azt a hipocikloist, ahol R=4*r (ld. az előző feladatot)!

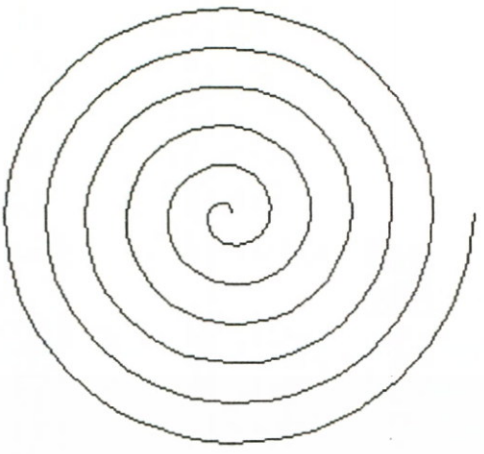
7.57

Határozzuk meg azt a hipocikloist, ahol R=2*r!

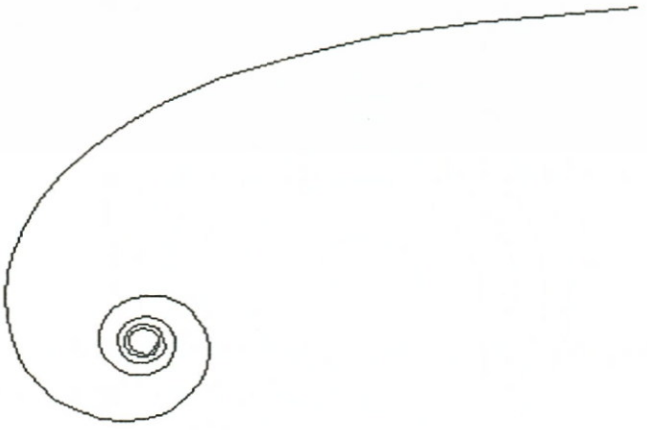
7.58

Írjunk programot, amely kirajzolja a következő polárkoordinátás egyenlettel megadott görbéket (polárkoordináták: r, φ – az origóból az adott pontba húzott szakasz hossza, illetve az x-tengellyel bezárt szöge).

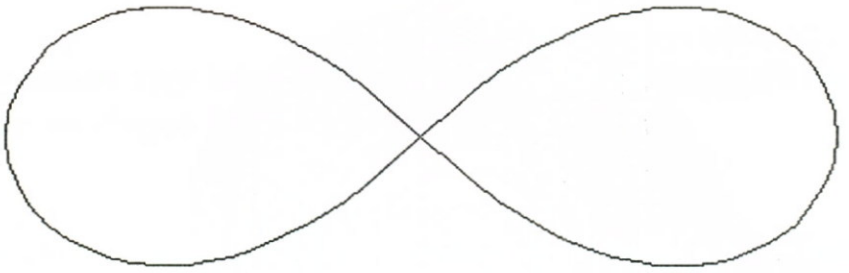
1. Archimedesi spirál: egy síkbeli pont egy rögzített O ponttól való távolsága egyenesen arányos a pont körüli elfordulásának egy kezdeti félegyenestől számított szögével: $r = a * \varphi$ csigavonalak távolsága állandó.



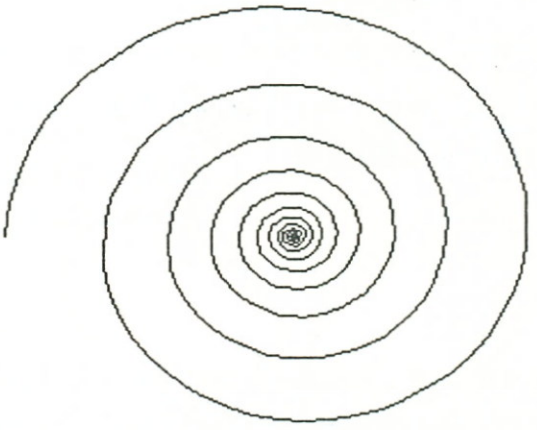
2. Hiperbolikus spirál: tetszőleges pontjának a rögzített O ponttól mért távolsága fordítottan arányos a pont körül végzett forgás szögével. $r = \frac{a}{\varphi}$ (a>0).



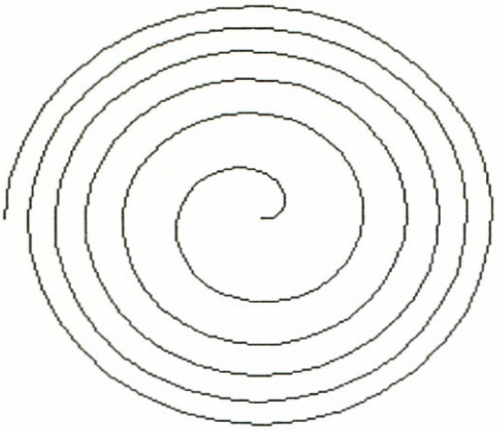
3. Lemniszkáta: olyan pontok halmaza a síkban, amelyeknek két ponttól (a két fókusztól) vett távolságszorzatuk állandó.



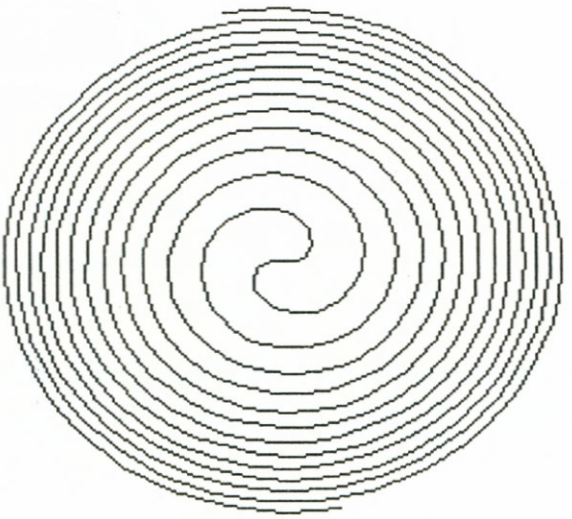
4. Logaritmikus spirál: a rögzített O pontból kiinduló félegyeneseket állandó ψ szög alatt metszi. A görbe egyenlete $r = a * e^{k*\varphi}$ alakú, ahol $k = \mathrm{ctg}(\psi)$



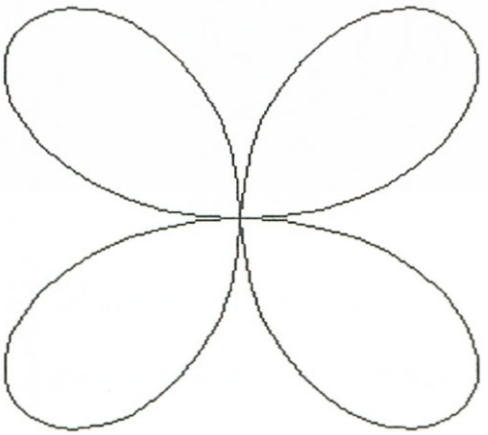
5. Parabolikus spirál: $r = a * \sqrt{\varphi} + 1$



6. Fermat-féle spirál: $r^2 = a^2 * \varphi$



7. Rozetta: $r = a * \sin(k * \varphi)$, illetve $r = a * \cos(k * \varphi)$, ahol a>0 és k>0. Próbáljunk 3, 4, 5 levelű rozettát előállítani!



7.1.5 Sorozatok, függvények

7.59

Közelítsük egy N pozitív szám négyzetgyökének értékét a Newton-féle gyökvonó algoritmus segítségével!

7.60

Készítsünk programot a kétoldali közelítés módszerével pozitív számok négyzetgyökének adott pontosságú kiszámítására!

7.61

Készítsünk iterációs programot az N szám köbgyökének adott pontosságú kiszámítására!

7.62

Készítsünk programot egy képlettel megadott sorozat tagjainak kiszámítására, ábrázolására és az esetleges határérték megsejtetésére!

7.63

Készítsünk programot az f(x)=0 alakú egyenletek [a, b] intervallumba eső gyökének meghatározására! (Feltesszük, hogy az intervallumba csak egy gyök esik, és f(a)*f(b)<0.)

1. Válasszunk az intervallum belsejéből véletlenszerűen egy pontot, és folytassuk a módszert azzal az intervallummal, amelyiknek végpontjaiban különbözik az előjel!
2. Az előző módszerhez hasonlóan felezzük az intervallumot, de most mindig középen!
3. Az intervallumot most az (a, f(a)) és (b, f(b)) végpontú szakasz és az x tengely metszéspontjával felezzük!
4. Használjuk Newton módszerét!

$$x_{i+1} = x_i - \frac{f(x_i)}{f'(x_i)}$$

Mindegyik esetben a közelítést akkor fejezzük be, ha elég jó a közelítés, vagy előre meghatározott lépésszámot elért! Hasonlítsuk össze a módszereket annak alapján, hogy különböző függvények esetén hányszor kellett f(x) értékét kiszámítani!

7.64 

Készítsünk programot az f(x)*g(x)>0 típusú egyenlőtlenségek megoldására!

7.65 

Szemléltessük az integrálszámítás alapjait! Írjunk programot, amely az alsó és felső közelítés, illetve a trapézmódszer alapján tetszőleges integrálható függvény határozott integrálját közelíteni tudja az adott felosztásnak vagy hibakorlátnak megfelelően! Ábrázoljuk a valódi érték és a közelítés közötti eltérést! Hasonlítsuk össze adott felosztás esetén az egyes módszereket!

7.1.6 Vegyes feladatok

7.66 

Írjunk programot, amelyik négyműveletes számológéppé butítja a számítógépet! (Például a 4+3= jelsorozat begépelésére írjon ki 7-et!)

7.67 

Szerkesszünk olyan programot, amelyik a beírt egész számot, mint a számjegyekből álló szöveget kezeli, és akár százjegyű számok között is úgy tudja elvégezni az összeadást, mint mi papíron ceruzával!

7.68 

Módosítsuk az előző programot az alábbi lehetőségek szerint:

1. terjesszük ki a másik három alapműveletre,
2. terjesszük ki más számrendszerekre,
3. terjesszük ki tizedes törtekre!

7.69 

Készítsünk nagypontosságú racionális aritmetikai modult az alábbi műveletekkel: összeadás, kivonás, szorzás, osztás, két egészből racionális készítése, racionális kerekítése egészre!

7.70 

Készítsünk nagypontosságú komplex egész aritmetikai modult az alábbi műveletekkel: összeadás, kivonás, szorzás!

7.71 

Készítsünk nagypontosságú fixpontos valós aritmetikai modult az alábbi műveletekkel: összeadás, kivonás, szorzás, osztás, racionálisról fixpontos valósra konvertálás, illetve fordítva!

7.72 

Készítsünk nagypontosságú lebegőpontos valós aritmetikai modult az alábbi műveletekkel: összeadás, szorzás, kivonás, osztás, fixpontosról lebegőpontosra, illetve vissza konvertálás, egészről lebegőpontosra, illetve vissza konvertálás!

7.73 

Készítsünk nagypontosságú síkbeli vektoraritmetikai modult az alábbi műveletekkel: összeadás, skaláris és vektoriális szorzás!

7.74 

Készítsünk nagypontosságú térbeli vektoraritmetikai modult az alábbi műveletekkel: összeadás, skaláris és vektoriális szorzás!

7.75 

Készítsünk tetszőleges számrendszerből tetszőleges számrendszerbe átalakító programot egész, racionális, fixpontos és lebegőpontos valós számokra!

7.76 

Készítsünk polinomaritmetikai modult: összeadás, kivonás, szorzás, osztás!

7.77 

Készítsünk nagypontosságú racionális aritmetikai modult az alábbi műveletekkel: a sin, a cos, a tan és a ctg, az exp és az ln függvények közelítése hatványsorukkal!

7.78 

Közelítsük a π értékét a geometriai valószínűség alapján! Egy négyzetbe írjunk kört, és szórjuk tele az egész alakzatot véletlenszerűen választott pontokkal! Számláljuk le a négyzeten és a körön belül levő pontokat! Arányuk a területek arányát közelíti.

7.2 Fizika

Ebben az alfejezetben háromféle feladatot találunk. Az első csoportba azok tartoznak, amelyek megoldása egy-egy mennyiség kiszámítását, táblázat vagy grafikon készítését kívánja meg. Ezekről a feladatokról nincs mit előrebocsátani.

A második csoportba tartozó feladatok megoldásakor olyan egyenletekkel találjuk magunkat szemben, amelyekben egy-egy mennyiség függ a saját változási gyorsaságától (esetleg még annak a változási gyorsaságától is). Az ilyen egyenleteket differenciálegyenleteknek nevezzük, megoldásuk egy függvény (pl. a sebesség függése az időtől). Ezeket a középiskolai tanulmányok alapján nem tudjuk pontosan megoldani. Olyan feladat is lesz, amelyiknek nem ismert a pontos megoldása.

Közelítő megoldást úgy kaphatunk, ha a keresett függvény értelmezési tartományát apró intervallumokra osztjuk, és mindig az előző eredmények felhasználásával intervallumról intervallumra kiszámítjuk a keresett mennyiségeket. A felosztás ésszerű finomításával az eredmény pontosabbá tehető.

Lássunk egy példát!

Számítsuk ki, mennyi idő alatt ér földet egy ejtőernyős 1000 m magasból! Tegyük fel, hogy a vízszintes sebessége elhanyagolható, és az ernyő rögtön kinyílik!

Megoldás:

Az ejtőernyős mozgásegyenlete: $a = g - \frac{c * v * |v|}{m}$, ahol g a nehézségi gyorsulás, c a közegellenállási együttható, v az esés sebessége, m az ejtőernyős tömege, az a pedig a gyorsulása. Válasszunk egyelőre önkényesen egy dt időtartamot, h pedig jelölje az eddig megtett utat! Nézzük a megoldás algoritmusát!

```
Program:
   H:=0: V:=0: G:=10: M:=80: C=22: DT=1: T:=0
   Ciklus amíg H<1000
      A:=G-C*V*abs(V)/M
      V:=V+A*DT
      H:=H+V*DT
      T:=T+DT
   Ciklus vége
   Ki: T
Program vége.
```

A dt időtartam akkor jó, ha megfelezésével már csak a pontossági igényünkön belül változik az eredmény. Kezdetben olyan rossz is lehet a dt értéke, hogy a valóságos mozgástól merőben különbözik a modell. (A példa is ilyen, felfelé esik az ejtőernyős.) Ha a feladat grafikon vagy pálya rajzolását követeli meg, először mindig ábrázolás nélkül futtassuk a programot, és tájékozódjunk az ábrázolandó értékek minimumáról és maximumáról! Ezután tudjuk csak úgy nagyítani vagy kicsinyíteni az ábrát, hogy kiférjen a képernyőre. Soha ne változtassuk az eredeti adatokat az ábrázolhatóság kedvéért!

A harmadik csoportba a szimulációs feladatok tartoznak. Ezeknél pontosan fordítva gondolkodunk, mint ahogy azt a kísérleti fizikában megszoktuk. Ott a sok szabadsági fokú anyaghalmazban megfigyelt jelenség magyarázatát keresve a részecskék mozgását kell kitalálnunk. Itt a részecskék egy bizonyos fajta mozgását feltételezve a modellbeli és a valóságos anyaghalmaz jelenségeit hasonlítjuk össze, és ennek alapján elfogadjuk vagy elvetjük a feltevésünket. Lássunk egy példát!

Vizsgáljuk meg egy gáz térkitöltését! A teret jelentse egy játéktábla, ennek N oszlopában és M sorában legyen P db „o”, ezek jelentsenek egy-egy molekulát! A

molekulák véletlenszerűen elmozdulnak valamelyik szomszéd helyre. Ha ott már volt molekula, helyet cserélnek. A szimuláció algoritmusa a következő:

```
Program:
   Inicializálás
   Ciklus amíg szükséges
      Szimulációs lépés
      Kijelzés
   Ciklus vége
Program vége.
Inicializálás:
   N, M, P beállítása
   TS(N, M) feltöltése a kezdeti eloszlással
   Kijelzés
Eljárás vége.
Szimulációs lépés:
   X:=RND(N): Y:=RND(M)
   Ciklus
      Ciklus
         V:=RND(3)-2: W:=RND(3)-2
      amíg V=0 és W=0
      Ciklus vége
      X1:=X+V: Y1:=Y+W
   amíg X1<1 vagy Y1<1 vagy X1>N vagy Y1>M
   Ciklus vége
   Csere (TS(X, Y), TS(X1, Y1))
Eljárás vége.
Kijelzés:
   Ciklus X=1-től N-ig
      Ciklus Y=1-től M-ig
         Ki: TS(X, Y)
      Ciklus vége
   Ciklus vége
Eljárás vége.
```

A programot felgyorsíthatjuk, ha nem külön eljárásként írjuk meg a kijelzést, hanem minden szimulációs lépésben csak a megváltozott mezőket módosítjuk.

A TS(N, M) kezdeti feltöltését megoldhatjuk a billentyűzetről is, de a véletlen számokkal is elvégezhetjük.

7.79

Készítsünk programot az egyenletesen gyorsuló mozgás jellemzőinek kiszámítására! (pl. idő és út ismeretében adjuk meg a gyorsulást stb.)

7.80

Készítsünk programot a rugalmas és rugalmatlan ütközések szemléltetésére! A program kérdezze meg az ütközés minőségét, a két test tömegét és sebességét, az ütközés bemutatása után írja ki az új sebességeket!

7.81

Rajzoltassuk a képernyőre egy rugóra akasztott test hely-idő grafikonját! A gyorsulás a rugó erőtörvényéből: $a = -\frac{D}{m} * y$. Használjuk az időtartamonkénti állandó sebesség és gyorsulás adta közelítés lehetőségét!

7.82

Az előző feladatot egészítsük ki a közegellenállással! Ekkor a gyorsulás értéke:

$$a = -\frac{D}{m} * y - C * v * |v|.$$

7.83

Modellezzük a rezgő testet akkor is, ha a közeg helyett súrlódás csillapítja! A súrlódási erő nagysága legyen állandó, és a sebességgel ellentétes irányú!

7.84

Az előző feladatban vegyük figyelembe azt is, hogy a test a szélső helyzetekben megáll, és újabb elindulásának az a feltétele, hogy a rugóerő nagyobb legyen a tapadási erőnél!

7.85

Írjunk programot párhuzamos rezgések összetételének szemléltetésére!

7.86

Írjunk programot merőleges rezgések összetételének szemléltetésére (Lissajous-görbék)!

7.87

Rajzoljuk fel két adott helyzetű hullámforrás interferenciaképét!

7.88

Két végén azonos magasságban felfüggesztett lánc lazán belóg. Az egyes láncszemekre ható erők figyelembevételével rajzoltassuk ki a lánc alakját!

7.89

Egy lánc félig lelóg az asztalról. Az asztalon lévő rész egyenes és az asztal szélére merőleges. Az asztal és a lánc között a súrlódás elhanyagolható. Ebből a

helyzetből a láncot elengedjük. Tetszőlegesen választott időközönként rajzoltassuk ki a lánc alakját!

7.90

Az egylépcsős rakéta akkor érhetné el a maximális sebességet, ha az üzemanyagát egy lépésben egyszerre égetné el. Írjunk programot, amely bármely beadott N természetes számhoz megadja, mekkora lenne a rakéta sebessége, ha az üzemanyagot N egyenlő részre osztva lövellné ki hátrafelé!

7.91

Terjesszük ki az előző program lehetőségeit kétlépcsős rakétára!

7.92

Newton törvényeiből és a gravitációs törvényből kiindulva rajzoltassuk a képernyőre a Föld hét naponkénti helyzetét! A koordináta-rendszer origójában legyen a Nap, a szükséges adatokat keressük táblázatból!

7.93

A Föld pályájával együtt rajzoltassuk fel a Vénuszét is (ld. az előző feladatot)! A pontok számából Kepler III. törvényére következtethetünk.

7.94

Képzeljünk el egy a Földhöz hasonló bolygót, amelyik fele akkora sebességgel halad! Rajzoltassuk fel a pályáját! Kepler I. és II. törvényére következtethetünk.

7.95

Készítsünk űrhajós programot! Számítsuk ki egy űrállomás körpályájának adatait, és helyét negyedóránként rajzoltassuk a képernyőre! Az előző pontot mindig töröljük! Mozogjon a képernyőn egy olyan űrhajó is, amelyiknek sebességét a billentyűzetről növelni, illetve csökkenteni tudjuk, egyébként pedig a gravitációs törvénynek megfelelően mozog! A cél az űrállomás elérése az űrhajóval.

7.96

Egy függőleges tengelyű hengeres edényből a víz az alján levő pici lyukon kifolyik. A kifolyási sebesség a Bernoulli-törvényből minden vízszinthez kiszámítható. Rajzoltassuk fel a vízszint alakulását az idő függvényében, ha az edény és a lyuk átmérőjét ismertnek tételezzük fel!

7.97

Készítsünk programot különböző sűrűségű hasábok különböző sűrűségű folyadékokban való elhelyezkedésének szemléltetésére!

7.98

A víz alatt levő buborék térfogata a mélység lineáris függvénye. Ebből meghatározhatjuk az átmérőjét, a felhajtóerőt és a közegellenállást. Rajzoltassuk fel a buborék sebesség-idő, illetve mélység-idő grafikonját!

7.99

Egy légszivattyú pumpájának térfogata a ritkított tér térfogatának huszadrésze. Ábrázoljuk a nyomást a pumpálások számának függvényében!

7.100

Az előző feladatban a külső és a belső nyomás különbségével arányos sebességgel áramlik be a levegő. Hogyan függ az elérhető légritkítás a pumpálás szaporaságától?

7.101

Szemléltessük gázok állapotváltozásait p-V diagramon grafikusan!

7.102

Adott 1 kg 0 fokos és 1 kg 100 fokos víz. Ha duplafalú edényben lehetővé tesszük a hőátadást, a veszteségektől eltekintve egy-egy kg 50 fokos vizet kapunk. Ha azonban először csak a forró víz egy részével melegítjük a hideget, majd a következő résszel és így tovább, milyen magas hőmérsékletet érhetünk el? Mi lesz az eredmény ellenáramú hőcserélővel (amelyben folyamatosan, „végtelen kicsiny" tömegű részek hőcseréje valósul meg)?

7.103

Írjunk programot, amely egy plánparalel üveglemezen áthaladó fénysugár menetét rajzolja ki! A fénysugár beesési szögének változtatásával szemléltessük, hogyan változik az eltolódás szöge!

7.104

Készítsünk programot, amely két ismert törésmutatójú, síklappal határolt közeg határán szemlélteti a fénytörést, különös tekintettel a teljes visszaverődés jelenségének bemutatására!

7.105

Írjunk programot, amely egy prizmán áthaladó fénysugár menetét rajzolja ki! A fénysugár beesési szögének változtatásával határozzuk meg a minimális eltérítés szögét!

7.106 

Készítsünk programot, amely egy lencse (előjeles) fókusztávolsága és a tárgytávolság ismeretében meghatározza a dioptriát, a képtávolságot és a keletkezett kép minőségét!

7.107 

Írjunk programot, amely egy gyűjtőlencse képalkotását szemlélteti a nevezetes sugármenetek megrajzolásával! Ügyeljünk arra, hogy tetszőleges bemenő adatok esetén is a képernyőre kerüljön a kép és a tárgy!

7.108 

Készítsünk programot, amely két (vagy több) pontszerű töltés elektromos mezejének energiáját közelíti! Bontsuk a teret olyan kis térfogatokra, amelyekben a mező homogénnek tekinthető!

7.109 

Készítsünk programot, amely két (vagy több) pontszerű töltés elektromos mezejébe ekvipotenciális görbéket rajzol!

7.110 

Modellezzük az elektron mozgását a síkkondenzátor belsejében, ha kezdősebessége a lemezekkel párhuzamos!

7.111 

Modellezzük egy elektromosan töltött test mozgását, miután vízszintes indukciójú mágneses mezőbe ejtettük!

7.112 

Számítsuk ki és rajzoljuk fel egy állandó feszültségről ellenálláson keresztül töltött kondenzátor feszültség–idő és áram–idő függvényét! A pillanatnyi áram az ellenálláson eső feszültségből Ohm törvényével számítható ki.

7.113 

Készítsük el a kondenzátor kisütésének grafikonjait is!

7.114 

Rajzoltassuk fel egy állandó feszültségű áramforrásra egy ellenállással sorosan kapcsolt tekercs feszültség–idő és áram–idő grafikonját!

7.115 

Egy tekercset tolóellenálláson keresztül egy akkumulátorra kötünk. A tolóellenállás csúszkáját egyenletesen húzzuk a növekvő ellenállás irányába. Határozzuk meg a tekercs feszültség–idő grafikonját!

7.116 

Határozzuk meg a rendszám, a tömegszám és az atommag tömege ismeretében a tömeghiányból az atommag kötési energiáját!

7.117 

Írjunk programot, amely az izotópok kötési energiáját tárolja is és a cseppmodell alapján számítja is! A két adatot a rendszám és a tömegszám alapján lehessen megkérdezni!

7.118 

Vegyük fel az előző program adatai közé a felezési időt és a bomlás módját! A kijelzés után egy billentyű lenyomásával lehessen kérni a bomlástermékek adatait is!

7.119 

Az atommag kötési energiájának mért és számított értéke közötti különbséget ábrázoljuk a rendszám függvényében! Válasszuk minden rendszámhoz a legstabilabb izotópot vagy a viszonylag stabilak energia-eltérésének átlagát!

7.120 

Egy Geiger-Müller számlálóval egyenlő időközönként több mérést végeztünk. Határozzuk meg a vizsgált rádioaktív anyag felezési idejét!

7.121 

Írjunk programot Rutherford szóráskísérletének modellezésére!

7.122 

Tegyünk kísérletet az atomerőmű működésének szimulálására! (Az ehhez szükséges alapvető ismeretek pl. a gimnáziumi 4. osztályos fizika tankönyvben megtalálhatók.) Minél több hatást vegyünk figyelembe!

7.123 

Készítsünk szimulációs programot a gázok nyomásának szemléltetésére! Előre megválasztott időtartamok alatt számláljuk a golyók falnak ütközését! Lehessen látni a képernyőn:

- a molekulákat a vizsgált térrészben,
- a molekulák számát,
- a falnak ütközések számát,
- grafikont a falnak ütközések számáról az utolsó néhány időegységben,
- az ütközések számának addigi átlagát.

A golyók egyenesen és egyenletesen haladjanak két falnak ütközés között!

7.124 

Írjunk programot a gázok egyenletes térkitöltésének szemléltetésére! A molekulák egy 2 részre osztott edény egyik feléből indulva véletlenszerűen mozogjanak, miközben a kiszemelt térrészben lévő részecskék számát hisztogram szemléltesse!

7.125 

Készítsünk kétdimenziós gázmodellt! Figyeljük meg a gáz eloszlását a játéktáblán!

7.126 

Az előző modellt módosítsuk úgy, hogy egy mezőre több molekula is kerülhessen, és ha a kiválasztott mező nem üres, a véletlen szomszédos mezőre tegyünk át egy molekulát!

7.127 

Vezessük be a gázmodellbe a gravitációt is! A véletlen szomszédok közül az alsókat nagyobb valószínűséggel válasszuk!

7.128 

Tegyünk az előző gázba egy falat, az egyik oldalára tegyük a gázt, és vizsgáljuk meg, hogyan függ a falon való áthaladáshoz szükséges idő a fal magasságától!

7.129 

A gáztartály egyik falát melegítsük! Ez annyit jelent, hogy itt gyorsabb molekulák vannak, tehát gyakrabban kell itteni mezőt választani. A sűrűség kisebb, tehát itt nagyobb valószínűséggel mennek felfelé a molekulák.

7.130 

Megvizsgálhatjuk a modellben a szél hatását is. Ha egy molekula balra eléri a falat, az hagyja el a játékteret, de helyette egy másikat hozzunk be ugyanolyan magasan jobbról! Milyen áramlás alakul ki ebben a térben egy fal mögött?

7.131 

Szemléltessük a kristályosodást is! Most gázmolekula helyett egy oldatban az oldott anyag molekuláira gondoljunk! A program csak akkor hajtsa végre a szomszédos mezők cseréjét, ha az elmozdított molekula szomszédainak száma nem csökken ettől!

7.132 

A gőz lecsapódásához juthatunk, ha a fallal való szomszédságot is figyelembe vesszük. Próbáljuk ki azt is, hogy a fal molekulái dupla erősséggel vonzzák a gőz molekuláit!

7.133

Egy kristályban N atom van. Az atomok rezgéseikkel energiát tárolnak, s energiájukat véletlenszerűen, adagokban cserélgetik egymás közt. Minden atom energiaadagjainak számával arányos valószínűséggel ad le energiát, s azt tetszőleges másik kaphatja. Készítsünk szimulációs programot, amely megjeleníti atomonként az energiaadagok számát, és az energiaadagok számának függvényében az éppen annyi energiával rendelkező atomok számát!

7.134

Készítsünk programot az egyik végén melegített rúdban a hő terjedésének szemléltetésére! (A melegítés annyit jelent, hogy a rúd végén újabb energiaadagok lépnek be.) Figyelembe vehetjük azt is, hogy a rúd egyes részei hőmérsékletüktől függő mértékben hőt adnak a környezetnek.

7.135

Egy szimulációs teret kettéosztunk egy féligáteresztő fallal. A szimulációs térben A és B gázmolekulák mozognak. Az A molekulák számára a fal átjárható, a B molekulák azonban csak balról jobbra képesek átlépni rajta. Készítsük el a szimulációs lépés algoritmusát!

7.136

Egy szimulációs teret kettéosztunk egy féligáteresztő fallal. A szimulációs térben A és B gázmolekulák mozognak. Az A molekulák számára a fal átjárhatatlan, a B molekulák pedig csak jobbról balra képesek átlépni rajta. Készítsük el a szimulációs lépés algoritmusát!

7.137

Egy szimulációs térben folyadékmolekulák mozognak. A szimulációs tér falához érő molekulák „megfagynak", azaz a helyükről nem képesek elmozdulni. Ha ezután egy folyadékmolekula „fagyott" molekulához ér, akkor szintén megfagy. Készítsük el a modell szimulációs lépését, valamint add meg a modell eredményét!

7.3 Biológia

Az ebbe a fejezetbe tartozó feladatok nagy része ún. szimulációs program. A szimuláció a kísérleteket helyettesítheti (esetleg kiegészítheti) olyan esetekben, ha a valóságos folyamat túl lassú, gyors, veszélyes, nagy, kicsi, a kísérlet túl drága, etikai akadályokba ütközik stb. Arra nincs mód, hogy e könyv keretein belül minden szimulációs modelltípus algoritmusára kitérjünk (diffúziós, sejtautomata stb.).

A szimulációs program általános (javasolt) szerkezete:

```
Program:
   Tájékoztató
   Kezdőértékadások
   Képernyők kezdeti állapotainak beállítása
      (életterek, grafikonok)
   Ciklus amíg szükséges
      Szimulációs lépés
      Eredményírás
   Ciklus vége
   Összesítés az eredményekről
Program vége.
```

A szimulációs lépés szerkezete pl.:

```
Folytonos modell:
   (I, J) hely választás
   Történés (I, J)-vel
Eljárás vége.
Diszkrét modell:
   Ciklus I=1-től N-ig
      Ciklus J=1-től M-ig
         UJ(I, J) előállítás
      Ciklus vége
   Ciklus vége
   Eredmény írás
Eljárás vége.
```

(A fejezet egyes feladataihoz ajánljuk „E. O. Wilson - W. H. Bossert: Bevezetés a populációbiológiába. Gondolat, 1981" könyvet.)

7.138

Adott két szülő AB0 vércsoport-rendszert meghatározó genotípusa! Határozzuk meg utódaik lehetséges vércsoportjait!

7.139

Írjunk programot, amely egy ember megfelelő genotípusának ismeretében megadja, milyen vércsoporthoz tartozik!

7.140

Adott egy diploid genotípus. Egy lokuszban lévő alternatív allélokat (s, G) vizsgálunk (szögletes, illetve gömbölyű borsó). Programunk döntse el egy egyedről, hogy heterozigóta-e, ha ismerjük a genotípusát, valamint adja meg a fenotípusát!

7.141

Adott egy diploid genotípus. Egy lokuszban levő alternatív allélokat (A, B) vizsgálunk. Tegyük fel, hogy az A allél P, B allél Q valószínűséggel található meg a populációban! Programunk készítsen táblázatot az AA, BB homozigóták, illetve az AB heterozigóták arányáról a populációban az egyensúly beállta után, ha P értékét 0 és 1 között változtatjuk 0.1-esével!

7.142

Adott egy cönológiai táblázat. A táblázat (i, j) eleme az i. hely j. faj általi borítottsága (a j. faj egyedeinek az i. helyen számlált darabszáma). Programunk határozza meg

1. a legfajgazdagabb helyet,
2. a leghomogénabb helyet (ahol a legkevesebbféle faj található),
3. a legelterjedtebb fajt (amely a legtöbb helyen megtalálható),
4. a legnagyobb egyedszámú fajt!

7.143

Egy Nx2-es táblázatban genotípusok leírását tároljuk (S és F allél). Töltsük fel a táblázatot allélgyakoriságok alapján! (Legyen az S allél gyakorisága P!)

7.144

Egy Nx2-es táblázatban genotípusok leírását tároljuk (S és F allél). Töltsük fel a táblázatot genotípus-gyakoriságok alapján! (Legyen az SS genotípus gyakorisága P, az FF-é Q.)

7.145

Egy Nx2-es táblázatban genotípusok leírását tároljuk (S és F allél). Számoljuk meg, hogy melyik genotípusú egyedből mennyi van!

7.146

Készítsünk növényhatározó adatbázist! A növény néhány tulajdonságának beírása után írja ki a program az összes megfelelő növényt minden tárolt tulajdonságukkal együtt! Kezdetben állapodjunk meg a tulajdonságok számában, az egyes tulajdonságok nevében és tárolásuk formájában! Az adatbázist tároljuk lemezen!

7.147

Készítsünk programot az RNS lánc véletlenszerű előállítására!

7.148

Írjunk programot, amely a DNS-molekula megkettőződésekor végbemenő néhányfajta mutációt szimulál:

- egy nukleotid kihagyása,
- egy nukleotid duplázódása,
- egy nukleotid megváltozása,
- két szomszédos nukleotid felcserélése!

7.149 

Készítsünk szimulációs demográfiai modellt! Kezdetben legyen N db nyúl! Élettartamuk maximálisan E év. Adjuk meg életkor szerint a halálozási H() és a születési valószínűséget S()! Tároljuk a nyulak korát a K() tömbben, de gondoljunk arra is, hogy az egyedszám a kezdetit meghaladhatja, illetve az elpusztultak helye felszabadul! (A fenti feltételek alapján belátható, hogy H(E)=1.) Vegyük sorra a nyulakat, és határozzuk meg ez évi sorsukat! Mire a sor végére érünk, már minden nyúl egy évvel öregebb lesz. Készítsünk grafikont az egyedszám változásáról, illetve a kor szerinti eloszlásról!

7.150 

Az előző modellben S() helyett vezessük be az utódok számának várható értékét V()!

7.151 

Vezessük be a demográfiai modellünkbe a járvány lehetőségét! Minden évben adott valószínűséggel következik be járvány, és ha járvány van, minden olyan egyed, amelyik az öregedés miatt nem pusztult el, még adott valószínűséggel elpusztulhat a betegségben. Ez utóbbi valószínűség függhet az életkortól.

7.152 

Az előbbi feladatban szerepelt demográfiai modellt egészítsük ki a kivándorlás jelenségével!

7.153 

Az előbbi feladatban szerepelt demográfiai modellt egészítsük ki a bevándorlás jelenségével!

7.154 

Írjunk programot ökológiai modellre! Egy négyzethálóval felosztott területen nyulak és rókák élnek. Válasszunk ki véletlenszerűen egy mezőt, és annak egy szomszédját! Cseréljük meg a két mező tartalmát! Ha a kiválasztott mezőn nyúl van, akkor az HN valószínűséggel meghal. Ha életben maradt, és van szomszédos üres mező, oda SN valószínűséggel utódot szül. Ha a kiválasztott mezőn róka van, HR valószínűséggel elpusztul. Ha életben maradt, és a szomszédjában van nyúl, egyet megeszik, különben meghal. Ha ennek ellenére életben maradt, és van szomszédos üres mező, SR valószínűséggel utódot szül. Ábrázoljuk a nyulak és rókák számának alakulását grafikusan!

7.155 

Egészítsük ki az előbbi feladatban szerepelt elemi zsákmányszerzési modellt azzal, hogy a zsákmányállatok és a ragadozók mozgását befolyásolja az, hogy mi van a környezetükben!

7.156 

Egészítsük ki az előbbi feladatban szerepelt elemi zsákmányszerzési modellt azzal, hogy a zsákmányállatok és a ragadozók mozgását befolyásolja az, hogy a saját fajtájú állatoktól a lehető legnagyobb távolságot akarják tartani.!

7.157 

Az előző feladat alapján készítsünk modellprogramot növényevésre! A nyulak eszik a füvet, akárcsak az előbb a rókák a nyulakat, de a fű nem mozog. A fű csak kiszárad, üres mezőn kinő, illetve a nyúl lelegeli.

7.158 

Írjunk programot versengési modellre! Két állatfaj él azonos területen, mozognak, születnek és meghalnak, mint a nyulak az előző feladatban. Ugyancsak az előző feladat szerint nő a táplálóknövényük. Egymás elől eszik a táplálékot, az A fajnak a túléléshez KA darab szomszédos mezőről kell legelnie, B-nek KB-ről. Ábrázoljuk a két faj egyedszámának alakulását!

7.159 

Készítsünk genetikai modellt a vércsoportok öröklődésére! Egy N elemű táblázatban adjuk meg a szülők vércsoportjának genotípusát! A szülők közül véletlenszerűen válasszunk kettőt, és ugyancsak véletlenül válasszuk ki egyik génjüket! Az így kapott génpár lesz az utód genotípusa. Végezzük el ezt N-szer, és töltsük fel az utódok táblázatát. Ha végeztünk, az így kapott táblázat legyen a szülők táblázata, és kezdjük újra a szimulációt! Ábrázoljuk a vércsoportok megoszlását!

7.160 

Az előző feladatot ki lehet bővíteni több jelenség szimulálásával:

A. Mutáció: a gén átadásánál bizonyos valószínűséggel hiba történik.
B. Migráció: a szülőtáblázatba új elemek jönnek.
C. Szelekció: a szülővé válás esélye függ a genotípustól.
D. Szimpátia: az elsőnek kiválasztott szülő genotípusától függ a másodiké.

Írjunk programot a fenti jelenségek szimulálására!

7.161 

Fejlesszük tovább a genetikai modellt a nemek figyelembevételével! (Vérzékenység, piros-zöld színtévesztés.)

7.162

Szokásos azt feltételezni, hogy a „mennyiségi" tulajdonságokat több lokusz alléljai additív módon határozzák meg. (Ezért nevezik ezeket poligénes tulajdonságoknak.) Ilyen tulajdonság pl. a magasság. Generáljunk K-szor N db egyed-magasságot befolyásoló allélpárt, és értékeljük ki az így kialakult populációt magasságeloszlás szerint!

7.163

Készítsünk programot adott számú lepke mutáció hatására történő színváltozásának szemléltetésére adott erősségű mutáció esetén!

7.164

Írjunk programot különböző tulajdonságú szarvasmarhák tulajdonságainak öröklődéséről! A program készítsen táblázatot, amelyből megállapítható a fenotípusok száma, azok %-os megoszlása, valamint a hozzájuk tartozó genotípusok száma.

7.165

Készítsünk programot, amely egy populáció növekedését szemlélteti

A. exponenciális,

B. logisztikus növekedés esetén!

7.166

Egy diploid genotípus egy lokuszán levő alternatív allélokat vizsgálunk (2 allél). Készíts táblázatot és grafikont az egyensúlyi genotípus–gyakoriságokról az egyik allél relatív gyakoriságának függvényében.

7.167

Módosítsuk az előző feladatot! Vegyünk figyelembe adott mutációs rátákat!

7.168

Módosítsuk az előző feladatot! Vegyünk figyelembe adott szelekciós együtthatókat (relatív rátermettség)!

7.169

Módosítsuk az előző feladatot! Rajzoljuk ki, s mutassuk meg, hogy szelekció esetén a populáció átlagos rátermettsége növekszik.

7.170

Egy diploid genotípus egy lokuszán levő alternatív allélokat vizsgálunk (3 allél). Készíts táblázatot és grafikont az egyensúlyi genotípus–gyakoriságokról a két allél relatív gyakoriságának függvényében.

7.171

Készítsünk táblázatot és grafikont beltenyésztett populációban a heterozigóták számának csökkenéséről! (A t. generációra a következő összefüggés igaz:

$$H = H_0 * \left(1 - \frac{1}{2} * N\right)^t .)$$

7.172

Egy idealizált ragadozó-zsákmány kölcsönhatást a Lotka-Volterra-egyenletek írják le. Készítsünk a két populáció létszámváltozását szemléltető programot!

7.173

K szigeten él egy-egy populáció, mindegyiken az elemi populációgenetikai modell (pl. a vércsoportok öröklődése) szerint. Az I. szigetről egy egyed kivándorlása az I-1.-re és az I+1.-re történhet B(I), illetve J(I) valószínűséggel. Készítsünk szimulációs modellt, amely követi a K szigeten történt eseményeket!

7.174

K szigeten él egy-egy populáció (pl. nyulak), mindegyiken az elemi demográfiai modell szerint. Az I. szigetről egy egyed kivándorlása az I-1.-re és az I+1.-re történhet B(I), illetve J(I) valószínűséggel. Készítsünk szimulációs modellt, amely követi a K szigeten történt eseményeket!

7.4 Kémia

A kémia területén – a számítógépes méréseken kívül – három fő alkalmazási területet különböztethetünk meg. Egyik a szimuláció, a másik kémiai számítások elvégzése, a harmadik pedig a gyakoroltatás.

7.175

Írjunk programot, amely a periódusos rendszer elemeiről néhány jellemzőt (ionizációs energia, atomsugár, sűrűség stb.) tárol, és ezekről a rendszám függvényében grafikont tud készíteni!

7.176

Az előző programot egészítsük ki úgy, hogy kérésünkre valamelyik jellemző szerint sorba rendezve írja ki az elemeket!

7.177

Készítsünk programot ionok meghatározására adott tulajdonságaik alapján, illetve adott ionok tulajdonságainak kiíratására (csapadéktáblázat).

7.178

Készítsünk programot bomlási folyamat szemléltetésére, amelyben egy anyag egy közbenső anyaggá bomlik, majd egy további anyag keletkezik! A bomlás jellemzőit hisztogramon és hidrodinamikai modellen ábrázolja. A bomlás sebessége változtatható legyen!

7.179

Készítsük el az A+B↔C kémiai reakciót szimuláló programot! A program folyamatosan jelenítse meg a reakciótér változásait, és készítsen grafikont a három anyag molekuláinak számáról!

7.180

Az előző szimulációs programban adjuk meg az egyesülés, illetve a bomlás valószínűségét is! Ha az átalakulás feltételei adottak és egy véletlenszám kisebb a valószínűségnél, akkor történjék átalakulás!

7.181

Fejlesszük tovább a szimulációt azzal, hogy az A és B közötti vonzást, illetve az azonosak közötti taszítást is figyelembe vesszük! Ha A szomszédja üres, de abban az irányban a második szomszédban B van, akkor közelebb vonzza. Ha a szomszédja A és a második szomszéd abban az irányban üres, akkor eltaszítja. B-re ugyanez igaz.

7.182

A következő reakciókkal egy katalizátor működését szimulálhatjuk: A+X→B; B→C+Y; C→A. X és Y szabadon ki-be áramolhat. Írjunk ilyen szimulációs programot!

7.183

Az előző feladatot módosítsuk úgy, hogy az utolsó reakcióban C-ből két A molekula keletkezzék!

7.184

Készítsünk kémiai szimulációs algoritmust az A+B↔C reakció szimulálására úgy, hogy az A és a B molekulák gőzfázisban vannak, a keletkező C termék pedig folyadékfázisban!

7.185

Készítsünk kémiai szimulációs algoritmust az A+B ↔ C reakció szimulálására úgy, hogy az A és a B molekulák folyadékfázisban vannak, a keletkező C termék pedig gőzfázisban!

7.186

Készítsünk kémiai szimulációs programot az A+B↔C reakció szimulálására, ha A és C szilárd anyag, a helyéről elmozdulni nem tud, B pedig gázmolekula. A szimulációs tér felső, bal és jobb oldala a B molekulák számára átjárható.

7.187

Készítsünk kémiai szimulációs programot az A+B↔C reakció szimulálására gravitációval, ha A és C szilárd anyag, a helyéről elmozdulni nem tud, B pedig folyadékmolekula. A szimulációs tér bal és jobb oldala a B molekulák számára átjárható.

7.188

Egy szimulációs tér alján szénatomok találhatók, fölötte pedig oxigénmolekulák. Egyetlen atomot „meggyújtunk", azaz a molekula és egy szomszédos oxigénmolekula széndioxid-molekulává alakul, miközben a szomszédságában levő szénatomok közül kettőt felmelegít. A meleg szénatomok oxigénmolekulával találkozva, ugyanígy viselkednek. Készítsük el a modell szimulációs lépését!

7.189

Készítsünk gyakoroltató programot, amellyel a periódusos rendszer megismerését segíthetjük elő. A program mutasson rá egy helyre a periódusos rendszerben, s a felhasználónak kelljen megadni az ott található atomot, fontosabb jellemzőit!

7.190

Oldjuk meg az előző feladatot fordítva is: egy – nevével vagy jellemzőivel – megadott atomot kelljen elhelyezni a periódusos rendszerben!

7.191

Készítsünk programot, amely 1 mólnyi ideális gáz nyomása, térfogata, illetve hőmérséklete közül bármelyik kettőből ki tudja számítani a harmadikat!

7.192

Megadtuk egy szénhidrogén szerkezeti mátrixát, amelyben az I. sor J. oszlopában az I. és a J. atom közötti kötésszám található, illetve 0, ha közöttük nincs kötés!

1. Adjuk meg a szén, illetve hidrogénatomok számát!
2. Döntsük el, hogy a mátrix helyesen van-e kitöltve (csak 1 és 4 kötésszámú atom van, s a mátrix egyetlen – összefüggő – molekula mátrixa)!
3. Mondjuk meg, hogy aromás szénhidrogénről van-e szó!
4. Döntsük el, hogy gyűrűs szénhidrogénről van-e szó!
5. Döntsük el, hogy telítetlen szénhidrogénről van-e szó!

6. Adjuk meg a szénhidrogén azon atomjait, amelyek gyűrűben vannak vagy gyűrűket kötnek össze!

7.193

Egy hidrogénből, jódgőzből és hidrogénjodidból álló elegyet vizsgálunk. Számítsuk ki, adott reakciósebességek mellett, időegységenként az egyes molekulák koncentrációit!

7.194

Egy többlépéses reakció esetén, adott reakciósebességek mellett, számítsuk ki az egyes molekulák koncentrációit!

7.195

Szimuláljuk a Millikan-kísérletet az elemi töltés meghatározására!

7.196

Szimuláljuk a galvánelem működését!

7.197

Készítsünk programot, amelynek segítségével a felhasználó galvánelemet állíthat össze elektródok, illetve oldatkoncentrációk alapján, majd a program megadja az áramerősséget, irányt!

7.198

Készítsünk számítógépes összerakójátékot (puzzle) az atommag szerkezetének szemléltetésére! A felhasználó tetszőleges módon tehessen össze protonokat, neutronokat, s a számítógép értékelje az így kapott atomot: mit sikerült összeállítani, miért nem stabil izotóp, ...

7.199

Készítsünk számítógépes összerakójátékot (puzzle) az elektronszerkezet szemléltetésére! A felhasználó helyezhessen el tetszőlegesen elektronokat, s a program adja meg, mit sikerült létrehoznia, mi miért nem szabályos!

7.200

Készítsünk számítógépes összerakójátékot (puzzle) szénhidrogének molekulaszerkezetének szemléltetésére! A felhasználó atomokból, atomcsoportokból építkezhet, s a program közölje, hogy mit sikerült alkotnia. Elképzelhető olyan változat is, amikor a program közli a célt, s a felhasználónak ki kell egészítenie egy félkész molekulát vagy pedig teljesen elölről kell felépítenie!

7.201 

Készítsünk gyakoroltató programot szénhidrogének elnevezésének gyakoroltatására! Konstitúciós képlet alapján kérdezze meg a molekula nevét, illetve név alapján kelljen megadni a konstitúciós képletet!

7.5 Technika

A szimulációs programokból játékot is csinálhatunk. Ha például a felhasználó valamelyik vasúti programban a forgalmista, akkor minden vonat helyes áthaladásáért kapjon pontokat, de a torlódások járjanak pontlevonással! Ha olyan rosszul irányítja a jelzőlámpákat, hogy baleset történik, küldje kényszerpihenőre, és veszítse el pontjait!

7.202 

Készítsünk programot, amely egy emeletes házban működő liftet szimulál! A ház legyen pl. 9 emeletes, a liftbe egyszerre legfeljebb hatan szállhatnak be. Az utasok túlnyomó többsége a saját emeletéről a földszintre, illetve vissza akar menni, és csak olyankor száll be a liftbe, ha az jó irányba megy. Hogy hol jelenik meg újabb utas, azt előre beállított valószínűséggel a véletlen határozza meg. Figyeljük emeletenként a várakozók számát!

7.203 

Módosítsuk az előző programot a következő lehetőségekkel:

1. a liftnek stratégiája van (pl. lefelé utasgyűjtő),
2. a lift irodaházban van, és mind az induló- mind a célállomás véletlenszerű,
3. a lift hívásakor az utas közli, hogy merre akar menni,
4. két lift is rendelkezésre áll, de a hívógombjuk közös!

7.204 

Szimuláljuk egy páternoszter működését! Minden emeleten véletlenszerűen megjelenő utasok véletlenszerű irányba akarnak menni.

7.205 

Készítsünk programot, amely egy útkereszteződés forgalmát szimulálja! A program állítsa automatikusan a lámpákat, a véletlenszerűen érkező autók mennyiségétől függően számítsa ki a váltási időket, jelenítse meg az autókat!

7.206 

Szimuláljuk egy főútvonal forgalmát! Adott az út hossza, a jelzőlámpák távolsága, a ki- és behajtási helyek. Lehessen beállítani a jelzőlámpák programját, a ki- és behajtások gyakoriságát, a keresztirányú forgalom gyakoriságát! A prog-

ram jelenítse meg az út pillanatnyi képét, az egyes lámpáknál várakozó autók számát, autónként a megállások számát és az átlagos áthaladási időt!

7.207

Szimuláljuk egy körforgalom forgalmát! Legyen több be- illetve kihajtási lehetőség!

7.208

Írjunk programot, amely egy vasútvonal forgalmát szimulálja! Legyen a vonalon N állomás, ezeken néhány vágány, az állomások közötti egyetlen vágányon közlekedjenek gyors-, személy- és tehervonatok!

7.209

Írjunk programot egy teherpályaudvar forgalmának szimulálására! Az ide érkező tehervonatokat szét kell bontani, más szerelvényeket kell összeállítani, és a lehetséges négy irányba továbbítani.

7.210

Írjunk programot egy metróállomás forgalmának szimulálására! Feltételek: két irányba induló vonatok, két mozgólépcső a hosszával és sebességével, az állomásra beférő utasok száma. Lehessen beállítani az utasok érkezésének gyakoriságát, irány szerinti eloszlását, a leszállók átlagos számát, a szerelvények követési idejét, a szerelvényen utazók maximális számát. A program jelezze a mozgólépcsőkre, illetve a szerelvényekre várakozók számát (irányonként), a mozgólépcsőn utazók számát és a felszállók számát!

7.211

Fejlesszük tovább az előző programot! Menet közben lehessen módosítani a követési időközt, lehessen a szükséges irányba be-, illetve kikapcsolni a harmadik mozgólépcsőt. A program figyelmeztessen a torlódásokra, kövesse nyomon a mozgólépcsők kihasználtságát!

7.212

Írjunk szimulációs programot egy repülőtér forgalmának modellezésére! Adott a menetrend és két (három) kifutópálya. Különböztessünk meg két esetet:

- minden gép pontosan érkezik, illetve indul;
- minden gép indulásának, illetve érkezésének a menetrendben előírttól legfeljebb 5 perccel szabad eltérnie (de ennyivel előbb vagy később is lehet)!

7.213

Készítsünk programot a morzejelek gyakorlására! A hangszóró vagy a hangkártya felhasználásával játsszon el előre adott ütemben egyet a memóriájában tárolt néhány szöveg közül!

7.214 

Készítsünk KRESZ-programot! A program néhány útkereszteződést tárol, ezek közül választ egyet, táblákat vagy lámpákat és járműveket helyez el benne! A felhasználónak kell megadnia az áthaladási sorrendet, ezt a gép értékelje!

7.215 

Készítsünk dinamikus modellt a hőtárolós kályha működésére! Az energia szakaszosan érkezik a hálózatból. Éjszakai áram esetén naponta egyszer, hangfrekvenciás körvezérlésnél véletlen eloszlásban. Az energiát akkor adja át a szobának, ha a levegő hőmérséklete egy küszöb alá csökken. A szoba levegője folyamatosan ad energiát a kinti (napszakonként változó hőmérsékletű) levegőnek. A két utóbbi energiaátadás sebessége a hőmérséklet-különbséggel arányos.

7.216 

Készítsünk programot a kondenzátor energiatároló szerepének szemléltetésére! Az $\left|\sin\left(\frac{2*\pi*t}{T}\right)\right|$ szerinti bemenő feszültség az egyenirányító belső ellenállásán keresztül tölti a kondenzátort, a kisütő áram pedig állandó. Legyen lehetőség a különböző kisütő áramokhoz és a szűrés megfelelő minőségéhez tartozó kapacitások meghatározására!

7.217 

Az előző programban próbáljuk ki a véletlenszerűen, illetve a periodikusan változó kisütő áramot!

7.6 Egyéb

Ebben a fejezetben többféle tantárgy anyagával kapcsolatos feladat szerepel. Úgy gondoljuk, hogy a számítógép szinte minden tantárgy keretében használható, tehát javaslunk hozzá feladatot. Így szerepel itt a nyelvtan, az irodalom, idegen nyelvek, a történelem, általában a társadalmi jelenségek, a földrajz és a műszaki tantárgyak.

Ez a sok tantárgy mégis egy részbe került. Ezt az indokolja, hogy még egyikből sem gyűlt össze olyan mennyiség, ami különválasztásukat indokolná.

7.6.1 Magyar nyelv

7.218 

Írjunk programot, amely a kötőszavak alapján megállapítja, hogy milyen típusú mellérendelő összetett mondatról van szó! A program kérje be a mondatot, külön megjelölve benne a tagmondatokat összekapcsoló kötőszót (többszörös összetétel esetén minden ilyet), majd írja ki az összetétel típusát!

7.219 

Készítsünk programot az ly és j használatának gyakoroltatására! Legyenek kiegészítendő szavak vagy a hiányos szavak közül az ly-osak sorszámát kelljen megadni!

7.220 

Készítsünk egyszerű (tehát nem összetett) szavak elválasztását gyakoroltató programot! (A szavakat a program tárolja, közülük véletlenszerűen választ.)

7.221 

Készítsünk verselemző programot időmértékes versek skandálására!

7.222 

Készítsünk vers hangtani elemzésére szolgáló programot az alábbiakkal:

A. magas és mély hangrendű szótagok eloszlása,
B. magas, vegyes és mély hangrendű szavak eloszlása,
C. zöngés és zöngétlen mássalhangzók eloszlása.

7.223 

Készítsünk vers ritmikai elemzésére szolgáló programot az alábbi funkciókkal:

A. hosszú és rövid szótagok eloszlása,
B. ritmikai elemek (daktilus, spondeus stb.) eloszlása,
C. szóhosszúságok eloszlása.

7.224 

Készítsünk vers gyakorisági elemzésére szolgáló programot az alábbi funkciókkal:

A. magas és mély hangrendű szótagok száma,
B. magas, vegyes és mély hangrendű szavak száma,
C. zöngés és zöngétlen mássalhangzók száma.
D. hosszú és rövid szótagok száma,
E. soronkénti szótagszám,
F. soronkénti szószám.

7.225 

Készítsünk programot egy vers rímképletének meghatározására!

7.6.2 Idegen nyelv

7.226 

Készítsünk szótárprogramot valamely idegen nyelvhez!

7.227 

Készítsünk idegen szavak tanulását segítő programot! Hol az egyik, hol a másik nyelvű szót írja ki a gép, és a másik nyelvűvel kelljen válaszolni! A program hagyjon javítási lehetőséget, és ismerje fel a helyes választ, hibás válasz esetén árulja el a helyeset! Adjon lehetőséget a szókincs menet közben történő bővítésére is!

7.228 

Készítsünk olyan programot, amelyikben idegen nyelvű szöveget kell kiegészíteni adott lehetőségek közül eggyel! Gyakoroltathatunk ragozást, igeidőt, névelőt, egyeztetést stb.

7.229 

Készítsünk nyelvi összerakójátékot! A játék elemei betűk, amelyeket helyes sorrendbe kell rakni (több helyes sorrend is lehet).

7.230 

Készítsünk nyelvi összerakójátékot! A játék elemei szavak, amelyeket helyes sorrendbe kell rakni (több helyes sorrend is lehet).

7.231 

Készítsünk nyelvi összerakójátékot! A játék elemei mondatok, amelyeket helyes sorrendbe kell rakni (több helyes sorrend is lehet).

7.232 

Készítsünk nyelvi kiegészítőjátékot, melyben adott szöveget kell kiegészíteni hiányzó szavakkal, adott szókészletből.

7.233 

Készítsünk nyelvi kiegészítőjátékot, melyben adott szöveget kell kiegészíteni hiányzó betűkkel, ha minden K. hiányzik.

7.6.3 Történelem, társadalom

7.234 

Írjunk programot, amely egy ország népességét szimulálja! Vegyük figyelembe a járványokat, háborúkat és a gazdasági helyzet alakulását!

7.235 

Írjunk programot, amely több ország adott időszakra eső történelmi eseményeit tárolja. Kérésre időrendben egymás mellé helyezve a kért néhány ország eseményeit írja a képernyőre!

7.236 

Készítsünk tesztprogramot a következő típusú tudáselemek kikérdezésére:

1. összetartozó fogalmak kiválasztása,
2. események és évszámok kapcsolata,
3. kronológiai sorrend megállapítása,
4. szakkifejezések és meghatározások párosítása!

7.237 

Készítsünk tesztprogramot nevezetes események helyének vaktérképen való megjelölésére, illetve a megjelölt helyhez fűződő esemény megnevezésére!

7.238 

Írjunk programot az Árpád-házi királyok uralkodási idejének gyakoroltatására! Kérdezze a program a király uralkodási idejét, elődjét és utódját a trónon, valamint az évszámhoz az uralkodó nevét!

7.239 

Készítsünk szimulációs programot a földbirtokok öröklődésének, szétaprózódásának szemléltetésére a középkorban! Induljunk ki egy adott birtokeloszlásból, játsszuk le az öröklést különböző szabályok szerint:

- a legidősebb fiú örököl,
- minden fiú örököl egyenlő részt,
- minden fiú örököl, de esetleg különböző részt,
- a király véletlenszerűen adományoz birtokot,
- a király annak ad birtokot, akinek nincs semmije,
- a király nagyobb valószínűséggel ad birtokot annak, akinek korábban is nagy birtoka volt,
- a király elkoboz birtokokat,
- ... (más szabályokkal is színesíthetjük a feladatot).

Ha külön szabály nem intézkedik, a birtok visszaszáll a koronára (bár ettől eltérően is intézkedhetünk).

7.240

Módosítsuk úgy az előző feladatot, hogy a lányok is örökölhessenek, s így házassággal is lehet birtokot szerezni!

7.241

Szimuláljuk egy megye falvai lakosságának alakulását! Ez egyrészt a születésektől, halálozásoktól függ, másrészt pedig a költözésektől. A költözést befolyásolja, hogy van-e a faluban iskola, munkalehetőség, KÖZÉRT, orvos stb. Ezeket viszont az befolyásolja, hogy hányan laknak a faluban. Egy falu akkor is kedvelt lakóhely lehet, ha közel van város, jók a közlekedési lehetőségek, kellemes üdülőhely stb.

7.242

A Föld lakosságának – idealizált – növekedését a következő egyszerű egyenlet írhatja le:

$$N' = (Sz - H) * N,$$

azaz a létszámváltozás a létszámmal arányos születésekből (Sz*N), valamint a szintén létszámmal arányos halálozásokból (H*N) származik. Írjunk programot, amely adott születési és halálozási ráták esetén bemutatja a Föld lakosságának változását! (Az ilyen modelleket hívják világmodellnek.)

7.243

Módosítsuk az előbbi feladatot úgy, hogy a születési rátát csökkentse a környezetszennyezés, a halálozási rátát pedig növelje a környezetszennyezés, az élelmiszerhiány, az energia- és nyersanyagforrások kimerülése!

7.244

Vezessünk be az előbbi modellbe természeti katasztrófákat, születésszabályozást stb.!

7.245

Az ipari termelés növekedésének bevezetésével kapcsoljuk össze a korábbi példákban szereplő hatóerőket! Az ipari termelés növeli a környezetszennyezést, csökkenti a nyersanyagtartalékokat.

7.6.4 Földrajz

7.246 

Írjunk programot, amely kikérdezi a felhasználót, hogy melyik város, melyik megyében van! Keressünk grafikus megoldást is a programhoz!

7.247 

Készítsünk programot, amely egy adott vaktérképen a gép által kiírt város helyének meghatározását értékeli!

7.248 

Írjunk programot az országok fővárosainak tanítására! A programnak legyen egy oktató és egy kikérdező része! A kikérdezéskor kérdezzen országgal és fővárossal is! A program értékelje is a teljesítményt!

7.249 

Készítsünk programot, amely kirajzolja az országot a megyehatárokkal együtt, és kikérdezi a megyeszékhelyeket!

7.250 

Készítsünk gazdaságföldrajzi tesztet adott ország jellemző tulajdonságainak meghatározására!

7.251 

Európában N helyen mértük a napi hőmérsékletet. Rajzoljunk ezen adatok alapján izotermatérképet!

7.252 

Írjunk programot, amely tárolt adatok alapján kirajzolja Európa függőleges metszetét a kívánt földrajzi szélesség vagy hosszúság mentén!

7.253 

Egy mátrixban tároljuk az ország egy részletének műholdtérképét. Minden egyes elem az adott pont fényességét tartalmazza 0 és 255 között. Készítsünk programot, amely kirajzolja a képernyőre az összes, adott fényességnél fényesebb pontot (ún. intenzitásvágás)!

7.254 

Egy mátrixban tároljuk az ország egy részletének műholdtérképét. Minden egyes elem az adott pont fényességét tartalmazza 0 és 255 között. Készítsünk programot, amely megadja a felhővel borított terület nagyságát!

7.255

Egy mátrixban tároljuk az ország egy részletének műholdtérképét. Minden egyes elem az adott pont fényességét tartalmazza 0 és 255 között. Készítsünk programot, amely megadja az éleket (a különböző területek határvonalait – ahol az intenzitás nagyot változik – ilyenek például a folyók, a tavak határvonalai stb.)!

7.256

N növényfajról tároljuk termesztésének optimális feltételeit. Írjunk programot, amely bekéri egy adott terület éghajlati adatait (hőmérséklet, csapadék stb.) és a földrajzi fekvést (szélességi kör, tengerszint feletti magasság stb.) majd kiírja az azon a helyen optimálisan termeszthető növényeket, ha ilyen nincs, akkor azokat, amelyek az optimálistól a lehető legkisebb mértékben térnek el!

7.257

Írjunk programot, amely legfeljebb öt kérdésre adott igen/nem válasz alapján kitalálja, hogy melyik megyére gondoltunk!

7.6.5 Testnevelés

7.258

Írjunk programot, amely a testnevelésben használatos piktogramok segítségével összeállít adott elemszámú (végrehajtható) gimnasztikai gyakorlatot!

7.259

Írjunk programot, amely egy testnevelésóra gyakorlatait állítja össze, ha ismert a lehetséges gyakorlatok köre, azok típusai, nehézsége, időigénye! Az óra legyen változatos gyakorlattípusban és nehézségben is!

7.6.6 Ipari középiskolai tantárgyak

7.260

Négy pólusú indukciós motort hat pólusú szinkron generátor táplál. A generátor fordulatszáma percenkénti N. Az indukciós motor szlipje S%. Írjunk programot, amely különböző bemenő adatok (N és S) mellett meghatározza a motor fordulatszámát!

7.261

Írjunk programot, amely elvégzi egy deltakapcsolás csillagkapcsolássá való átalakítását! A program bemenő adata legyen a deltakapcsolás három ellenállása, kimenő adata pedig a csillagkapcsolás három ellenállása.

7.262

Írjunk programot, amely elvégzi egy csillagkapcsolás deltakapcsolássá való átalakítását! A program bemenő adata legyen a csillagkapcsolás három ellenállása, kimenő adat pedig a deltakapcsolás három ellenállása.

7.263

Egy papírszigetelésű kondenzátor két fegyverzete B=40 mm széles, L=5 m hosszú fémfóliából áll. A fegyverzeteket egymástól D=0.03 mm vastagságú kondenzátorpapír választja el, amelynek relatív dielektromos állandója E=2.5 . Mekkora ennek a kondenzátornak a kapacitása, ha

A. a fegyverzetek síkban vannak kiterítve,

B. a fegyverzetek henger alakban fel vannak tekerve?

Írjunk programot, amely a fenti feladatot különböző bemenő adatok (B, L, D, E) mellett meg tudja oldani!

7.264

Egy váltakozó áramról táplált adott ellenállású fűtőbetét teljesítményét tirisztorral szabályozzák. Számítsuk ki a teljesítményt annak függvényében, hogy milyen feszültségnél gyújt be a tirisztor!

7.265

Az előző feladat megoldásánál vegyük figyelembe, hogy a teljesítmény növekedésével a fűtőbetét hőmérséklete is nő, ezért az ellenállása változik!

7.266

Készítsük el az alábbi feladat bármely (reális) adattal történő kiszámítására használható programot! Egy centrifugálszabályzó karjai L hosszúságúak, fordulatszáma N. Mekkora a rudak függőlegessel bezárt szöge?

7.267

Rajzoltassuk fel a képernyőre a Diesel-körfolyamat elméleti p-V diagramját! Valamilyen módon jelöljük az égés helyén az energia-bevezetést, az expanzió végén a hőelvezetést!

7.268

Írjunk programot, amely az Otto-körfolyamat termikus hatásfokának (közelítő) meghatározását végzi el! A program irreális kompresszióviszony adatok esetén figyelmeztesse erre a felhasználót!

7.269

Adott egy kéttámaszú tartó, amelyre N db erő hat. Adott mindegyik erő támadáspontja. Határozzuk meg a nyíróerő és a nyomaték értékét

A. egy adott x helyen,

B. ha dx lépésközzel végigmegyünk a tartón!

7.270

Egy kéttámaszú tartó esetében számítsuk ki programmal a támaszerőket egyenlőtlenül megoszló teher esetén!

7.271

Egy kéttámaszú tartó esetén készítsünk:

1. kötélábrát,
2. nyomatéki ábrát,
3. nyíróerő-ábrát!

7.272

Egyik végén befogott vízszintes tartó végpontjában egy F függőleges erő hat. Készítsünk programot a tartó lehajlásának kiszámítására! Az eredmény egy táblázat legyen, amelyben a szabad végtől való távolság 0-tól dx lépésenként változik Xv-ig, F értéke pedig F0-tól dF lépésenként Fv-ig.

7.273

Szimuláljuk számítógép segítségével a gravitációs ülepítés folyamatát! Legyen megadható az ülepítendő részecske átmérője, sűrűsége, a közeg viszkozitása, sűrűsége, a folyadékszint magassága!

7.274

Szimuláljuk számítógép segítségével a szűrés folyamatát! Legyen megadható a szűrőfelület, valamint az alkalmazott nyomáskülönbség!

7.275

Készítsünk szimulációs programot ipari szárítóberendezések működésének bemutatására! Legyen megadható a szárítandó anyag, a szárítólevegő, a fűtőgőz tömegárama, a fűtőgőz nyomása!

7.276

Készítsünk programot, amely egy csővezetékben az egyes szakaszok áramlási sebességeit számolja!

8. FEJEZET

AZ ISKOLAI ÉLET SZERVEZÉSE

8. AZ ISKOLAI ÉLET SZERVEZÉSE

A fejezet elkészítésekor abból indultunk ki, hogy a feladatgyűjteményt elsősorban iskolák fogják használni. Ellentétben a legtöbb fejezettel, itt zömében olyan feladatokat találnak, amelyek nem annyira a számítástechnika oktatásához, inkább alkalmazásához igyekszenek ötleteket adni. Megpróbáltuk összegyűjteni valamennyi, az iskolai élettel összefüggő területről azokat a problémákat, amelyek számítógépes megoldása célszerű, tanulságos vagy jelentős munkát takarít meg.

Felhívnánk a számítástechnikát tanító tanárok figyelmét, hogy ezek között a feladatok között több olyat találnak, amelyek segítséget nyújthatnak meggyőzni a hozzá nem értőket a számítógépek hasznosságáról és fontosságáról. Néhány az érintett témák közül:

- órarend-készítés támogatása,
- helyettesítések,
- bérek, jutalmak,
- szociometria,
- iskolai versenyek,
- személyi nyilvántartás,
- leltárok,
- statisztikák.

A megjegyzésekben elsősorban a feladatokkal kapcsolatos elvárásokra tettünk néhány utalást.

Még egy általános megjegyzés a feladatokban szereplő nyilvántartás készítésekkel kapcsolatosan: Minden „élesben" használható nyilvántartó rendszer legalább három (de inkább 4-5) részprogramból áll. Az egyik, amivel létrehozunk, illetve feltöltünk egy adatbázist, egy, amivel karbantartjuk, és egy, amivel különböző kereséseket végezhetünk benne.

Minden olyan feladatnál, amelyben valamilyen nyilvántartás létrehozásáról van szó, tartsuk szem előtt ezeket a megoldandó feladatokat!

Ne feledkezzünk meg továbbá a barátságos, egyértelmű képernyőtervezésről, valamint az adatok biztonságos tárolásáról, az adatmentés fontosságáról sem!

8.1 

Az ellátandó óraszámok ismeretében írjunk a tantárgyfelosztás készítését segítő programot!

8.2 

Készítsünk programot, amely ismert tantárgyfelosztás esetén a kieső tanár állandó helyettesítésére tesz javaslatot (legyen egyenletes a túlóraelosztás)!

8.3 

Az adott tantárgyfelosztás ismeretében írjunk programot, amely a következő évi tantárgyfelosztás felmenő rendszerben történő elkészítéséhez nyújt segítséget (emeljük ki a szétosztandó órákat)!

8.4 

Írjunk órarendkészítést támogató programot! A program kérjen a felhasználótól javaslatokat az órák elhelyezésére, ne engedje hibás órarend elkészítését, kérésre közölje, hol van még szabad lehetőség (terem, tanár, stb.), adjon módot órák áthelyezésére stb.!

8.5 

Készítsünk órarend-nyilvántartó programot memóriatakarékosan, amely tud válaszolni a következő kérdésekre:

A. Melyik tanárnak, mikor és hol van órája?
B. Egy adott helyen kinek van órája?
C. Egy adott időben egy adott osztálynak milyen órája van és hol?

8.6 

Írjunk programot, amely ismert órarend esetén javaslatot tesz bizonyos órára helyettes tanár személyére a következő szempontok figyelembevételével:

- szakszerű legyen a helyettesítés;
- ha nem az, akkor az osztályban tanító tanárt jelöljön ki;
- a helyettes tanárnak a lehető legkevesebb lyukasórája legyen!

8.7 

Készítsünk órarend-ellenőrző programot! A program a következő funkciókat ismerje: (bemenő adatok: tanár neve, tantárgy, osztályterem, időpont).

A. Adjuk meg egy osztály órarendjét!
B. Adjuk meg egy tanár órarendjét!
C. Adjuk meg egy terem beosztását!
D. Ellenőrizzük, hogy nincs-e ütközés az órarendben!

8.8 

Írjunk algoritmust, amely ismert órarend esetén javaslatot tesz óracserékre:

A. adott órát bármikorra,

B. adott órát adott napra,

C. bárhonnan egy adott órára!

8.9 

Kész órarend alapján írjunk terembeosztást készítő programot, biztosítva a szaktantermek egyenletes kihasználtságát (minden osztály a lehető legnagyobb óraszámban kerüljön oda)!

8.10 

Ismert terembeosztás esetén keressük meg a legalkalmasabb tantermet egy rendkívüli foglalkozáshoz:

A. létszám szerint,

B. eszközigény szerint!

8.11 

Készítsünk programot, amely nyilvántartja az iskolában oktató tanárok személyi adatait, szolgálati éveinek számát, kulcsszámát, kitüntetéseit, alapfizetését, pótlékait, óradíját!

8.12 

A tanárok adatait nyilvántartó program és az ellátandó óraszámok ismeretében írjunk algoritmust, amely javaslatot tesz arra, hogy milyen szakos tanárt keressünk, maximált fizetést figyelembevéve!

8.13 

Készítsünk programot, amely adott összegű fizetésemelést szétoszt a tanárok között úgy, hogy a nettó béremelések aránya meghatározott legyen!

8.14 

Készítsünk programot, amely adott összegű jutalmat szétoszt a tanárok közt úgy, hogy a nettó jutalmak aránya meghatározott legyen!

8.15 

Készítsünk programot, amely az életkor és a nettó fizetés közti kapcsolatot közelíti és az attól való abszolút, relatív eltérés szerint sorbarendezi a tanárokat!

A. Adott egyenessel.

B. Regressziós egyenessel.

C. Más adott vagy számolható függvénnyel.

8.16 

Készítsünk programot, amely javaslatot tesz érettségin, versenyen stb. felügyelő tanárok személyére az egyenletes leterhelés, szakos megkötöttség, személyi adottságok figyelembevételével.

8.17 

Készítsünk programot, amely nyilvántartja az iskola tanulóinak személyi adatait (név, személyi szám, osztály, tanult tantárgyak, és a kapott osztályzatok). Tudjunk a programmal a nyilvántartott adatokból különböző szempontok alapján keresni, listázni!

8.18 

Készítsünk programot, amellyel évenként aktualizálni lehet a tanulónyilvántartást!

8.19 

Készítsünk programot, amely egy osztály félévi, évvégi értékelését végzi el! Tudjon számolni:

A. osztály-tantárgyi átlagokat,
B. osztályátlagot,
C. tantárgyankénti bukásarányt,
D. osztályban bukásarányt,
E. egyéb, osztálystatisztikához szükséges jellemzőket,
F. rendezze különböző szempontok szerint az osztály tanulóit!

8.20 

Készítsünk programot, amely az iskola félévi, évvégi értékelését végzi! Tudja számítani:

A. egy osztály tantárgyi átlagát,
B. osztályátlagot,
C. tantárgyankénti bukásarányt,
D. az iskola tantárgyi átlagait,
E. teljes iskolaátlagot,
F. tantárgyankénti bukásarányt!

8.21 

Rendezzük különböző szempontok szerint (tanulmányi átlag, hiányzás stb.) sorrendbe az iskola tanulóit:

A. osztályonként,
B. évfolyamonként,
C. iskolaszinten!

8.22 

Veszélyeztetett és hátrányos helyzetű tanulók és a velük kapcsolatos tennivalók nyilvántartására és nyomon követésére készítsünk programot!

8.23 

Készítsünk programot, amely a diákkörök eszközállományát és pénzügyi helyzetét követi nyomon, napra készen!

8.24 

Írjunk programot, amely a szülők keresetének, a szociális helyzetnek, a tanulók tanulmányi eredményének figyelembevételével megállapítja a menzadíjat!

8.25 

Készítsünk programot, amely osztályonként a napi tej, kakaó, péksütemény mennyisége alapján kiszámítja:

A. a havi pénzbefizetést osztályonként,
B. a havi rendeléseket a tej- és a sütőiparnak, naponkénti bontásban,
C. a havi összesített számlát a tejiparnak és a sütőiparnak!

8.26 

Készítsünk programot az iskola eszközállományának a nyilvántartására szakleltáronként, értékek szerint (állóeszköz, fogyóeszköz)!

8.27 

Írjunk algoritmust, amely nyilvántartja a fejlesztési igényeket!

8.28 

Készítsünk programot, amely az eszközállomány, a fejlesztési igények és a rendelkezésre álló pénzmennyiség ismeretében javaslatot tesz a beszerzésekre! (Tartalmazhat több alternatívát is).

8.29 

Írjunk programot, amely az iskolai könyvtár anyagát tárolja, (cím, szerző, kölcsönzés ténye, időpontja, lejárat ideje bontásban). Tudjunk a programmal a tárolt adatok alapján keresni, és listát készíteni!

8.30 

Készítsünk programot, amely a kölcsönzéseket napra készen követi nyomon, figyelmeztet a kölcsönzési határidő lejártára, várakozó listát készít!

8.31 

Írjunk programot, amely a könyvállomány, a beszerzési igények és a pénzmennyiség ismeretében javaslatot tesz a beszerzésekre, és figyelmeztet a vár--

ható megjelenés előtt a megrendelés elküldésére! (Tartalmazhat több alternatívát is!)

8.32 

Készítsünk kísérletekhez szükséges eszközösszeállító programot!
Bemenő adatok:

- a kísérlet azonosítója,
- demonstrációs vagy tanulókísérlet,
- a csoport létszáma.

Kimenő adatok:

- eszközök neve, darabszáma,
- a tanulókísérleteknél csoportosítás,
- az összes eszköz mennyisége.

8.33 

Készítsünk programot, amely az előző feladathoz szükséges adatbázist létrehozza!

8.34 

Írjunk programot, amely az I. osztályos középiskolai tanulók érdemjegyeit összeveti az általános iskolai érdemjegyekkel és az eltéréseket különböző szempontok alapján értékeli (tanár, tantárgy, tanuló, általános iskola ...)!

8.35 

Készítsünk programot, amely a középiskolába jelentkező tanulókat érdemjegyeik alapján rangsorolja (kiemelt tárgyak, javuló/romló tendencia, meghatározott maximált pontszám)!

8.36 

Készítsünk programot, amely egy osztály tanulóinak és érdemjegyeinek ismeretében felelőt választ ki!

8.37 

Írjunk keretprogramot, amelynek segítségével tetszőleges témájú felelet-választós feladatlapot készíthetünk!

8.38 

Készítsünk programot, amely adott feladatlap alapján vizsgáztatja a tanulókat! (A kérdéseket kiírja a képernyőre és értékeli a billentyűzetről bevitt válaszokat!)

8.39 

Készítsünk programot, amely egy feladatsor pontszámai alapján osztályzási javaslatot tesz:

A. adott ponthatárokkal,
B. adott átlagra adott intervallumhatárokkal,
C. százalékos teljesítmények alapján,
D. teljes megoszlás alapján,
E. egyéb szempontok alapján!

8.40 

Készítsünk programot, amely egy osztály ülésrendjét tárolja!

8.41 

Készítsünk programot, amely a tanulók testmagassága, neme, esetleges egészségügyi problémáik figyelembevételével ülésrendet készít!

8.42 

Készítsünk programot, amely elkészíti az osztály riadóláncát készít telefonszám és lakhely ismeretében úgy, hogy akármelyik tanulótól indulva a legrövidebb idő alatt jusson el a hír mindenkihez!

8.43 

Írjunk programot, amely a korábbi kirándulások, a kirándulás időtartama, az előre meghatározott lehetséges látnivalók és a pénzügyi lehetőségek ismeretében javaslatot tesz a kirándulás útvonalára és a közlekedési eszközökre!

8.44 

Írjunk programot, amely egy kirándulás útvonaltervének ismeretében az időbeosztás és a költségvetés elkészítését segíti!

8.45 

Készítsünk programot, amely kitöltött szociometriai lapok feldolgozását segíti!

8.46 

Készítsünk programot, amelynek segítségével szociometriai felméréseket lehet végezni! A program értékeljen is!

8.47 

Készítsünk programot, amely az ismert adatok felhasználásával szociometriai hálót készít, felhívja a figyelmet a periférikus gyerekekre!

8.48 

Készítsünk programot, amely nyilvántartja egy tanuló házi feladatait! Bemenő adatok: órarend, iskolán kívüli elfoglaltság, alkalmi program, házi feladatra szükséges idő.

8.49 

Készítsünk élő határidőnaplót!

8.50 

Készítsünk programot, amely a tanuló betegsége esetén tanácsot ad, hogy kitől kérdezze meg az egyes tantárgyak házi feladatait vagy tananyagait!

8.51 

Készítsünk programot, amely az iskolai tanulmányi mozgalom értékelését végzi!

8.52 

Készítsük el az iskolai labdarúgó (kézilabda stb.) bajnokság sorsolását! Mindenki játsszon mindenkivel, páratlan csapatszám esetén 1 csapat mindig szabadnapos.

8.53 

Készítsünk programot, amely az iskolai bajnokság eredményeit nyilvántartja! Minden forduló után táblázatot készít a bajnokság állásáról.

8.54 

Írjunk programot, amely nyilvántartja a különböző postai befizetéseket! A határidő lejárta után írásban értesíti a késlekedőket.

8.55 

Készítsünk programot, amely kollégiumi szobabeosztást készít! Feltételek, hogy a lányok és a fiúk lehetőleg külön emeletre kerüljenek, továbbá mindenki legalább egy osztálytársával legyen egy szobában!

9. FEJEZET

GÉPKÖZELI FELADATOK

MÉRÉS – VEZÉRLÉS
GÉPI LEHETŐSÉGEK PROGRAMOZÁSA

9. GÉPKÖZELI FELADATOK

Ebben a fejezetben olyan feladatok szerepelnek, amelyek valamilyen szempontból igénylik a számítógép mélyebb ismeretét. Ennek ellenére mindegyik programozási feladat. Úgy gondoltuk ugyanis, hogy ebbe a gyűjteménybe nem illenek bele műszaki jellegű feladatok. A fejezet két részből áll:

- mérés-vezérlés,
- gépi lehetőségek programozása.

Az első rész, bár programozással kapcsolatos, mégis igényelhet műszaki tevékenységet: a mérő, illetve vezérlő rendszer összeállítását.

A másodikban gépi kódú feladatokkal foglalkozunk, ebben az esetben a megoldás függ az alkalmazott processzor típusától.

Javasolt irodalom:

1. Agárdi G.: Gyakorlati Assembly
   LSI, 1994
2. Agárdi G.: Gyakorlati Assembly haladóknak
   LSI, 1995
3. Ila L.: PC műhely 1-3.
   Panem, 1997
4. Sas T.: Vezérlések párhuzamos porton keresztül
   LSI, 1995

9.1 Mérés - vezérlés

A most következő feladatok többségének megoldása akkor látványos, ha megfelelő interfésszel a szövegben szereplő perifériákat is működtetni tudjuk. Érdemes az adott interfészhez kimeneti és bemeneti szubrutint írni. A kimenetinek adjuk meg a K(1)...K(16) változókban a kimeneti bitek értékét, a bemenetitől pedig a B(1)...B(16)-ban kapjuk vissza az információt. Ha nincs interfészünk, próbáljuk a működését modellezni! A bemeneti alprogramhoz a billentyűzeten, a kimeneti alprogramhoz a képernyőn vagy a hanggenerátorban keressünk megfelelő egyszerű helyettesítő funkciót, de semmiképpen sem érdemes vaktában próbálkozni a szá-

mítógép ki- és bemeneteivel, mert könnyen okozhatunk maradandó változást az áramkörökben.

A feladatok megoldásához ésszerű Pascal és Assembly nyelvű rutinokat alkalmazni.

9.1 

A nyolc elemű Kimenet tömb minden eleme 0 vagy 1. Jelentsék ezek helyiérték szerint indexelve egy bájt bitjeit! Készítsünk programot, ami előállítja a bájt értékét!

9.2 

Adott egy bájt. Állapítsuk meg a 32-es helyiértékű bitjének értékét!

9.3 

Adott egy bájt, bitjeinek értékét válogassuk szét helyiérték szerint növekvő indexekkel a nyolc elemű Bemenet tömbbe!

9.4 

Adott egy ismeretlen értékű bájt. A legkisebb helyiértékű két bitjét állítsuk 1-re! (A többi bit maradjon változatlan!)

9.5 

Adott egy ismeretlen értékű bájt. A legkisebb helyiértékű két bitjét állítsuk 0-ra!

9.6 

Az előző két feladatot oldjuk meg úgy is, hogy nem vizsgáljuk meg a kérdéses bitek értékét!

9.7 

Az X változó értéke 2 vagy 3 lehet. Írjunk olyan eljárásokat, amelyek a két értéket éppen felcserélik!

9.8 

Az X változó értéke egész szám 0-tól 3-ig. Írjunk olyan eljárást, amelyik a következő függvényt állítja elő: f(0)=0, f(1)=1, f(2)=3 és f(3)=2!

9.9 

Készítsünk modulo 4 számláló programot! (Néggyel osztva a lehetséges maradékokat állítsa elő sorra!)

9.10 

A legegyszerűbb léptetőmotorokat (pl. egyes lemezjátszók motorja is lényegében ilyen) úgy lehet vezérelni, hogy a két tekercs polaritását felváltva változtatjuk. Minden változás egy lépéssel (pl. 9 fokkal) fordítja a tengelyt. A kimeneti

függvény Kimenet tömbjének alsó két elemével állítja be a tekercsek polaritását: ha Kimenet[1]=1, akkor pozitív feszültséget kap az 1. tekercs (kezdőpontja), ha Kimenet[1]=0, akkor negatívat. Írjunk olyan programot, amelyik a léptetéshez szükséges kimeneti értékeket előállítja, és a kimeneti függvénnyel kiküldi a motornak! [(0, 0), (0, 1), (1, 1), (1, 0), (0, 0) stb.]

9.11

Az előző feladatot oldjuk meg úgy, hogy visszafelé forogjon a motor!

9.12

Változtassuk meg az előző programokat úgy, hogy adott számú lépést tegyen meg a motor! Ügyeljünk arra, hogy a motor pillanatnyi állapotából induljon a számlálás, tehát az adott elfordulás megkezdésekor ne legyen kiszámíthatatlan irányú és méretű ugrás!

9.13

Az előző két feladatot vonjuk össze úgy, hogy előjelesen lehessen megadni a lépések számát!

9.14

A számítógép két jelfogó közbeiktatásával egy egyenáramú motort vezérel. Ha a kimeneti függvény meghívásakor Kimenet[1] értéke 1, akkor bekapcsolja, ha 0, kikapcsolja. Kimenet[2]=1 esetén előre, Kimenet[2]=0 esetén hátrafelé forog (csak, ha be van kapcsolva). Írjunk programot, amely a három lehetséges bemeneti szót (előre, hátra, ki) feldolgozza, és végrehajtatja a motorral!

9.15

A legegyszerűbb útkereszteződésben a jelzőlámpának négy állapota lehet: piros, piros-sárga, zöld, sárga. Tervezzük meg a lámpák vezérlését, és írjuk fel az egyes lámpák logikai függvényét, ha három kimeneti bit (K(1), K(2), K(3)) áll rendelkezésünkre! Kézenfekvőnek tűnik, hogy az egyik bit a sárga lámpákat vezérelje. (Például az első útvonal zöld lámpája a Zöld1=Kimenet[3] ÉS NEM Kimenet[2] szerint működhet.)

9.16

Oldjuk meg az előző feladatot két kimeneti bittel is!

9.17

Írjunk programot a közlekedési lámpák irányítására! A program először kérdezze meg a lámpa egyes állapotainak időtartamát, majd eszerint az időterv szerint állítsa be a kimeneti biteket, és hívja meg a kimeneti alprogramot!

9.18

Az előző programban lehessen a lámpát a billentyűzetről sárga villogóra és vissza is kapcsolni! Tegyük lehetővé az időtartamok változtatását a lámpák működése közben is! Ügyeljünk arra, hogy a bemenet feldolgozása ne változtassa meg az éppen mért időtartamot!

9.19

A számítógép egy fotoellenállás állapotát olvassa be. Ha világos van, Bemenet[1] értéke 0, különben 1. Írjunk programot a sötét állapotok megszámlálására (pl. munkadarab haladt el az érzékelő előtt)! Az aktuális értéket lehessen mindig leolvasni a képernyőről!

9.20

Módosítsuk az előző programot úgy, hogy (átlag) sebesség mérésére legyen alkalmas! Mérjük például egy 1 cm széles test áthaladásának idejét!

9.21

Az előző feladatban leírt test közepére vágjunk egy nyílást! A fotókapu két sebességet tud így mérni. Számítsuk ki a mért adatokból a test (átlagos) gyorsulását!

9.22

A sebességmérő-programból készítsünk fordulatszámmérőt! A mérendő tengelyre szereljünk egy hasítékkal ellátott korongot! Ennek áthaladási gyakoriságát kell mérni a fotókapuval.

9.23

Az előző két feladat összekapcsolásával mérjük meg, hogyan függ az inga lengésideje a kitéréstől! A kitérést az alsó helyzetben mérhető sebességből számíthatjuk ki a helyzeti a mozgási és a forgási energia összevetésével. (Erre a célra kiváló ingát készíthetünk az iskola forgómozgást vizsgáló készülékéből, ha a tengelyét vízszintesre állítjuk.)

9.24

Az egyenáramú motor vezérlését kapcsoljuk össze a fordulatszám-mérésnél használt ötlettel (9.22), és modellezzük a léptetőmotort! Kapcsoljuk be a motort, és addig hagyjuk forogni, míg adott számú fordulatot meg nem tesz! A korongon több bevágás is készíthető, így törtfordulatokat is tudunk számlálni.

9.25

Bővítsük a 9.19-es feladat fotókapuját úgy, hogy két egymás utáni érzékelő adja a jeleket, és a program azt állapítsa meg, hogy melyik irányba ment pl. a múzeum látogatója!

9.26

Az előző feladatot módosítsuk úgy, hogy a gép mindig a múzeumban lévő személyek számát írja ki! (Más kijárat nincs, az emberek nem takarhatják egymást, mert keskeny a kapu.)

9.27

A Bemenet tömb első két elemének állapotát megint a fény határozza meg, de ahogy nő a megvilágítás erőssége, először a Bemenet[1], később Bemenet[2] válik eggyé. Írjuk ki a képernyőre, hogy világos, sötét vagy félhomály van-e!

9.28

A bemeneti függvényben a Bemenet[1] értékét egy mikrofon határozza meg, de soha nem tudhatjuk, hogy mi Bemenet[1] alapállapota, csak azt, hogy ha tapsolunk, legalább egyszer változik. Írjunk programot, amelyik tapsra kiírja, hogy „Ne zajongj!"!

9.29

Írjunk programot, amelyik megméri, hogy mennyi ideig volt lenyomva egy billentyű, és később ugyanennyi időre bekapcsol egy hangot!

9.30

Egy lánctalpas jármű két lánctalpát egy-egy nyomógombbal tudjuk működtetni. A nyomógombok állapota a bemeneti alprogramban Bemenet[1] és Bemenet[2] állapotát is beállítja. Készítsünk tanulórobotot, amelyik a nyomógombok egymás után következő állapotait és az állapotváltozások közötti időtartamokat megjegyzi, és a sorvég-billentyű lenyomására megismétli a mozgássorozatot! A Kimenet[1] és Kimenet[2] a kimeneti függvényben ugyanúgy kapcsolja a motorokat, mint a két nyomógomb.

9.31

Irányítsuk az előző járművet a billentyűzetről az „ELŐRE", „HÁTRA", „JOBBRA" és „BALRA" (90 fokos fordulat) szavakkal!

9.32

Egy lánctalpas jármű két lánctalpát a kimeneti alprogram a Kimenet[1], illetve a Kimenet[2] értékétől függően ki- vagy bekapcsolja. A bemeneti alprogram Bemenet[3] értékét aszerint állítja 0-ra vagy 1-re, hogy balról vagy jobbról jön-e erősebb fény. Írjunk olyan programot, amelyik a járművet a fényforrás felé vezeti!

9.33

Írjuk át az előző programot a fényforrástól való menekülésre!

9.34

Egy motor függőleges tengely körül tud forgatni két fényérzékelőt. Ezek kis szöggel különböző irányba néznek. Ha a baloldali „lát" több fényt, akkor a bemeneti függvényben Bemenet[3]=1, különben Bemenet[3]=0. A kimeneti alprogram Kimenet[1]=1 esetén bekapcsolja a motort, Kimenet[2]=1 esetén jobbra, Kimenet[2]=0 esetén balra forgatja. Írjunk programot, amely „tekintetével" követi a fényforrást!

9.35

A számítógéphez egy soros analóg-digitális átalakítót kapcsoltunk. A kimeneti eljárás számára megadott Kimenet tömb nyolc bináris számjegyének megfelelő számmal arányos feszültséget állít elő, és ezt hasonlítja össze a mérendő feszültséggel. Ha a mérendő feszültség nagyobb, a bemeneti eljárásban Bemenet[1] értéke 0 lesz, különben 1. Írjunk programot, amely 0-tól növelve a feszültséget megállapítja a mérendő feszültség értékét!

9.36

Írjuk át az előző programot úgy, hogy az első mérést 0-ról kezdje, de minden továbbit az előző mérés eredményétől szükség szerint felfelé vagy lefelé lépkedve végezzen el!

9.37

Az előző programot írjuk át úgy, hogy az intervallumfelezés módszerével keresse meg a mérendő feszültséget! Ennek az eljárásnak erénye a gyorsaság és az értéktől független átalakítási idő.

9.38

Egy potenciométer (fél botkormány) állapotát úgy is beolvashatja a számítógép, hogy a potenciométer egy időzítő áramkör időtartamát állítja be, és a gép ezt az időtartamot méri.

9.2 Gépi lehetőségek programozása

Az egyes mikroprocesszorokat sajnos eléggé eltérő módon kell programozni, ezért ez az alfejezet nem lehet annyira független a géptől, mint a többi. A másik nehézség abban rejlik, hogy még az azonos processzorral dolgozó gépek között is rengeteg különbség lehet a gépi kódú programozás szempontjából.

Az itt következő feladatok első csoportja egy-egy szubrutin vagy program írását követeli meg. Ezeket tetszőleges személyi számítógépen meg lehet oldani. Nagy segítséget jelent, ha van a géphez egy ASSEMBLER program. Bizonyos funkciókat magasszintű programnyelven is megvalósíthatunk (memóriakiírás, tizes-

tizenhatos átszámítás, egy bájt töltése a memóriába stb.). Egyes feladatokhoz elkerülhetetlen a memóriafelosztás ismerete, másoknál nagyon jó hasznát vehetjük az egyes ROM rutinoknak, ezek megismerhetők a szakirodalomból.

A feladatok második része programelemzés, hibakeresés.

9.39 

Írjuk tele a képernyőkezelő megszakításrutinok segítségével a képernyőt A betűkkel!

9.40 

Húzzunk a képernyő bal alsó sarkából átlósan egy vonalat csillagokból!

9.41 

Írjunk programot, amely a táblázatrajzoló karakterek felhasználásával egy téglalapot állít elő a képernyőn! A méretet a billentyűzetről adjuk meg!

9.42 

Készítsünk olyan szubrutint, amely a számítógép képernyőjének bal- és jobboldalát megcseréli (tükrözi a képfelező egyenesre)!

9.43 

Készítsünk szubrutint, amely a számítógép képernyőjének bal vagy jobboldalát törli le! Bővítsük úgy a feladatot, hogy a szubrutin a képernyő tetszőleges téglalapját tudja törölni!

9.44 

Készítsünk olyan szubrutint, amely a számítógép képernyőjét inverzre változtatja (mintha fénykép negatív lenne)!

9.45 

Írjunk gépikódú szubrutint, amely 32 bites sorozatot léptet körbe balra bitenként!

9.46 

Írjunk programot egy bájt bitsorrendjének megfordítására!

9.47 

Készítsünk programot a képernyő függőleges oldalfelező merőlegesére való grafikus tükrözésre!

9.48 

Készítsünk szubrutint, amely a képernyőt

1. egy karakteres oszloppal balra lépteti,

2. egy karakteres oszloppal jobbra lépteti,
3. egy karakteres sorral felfelé lépteti,
4. egy karakteres sorral lefelé lépteti!

9.49 

Készítsünk szubrutint, amely a képernyőt ciklikusan (azért, mert ami az egyik szélén kimegy, az a másik szélén azonos oszlopban, ill. sorban visszajön.)

1. egy karakteres oszloppal balra lépteti,
2. egy karakteres oszloppal jobbra lépteti,
3. egy karakteres sorral felfelé lépteti,
4. egy karakteres sorral lefelé lépteti!

9.50 

Készítsünk szubrutinokat a képernyő és egy lemezegység közötti képátvitelre!

1. Képernyő kimentése háttértárra.
2. Kép beolvasása háttértárról.

9.51 

Készítsünk szubrutint, amely a képernyő tartalmát elmenti egy megadott memóriaterületre, illetve onnan vissza tudja hozni!

9.52 

Készítsünk szubrutint, amely a képernyőt

1. egy grafikus oszloppal balra lépteti,
2. egy grafikus oszloppal jobbra lépteti,
3. egy grafikus sorral felfelé lépteti,
4. egy grafikus sorral lefelé lépteti!

9.53 

Készítsünk szubrutint, amely a képernyőt ciklikusan

1. egy grafikus oszloppal balra lépteti,
2. egy grafikus oszloppal jobbra lépteti,
3. egy grafikus sorral felfelé lépteti,
4. egy grafikus sorral lefelé lépteti!

9.54 

Töröljük le a grafikus képernyő adott színnel befestett pontjait!

9.55 

Másoljunk a képernyőre egy elmentett képet úgy, hogy a két kép között AND, OR, XOR műveletet végzünk.

9.56 

Írjunk szubrutint a képernyő négy irányú grafikus léptetésére! Az irányt a billentyűzetről adhatjuk meg, és minden billentyűleütés egy ponttal léptesse a képernyőt!

9.57 

Mozgassunk egy pontot a képen úgy, hogy

A. a képernyő szélén álljon meg,
B. a képernyő széléről verődjön vissza,
C. a képernyőn lefelé gyorsuljon (hajítások),
D. egy adott pont felé gyorsuljon (bolygómozgás),
E. lefelé gyorsuljon, visszaverődésnél sebessége csökkenjen,
F. a képernyőn levő tetszőleges ábráról verődjék vissza,
G. folyamatosan lassuljon (közegellenállás)!

9.58 

Írjunk függvényábrázoló programot! Ha az értelmezési tartomány nem fér a képernyőre, lehessen a grafikont jobbra-balra tologatni! Egy billentyű lenyomására a képernyő középpontjából nagyítsa kétszeresére a grafikont, egy másikra pedig kicsinyítse a felére!

9.59 

Írjunk szubrutint két képpont egyenessel való összekötésére!

9.60 

A koordinátáikkal megadott két pontot összekötő egyenesnek keressük meg a képernyőre eső szakaszát, és csak azt rajzoljuk fel!

9.61 

Írjunk szubrutint adott sugarú és adott középpontú kör megrajzolására!

9.62 

Írjunk szubrutint adott méretű és helyzetű téglalap rajzolására!

9.63 

Írjunk szubrutint a képernyő egy zárt görbével határolt konvex részének beszínezésére! (Tegyük fel, hogy a konvex, zárt görbét a program egy másik, már létező függvényével hozzuk létre!)

9.64 

Próbáljuk módosítani az előző programot bármilyen tartomány befestésére!

9.65 

Készítsünk függvényeket mozgó képernyőalakzatok kezelésére! Az egyik az alakzat leendő helyéről egy téglalapot elment a memóriába, majd rárajzolja az alakzatot. A másik az előző téglalap tartalmát visszaállítja, és az alakzatot elmozdítja a paramétereknek megfelelően. Ügyeljünk a mozgó alakzat határára!

9.66 

A képernyő egy területének gyors váltogatásával állítsunk elő egyszerű mozgásokat! Néhány ötlet: repülő madár, araszoló hernyó, integető vagy lépegető ember.

9.67 

Többfázisú animációval mutassuk be egy húr vagy hártya állóhullámainak mozgását!

9.68 

Írjunk programot billentyűvezérelt rajzolásra! A pont elmozdulásának irányát a kurzormozgató billentyűkkel adjuk meg! Ha a SHIFT billentyűt is lenyomjuk, töröljön!

9.69 

Az előző programot írjuk át úgy, hogy a billentyűkkel mindig egy szakasz végpontját mozgassuk! Ha lenyomjuk az ENTER billentyűt, az rögzíti a pontot, és ez lesz az újabb szakasz kezdőpontja. Az egész legyen olyan, mintha egy helyenként leszögezett gumiszalaggal alakítanánk az ábrát!

9.70 

Írjunk programot billentyűvezérelt teknőcgrafikára! A teknőc előre, illetve hátra mozdulását a kurzor fel, illetve kurzor le billentyűvel adjuk meg, a 90 fokos elfordulásokat a kurzor jobbra, illetve kurzor balra billentyűvel! Elmozdulás közben a teknőc hagyjon nyomot maga után!

9.71 

Írjunk programot, amely a bal felső sarokból kezdve a lenyomott billentyűk jelét sorban a képernyőre írja addig, míg ENTER-t nem ütünk, vagy be nem telik a képernyő!

9.72 

Készítsünk programot, amely egy speciális billentyűre törli a képernyőt, egy másikra a bal felső sarokba megy, a képernyőn a kurzor helyére kiírja a lenyomott billentyű betűjét, a kurzor a kurzormozgató billentyűk hatására pedig elmozdul! A program tudja kinyomtatni a képernyő tartalmát!

9.73 

Cseréljük egy szöveg kisbetűit nagybetűre! A többi jel maradjon változatlan!

9.74 

Készítsünk videó-feliratozó programot! Tervezzünk nagyméretű karaktereket vonalakból vagy pontokból, és ezeket írja ki a számítógép a megfelelő billentyű lenyomására!

9.75 

Készítsünk fényújságot! A memória egy területén tárolt szöveget nagyméretű karakterekkel írjuk ki a képernyőre, és a szokásos módon forgassuk! (Grafikus pontonkénti léptetéssel nem villog bántóan a szöveg.)

9.76 

Készítsünk képújságot! A memória egy területén tárolt szövegeket és ábrákat ciklikusan cserélgesse a program! Az oldalakat a képernyőn készítsük el, és blokkátvitellel tegyük a memóriába, és ugyanígy vigyük vissza a képernyőre, amikor éppen kell! A lapváltást a felhasználó billentyűlenyomással kérje, ezzel esetleg befolyásolhatja azt is, hogy melyik legyen a következő lap!

9.77 

Írjunk ki egy memóriatartományt oktálisan a képernyőre!

9.78 

Írjunk ki egy memóriatartományt binárisan a képernyőre!

9.79 

Írjunk ki egy memóriatartományt hexadecimálisan a képernyőre!

9.80 

Írjunk ki egy memóriatartományt karakteres formában a képernyőre! A nem kiírható karakterek helyére tegyünk pontot!

9.81 

Írjunk programot, amely egy memóriatartományt hexadecimális és karakteres formában is kiír!

9.82 

Készítsünk programot, amely az egyik regiszterben lévő bináris egész számot hexadecimális formában a képernyőre írja!

9.83 

Készítsünk programot, amely a binárisan beírt számokat hexadecimálisan írja ki!

9.84 

Írjunk szubrutint, amely 128 biten ábrázolt előjeles egész számokat ad össze!

9.85 

Írjunk szubrutint, amely 80 biten ábrázolt előjeles egész számokat von ki!

9.86 

Számítsuk ki két hexadecimális szám összegét és különbségét!

9.87 

Írjunk programot kétbájtos számok kettővel való szorzására!

9.88 

Írjunk programot kétbájtos számok tízzel való szorzására!

9.89 

Írjunk szubrutint kétbájtos számok tízzel osztására!

9.90 

Készítsünk morzejelekre fordító programot! A morzejelek a következők:

A	.-	K	-.-	U	..-
B	-...	L	.-..	V	...-
C	-.-.	M	--	W	.--
D	-..	N	-.	X	-..-
E	.	O	---	Y	-.--
F	..-.	P	.--.	Z	--..
G	--.	Q	--.-	Á	.--.-
H	....	R	.-.	É	..-..
I	..	S	...	Ö	---.
J	.---	T	-	Ü	..--
1	.----	6	-....	Pont	.-.-.
2	..---	7	--...	Hiba	....
3	...--	8	---..	SOS	...---...
4	....-	9	----.		
5	.....	0	----		
Üzenet kezdete	-.-.			Üzenet vége	.-.-.

A jelek között szünetet, a szavak között háromszoros szünetet kell tartani. A program egysoros üzenet fordítását végezze el! A szüneteket jelöljük / (törtvonal)-lal!

9.91 

Készítsünk programot, amely a billentyűzetről morzejeleket kér, és az így beadott szöveget megfejti!

9.92 

Kísérletileg határozzuk meg a számítógépünkben még ábrázolható legkisebb és legnagyobb abszolútértékű számokat!

9.93 

Határozzuk meg a számítógépünkben az egyes változótípusokkal végzett aritmetikai műveletek időszükségletét! Figyeljünk arra, hogy más körülményt ne változtassunk!

9.94 

A, B, C és X jelentsen 16 bites egész számot! Kódoljuk a következő algoritmust! Mi lesz a változók értéke a ciklus lefutása után?

```
A:=0: B:=1: C:=1
Ciklus amíg B≤X
  A:=A+1
  C:=C+2
  B:=B+C
Ciklus vége
```

A kódolás elején döntsük el azt, hogy melyik változónak melyik regiszterpár felel meg!

9.95 

Az A és a B szám egybájtos egész. Szorzatuk a kétbájtos Z-ben keletkezik. Kódoljuk az algoritmust!

```
X:=A: Y:=B: Z:=0
Ciklus amíg Y≠0
  Ha Y páros
    akkor X:=2*X: Y:=Y/2
    különben Z:=Z+X: Y:=Y-1
  Elágazás vége
Ciklus vége
```

Ha A és B kétbájtos egész is lehet, akkor egészítsük ki a megoldást túlcsordulás figyelésével!

9.96 

Gépi kódú szubrutinnal kezelünk egy egeret. Az IN AL,34 utasítás olvassa be az egér pillanatnyi állapotát. Az AL regiszter felső négy bitje mindig 1, a következő kettő az előre mozgás miatt a 00, 01, 11, 10, 00 sorozatot adja, hátrafelé éppen fordítva. Az utolsó kettő a balra, illetve jobbra mozgásnál ugyan-

így viselkedik. Írjuk meg a hiányzó programrészletet! Mi a szerepe a BL, BH, DL, DH regiszternek?

```
        MOV  SI,10   ;KÉPKOORDINÁTÁK AHOVA AZ EGÉR
        MOV  DI,10   ;NYOMÁT RAJZOLNI KELL
        MOV  DL,0
        MOV  DH,0
    CIKLUS:
        IN   AL,34   ;AZ EGÉR ÁLLAPOTA
        PUSH AX
        AND  AL,3
        MOV  BL,AL
        POP  AX
        AND  AL,12
        RCR  AL,1
        RCR  AL,1
        MOV  BH,AL
        ;VIZSGÁLJUK MEG, HOGY AZ EGÉR ELMOZDULT-E,
        ;ÉS HA IGEN, MÓDOSÍTSUK AZ SI éS DI ÉRTÉKÉT!
        ;TEGYÜK FEL, HOGY A RAJZOLÁST RAJZOL NEVŰ
        ;SZUBRUTINBAN MÁR MEGÍRTÁK!
        MOV  DL,BL
        MOV  DH,BH
        CALL RAJZOL
        JMP  CIKLUS
```

9.97

Elemezzük a következő 8086-os programot!

A. Mit csinál a program?

B. Mikor jut el a vezérlés a VE címkére? Mi lesz ekkor az AL és a BL regiszterben, illetve a CARRY jelzőbitben?

C. Mi történik, ha az OR 80H helyére AND 7FH-t írunk? És ha nem írunk a helyére semmit?

D. Mi történik, ha a JNC HU helyére JC HU-t írunk?

```
PR:
   MOV  BL,01H
   OR AL,80H
HU:
   RCL  AL,1
   XOR  AL,BL
KI:
   RCL  BL,1
   JNC  HU
VE:
   END
```

9.98

Mit csinál az alábbi 8086-os program? Mi lesz a regiszterek és a jelzőbitek értéke, ha P1-nél illetve P2-nél lépünk be?

A DB direktíva az assemblernek szól, azt jelenti, hogy fordítás közben a program adott helyén a DB után következő számokat vagy karakterek kódjait kell a memóriába lerakni, és utána folytatni a fordítást.

```
P1:
   MOV  SI,OFFSET PUFF1
   CALL MITESZ
VEGE1:
   RET
PUFF1:
   DB 'O', '1', '4', '0', '7', '0'; FIGYELEM! '0'<>'O'
P2:
   MOV  SI,OFFSET PUFF2
   CALL MITESZ
VEGE2:
   RET
PUFF2:
   DB 'D', '1', '4', '0', '7', '0'
MITESZ:
   MOV  CH,0
   MOV  DI,0
   MOV  AL,0[SI]
   CMP  AL,'O'
   JNZ  TALAND
   MOV  CL,4-1
   JMP  ATALAK
```

```
TALAND:
   CMP  AL,'D'
   JNZ  LEPJKI
   MOV  CL,5-1
ATALAK:
   MOV  AH,CL
CIKLUS:
   INC  SI
   MOV  AL,0[SI]
   SUB  AL,'0'
   JS   LEPJKI
   MOV  CL,AH
   ADD  DI,DI
   PUSH DI
   POP  BX
BELCIK:
   ADD  DI,BX
   LOOP BELCIK
   MOV  BL,AL
   MOV  BH,0
   ADD  DI,BX
   JMP  CIKLUS
LEPJKI:
   RET
   END
```

9.99

Mit csinál az alábbi szubrutin? Bemenete DL és CL.

```
CIK:
   MOV  BL,DL
   CMP  BL,CL
   JZ   NEM
   MOV  BL,DL
   SUB  BL,CL
   MOV  DL,BL
   JMP  CIK
NULLD:
   PUSH CX
   AND  CL,1
   POP  CX
   JZ   NULLE
   SHR  DL,1
   JMP  CIK
NULLE:
   SHR  DL,1
   SHR  CL,1
   JMP  CIK
NEM:
   STC
   RET
IGEN:
   OR   BL,BL
   RET
   END
```

9.100

Milyen hibák vannak a következő programban?

```
   MOV  SI,68000
   MOV  AL,0[SI]
CIK:
   DEC  AL
   MOV  DX,0-10
   ADD  SI,DX
   MOV  AL,5
   OR   AL,AL
   JNZ  CIK
   CALL SZUB
   RET
```

```
SZUB:
   MOV  0[SI],AL
   DEC  SI
   JP   CIK
   END
```

9.101

Mi a következő program hibája? (A CX/DX hányadost SI-be kellene tennie, a maradékot pedig BX-be.)

```
   MOV  SI,0FFFFH
   MOV  BL,CH
   MOV  BH,CL
CIK:
   INC  SI
   SBB  BX,DX
   JMP  CIK
   END
```

9.102

Mit csinál a következő 8086-os szubrutin? (SI a bemenet)

```
KODSZEG SEGMENT
ASSUME  CS:KODSZEG
   PUSH AX
   PUSH CX
   PUSH DX
   MOV  DX,0F74DH
   MOV  BH,0
   MOV  AH,0AH              ;INTERRUPT ELŐKÉSZîTÉSE
   MOV  CX,16
CIK:
   MOV  AL,'0'
   SHR  DX,1
   ADC  AL,0
   INT  10H                 ;AL KIÍRÁSA CX-SZER
   LOOP CIK
   POP  DX
   POP  CX
   POP  AX
   MOV  AH,4CH
   INT  21H
KODSZEG ENDS
   END
```

9.103

Mit csinál a következő 8086-os szubrutin? Mi lesz DX értéke a VEGE illetve a HIBA címke után?

```
      MOV  DX,0
   CIK:
      MOV  AH,0
      INT  16H          ;BILLENTYŰ BEOLVASÁSA
                        ;AL-BE
      CMP  AL,0DH       ;SOR VÉGE KARAKTER
      JZ   VEGE
      CMP  AL,'0'
      JS   HIBA
      CMP  AL,'9'
      JNS  HIBA
      SHL  DX,1
      MOV  BX,DX
      MOV  CX,2
   CIK1:
      OR   AL,AL
      ROL  DX,1
      LOOP CIK1
      ADD  DX,BX
      MOV  BH,0
      SBB  AL,'0'
      MOV  BL,AL
      ADD  DX,BX
      JMP  CIK
   HIBA:
                        ;HIBAÜZENET KIÍRÁSA
      RET
   VEGE:
      RET
      END
```

9.104

Mi lesz az AX és SI regiszterek értéke?

```
CIM  PROC    FAR
     MOV  SI,OFFSET CIM
     MOV  AX,0C3H
     MOV  CS:[SI],AX
     CALL CIM
     DEC  SP
     DEC  SP
     DEC  SP
     DEC  SP
     POP  AX
     POP  SI
```

Megjegyzés: A 0C3H kód a 8086 mikroprocesszor RET utasításának kódja.

10. FEJEZET

PROGRAMELEMZÉS

MIT CSINÁL A PROGRAM?
TESZTELÉS
HIBAKERESÉS
HATÉKONYABBRA ÍRÁS

10. PROGRAMELEMZÉS

A programozás tanulása során az önálló feladatmegoldással egyenrangú módszer lehet mások által írt, kész programok elemzése, vizsgálata, átalakítása valamilyen szempontból. Mindez megtehető algoritmusokkal, illetve valamilyen nyelven kódolt programokkal.

Ez a programelemzés hasznos saját programok készítése közben is, ugyanis így tudjuk programjaink minőségét garantálni. Éppen ezért e fejezetben olyan kérdésekre próbálunk választ adni, hogy:

- mit csinál a program (algoritmus)?
- milyen adatokkal kell kipróbálni a programot?
- mi a hiba a programban?
- hogyan lehet a programot (algoritmust) hatékonyabbra írni, azaz időben gyorsabbra, kevesebb memóriát használóra (rövidebb programszöveg, kevesebb változó), valamint egyszerűbbre?

Egy program helyes működését teszteléssel ellenőrizhetjük, amely nem garantálja a program hibamentességét, csupán valószínűsíti azt. Ha a tesztelés során azt látjuk, hogy a program hibásan működik, akkor meg kell keresnünk a hibát. Ez a hibakeresés. Ha megtaláltuk, majd kijavítottuk, a folyamat újra kezdődik, hiszen meg kell győződnünk róla, hogy most már helyesen működik programunk.

Szükségünk lehet más által írt eljárásokra, programokra is. Ebben az esetben el kell tudnunk dönteni, hogy mit csinál a program, esetleg tesztelnünk is kell.

Néhány feladat a helyességbizonyítás témakörébe tartozik. Ilyen bizonyításokat a fejezet első feladatainál (10.2, 10.3, 10.4, ...) is végezhetünk. A 10.4. feladatban például a következő programrészletről kellett eldönteni, hogy mit csinál:

```
Eljárás:
   Be: X
   A:=0: B:=1: C:=1
   Ciklus amíg B ≤ X
      A:=A+1: C:=C+2: B:=B+C
   Ciklus vége
   Ki: A
Eljárás vége.
```

A CIKLUS-ra igaz a következő invariáns állítás:

```
C=2*A+1 és B=(A+1)*(A+1) és A*A ≤ X.
```

Ha a ciklusmag valamelyik végrehajtásakor valamely egész k>0 és k*k ≤ X-re A értéke k-1, C=2*(k-1)+1 és B=k*k, akkor a ciklusmag következő végrehajtásakor A=k, C=2*k+1 és B=k*k+2*k+1=(k+1)*(k+1).

A ciklusmag első végrehajtásakor az invariáns állítás k=1 érték mellett igaz. Ez megalapozza az indukció elve alapján, hogy A értéke az az utolsó k-1, amelyre már k*k>X, vagyis az A értéke INT(SQR(X)) lesz.

A ciklus befejeződésének bizonyítására más módszert kell keresni! Készítsünk egy ún. variáns függvényt, amelyre az invariáns állításból következik, hogy nemnegatív egész értékű, és a ciklusmag egyszeri végrehajtása alatt értéke legalább eggyel csökken! Mivel ennek kezdőértéke egy véges szám, ez a tulajdonsága garantálja, hogy a ciklusmagot csak véges sokszor hajtjuk végre.

Ez a variáns függvény lehet például a következő:

```
X-A*A.
```

Az invariáns tulajdonság utolsó tagjából következik, hogy ez mindig nemnegatív értékű, a ciklusmagban pedig van egy A=A+1 utasítás, ami ennek az értékét legalább eggyel csökkenti. Így tehát beláttuk, hogy ez nem lehet végtelen ciklus.

JAVASOLT IRODALOM:

1. Várkonyi Zs.: Bevezetés a modern programtesztelésbe
Műszaki Könyvkiadó, 1979
2. Varga L.: Programok analízise és szintézise
Akadémiai Kiadó, 1981
3. Szlávi P. - Zsakó L.: Módszeres programozás
Műszaki Könyvkiadó, 1986
4. Számítástechnika középfokon
OMIKK, 1987

10.1 Mit csinál a program?

Más által írt programok használatakor, vagy hibásan működő programrészek hibáinak megfejtéséhez szükségünk lehet annak felderítésére, hogy mit csinál az adott programrész. Hosszabb program esetén ez meglehetősen bonyolult feladat, ezért célszerű részekre bontani. Néhány soros programrészletekről könnyebben eldönthetjük, hogy milyen bemenő adatokra, milyen utasításokat végez el a gép, és milyen kimenő értékeket kapunk.

Ugyanaz a programrész természetesen más ágon futhat le, más eredményt adhat attól függően, hogy melyek a bemenő adatok. Ezért a program működését nem csak kitalálni kell, hanem lehetőleg szabatos matematikai gondolkodás alapján belátni.

10.1

Mit csinál a következő program?

```
Eljárás:
  Be: A, B, C
  A:=B: C:=A
  Ha A=B és B=C akkor Ki: "EGYENLŐK"
Eljárás vége.
```

10.2

Mit csinál a következő program? (X, Y egész számok!)

```
Eljárás:
  Be: X, Y
  Z:=0
  Ciklus amíg Y≠0
    Ha Y páros akkor X:=2*X: Y:=Y/2
               különben Z:=Z+X: Y:=Y-1
  Ciklus vége
  Ki: Z
Eljárás vége.
```

10.3

Mit csinál a következő program? (X, Y pozitív egészek!)

```
Eljárás:
  Be: X, Y
  Z:=1
  Ciklus amíg Y≠0
    Ha Y páros akkor Y:=Y-1: Z:=Z*X
               különben X:=X*X: Y:=Y/2
  Ciklus vége
  Ki: Z
Eljárás vége.
```

10.4 

Mit csinál a következő program? (X>0 !)

```
Eljárás:
  Be: X
  A:=0: B:=1: C:=1
  Ciklus amíg B≤X
    A:=A+1: C:=C+2: B:=B+C
  Ciklus vége
  Ki: A
Eljárás vége.
```

10.5 

Mit csinál a következő program? (A>0, egész!)

```
Eljárás:
  Be: A
  C:=0
  Ciklus I=5-től 1-ig -1-esével
    Ciklus amíg A≥0
      C:=10*C-1: K:=10^I
    Ciklus vége
    A:=A+K
  Ciklus vége
  Ki: C
Eljárás vége.
```

10.6 

Mit csinál a következő program? (A, B pozitív egészek!)

```
Eljárás:
  Be: A, B
  Ciklus amíg A≠B
    Ha A<B akkor B:=B-A különben A:=A-B
  Ciklus vége
  Ki: A
Eljárás vége.
```

10.7 

Mit csinál a következő program?

```
Eljárás:
  Be: X, Y
  X:=X+Y: Y:=X-Y: X:=X-Y
  Ki: X, Y
Eljárás vége.
```

10.8 

Mit csinál a következő program?

```
Eljárás:
  Be: A, B, C
  Ha A>B akkor X:=A: A:=B: B:=X
  Ha A<C akkor X:=A: A:=C: C:=X
  Ha A<B akkor X:=A: A:=B: B:=X
  Ki: A, B, C
Eljárás vége.
```

10.9 

Mit csinál a következő Pascal programrészlet (a változók megfelelő módon deklarálva vannak)?

```
SZ:=0; HELY:=' ';
REPEAT
  SZ:=SZ+1; BETU:=READKEY;
UNTIL BETU <> HELY;
WHILE BETU <> '.' DO
BEGIN
  WRITE(BETU);
  SZ:=0;
  REPEAT
    SZ:=SZ+1; BETU:=READKEY;
  UNTIL BETU <> HELY;
  IF SZ>1 THEN WRITE(',');
END;
```

10.10 

Mit csinál a következő Pascal nyelvű programrészlet?

```
READLN(A, B);
IF A<B THEN X:=A ; A:=B ; B:=X ; Y:=A
ELSE; Y:=B; WRITE(Y);
```

10.11 

Mit ír ki a következő Pascal nyelvű programrészlet?

```
VAR I, J, K: INTEGER;
PROCEDURE ALFA(VAR J: INTEGER);
  VAR I: REAL;
BEGIN
  WRITELN(J, K);
  I:=1; IF J>1 THEN ALFA(ROUND(I));
END;
```

```
BEGIN
  I:=1; J:=5; K:=1; ALFA(K);
  ...
END;
```

10.12

Mit csinál a következő eljárás?

```
Program:
  Szóolvas(szó, elválasztó)
  Ki: szó
  Ciklus amíg elválasztó=","
    Szóolvas(szó, elválasztó)
    Ki: szó
  Ciklus vége
Program vége.
Szóolvas(szó, elválasztó)
  szó:= üres
  Betűolvas(betű)
  Ciklus amíg betű ≠ "," és betű ≠ "."
    Szó végére betű
    Betűolvas(betű)
  Ciklus vége
  Elválasztó:= betű
Eljárás vége.
Betűolvas(betű)
  Be: betű
  Ciklus amíg betű=" "
    Be: betű
  Ciklus vége
Eljárás vége.
```

10.13

Állapítsuk meg, hogy mit csinál a következő program!

```
Eljárás:
  OSZT:=100
  HELY:=1
  Y:=2: Z:=(2*Y) mod OSZT
  SOR(HELY):=Z
  Ciklus
    HELY:=HELY+1
    Y:=Z: Z:=(2*Y) mod OSZT
    SOR(HELY):=Z
```

```
      E:=1
      Ciklus amíg E ≤ HELY-1 és SOR(E) ≠ SOR(HELY)
        E:=E+1
      Ciklus vége
    amíg SOR(E)=SOR(HELY)
    Ciklus vége
    HOSSZ:=HELY-E
    Ki: HOSSZ
  Eljárás vége.
```

10.14

A lista alapján állapítsuk meg, hogy mit csinál a következő program! (Mit határoz meg, és mit ír ki az egyes kiíratások alkalmával?)

```
  Eljárás:
    I:=1
    A:=2
    B(I):=(2*A) mod 1000
    Ciklus
      I:=I+1
      B(I):=(2*A) mod 1000
      K:=1
      Ciklus amíg K ≤ I-1 és B(K) ≠ B(I)
        K:=K+1
      Ciklus vége
    amíg B(K)=B(I)
    Ciklus vége
    Ki: "ELSŐ KIÍRATÁS"
    Ki: I-K
    Ki: "MÁSODIK KIÍRATÁS"
    Ki: K
    Ki: "HARMADIK KIÍRATÁS"
    Ciklus J=K-tól I-1-ig
      Ki: B(J)
    Ciklus vége
  Eljárás vége.
```

10.15

Futtatás nélkül állapítsuk meg, hogy miben különbözik a következő két program futási képe!

```
program sz10_15_1;
  uses crt;
  var kar:string;
      k,l,n,i: integer;
begin
  clrscr; kar:='';
  repeat
    readln(n);
  until (n>=2) and (n<=8) and (n=int(n));
  for i:=1 to 2*n do kar:=kar+'*';
    for k:=1 to n do writeln(kar:4*k+n);
end.
```

```
program sz10_15_2;
  uses crt;
  var kar:string;
      k,l,n,i: integer;
begin
  clrscr; kar:='';
  repeat
    readln(n);
  until (n>=2) and (n<=8) and (n=int(n));
  for i:=1 to 2*n do kar:=kar+'*';
    for k:=1 to n do writeln(kar:2*k+n);
end.
```

10.16

Futtatás nélkül állapítsuk meg, hogy miben különbözik az alábbi két program futási képe!

```
program sz10_16_1;
  uses crt;
  var kar:string;
      k,l,n,i: integer;
begin
  clrscr; kar:='';
  repeat
    readln(n);
  until (n>=2) and (n<=8) and (n=int(n));
  for i:=1 to 2*n do kar:=kar+'*';
    for k:=1 to n do writeln(kar:2*n-k+1);
end.
```

```
program sz10_16_2;
  uses crt;
  var kar:string;
      k,l,n,i: integer;
begin
  clrscr; kar:='';
  repeat
    readln(n);
  until (n>=2) and (n<=8) and (n=int(n));
  for i:=1 to 2*n do kar:=kar+'*';
    for k:=n downto 1 do writeln(kar:k+n);
end.
```

10.2 Tesztelés

A program megírása után meg kell vizsgálnunk, hogy azt teszi-e, amit elvárunk tőle. Ezt nevezzük tesztelésnek. A tesztelésnek két logikailag eltérő módszere van:

1. Statikus tesztelés, amikor a program szövegét vizsgáljuk. Eközben kódellenőrzést, formai és tartalmi ellenőrzést végzünk.

2. Dinamikus tesztelés: a program végrehajtását vizsgáljuk. Ezt a módszert még két további csoportra szoktuk osztani aszerint, hogy a program adatait vizsgáljuk-e, s eszerint próbálunk tesztadatokat gyártani (fekete doboz módszer) vagy a szerkezete, a logikája alapján (fehér doboz módszer).

Egy program ellenőrzését célszerű az első módszerrel kezdeni, mert így a hibák jelentős része kiszűrhető.

10.17

Írjunk olyan tesztadatokat, amellyel a következő program összes utasítása legalább egyszer végrehajtható! Olyan tesztadatokat is készítsünk, amellyel minden feltétel igaz és hamis értékű is lehet!

```
Eljárás:
  Be: A, B
  Ciklus amíg A≠B
    Ha A<B akkor B:=B-A: Ki: "első"
         különben A:=A-B: Ki: "második"
  Ciklus vége
  Ki: A
  Ki: "harmadik"
Eljárás vége.
```

10.18

Írjunk olyan tesztadatokat, amellyel a következő program összes utasítása legalább egyszer végrehajtható! Olyan tesztadatokat is készítsünk, amellyel minden feltétel igaz és hamis értékű is lehet!

```
Eljárás:
  Be: X, Y
  Z:=1
  Ciklus amíg Y ≠ 0
    Ha Y páros akkor Y:=Y-1: Z=Z*X: Ki: "/1"
            különben X=X*X: Y:=Y/2: Ki: "/2"
  Ciklus vége
  Ki: Z
  Ki: "/3"
Eljárás vége.
```

10.19

Írjunk olyan tesztadatokat, amellyel a következő program összes utasítása legalább egyszer végrehajtható! Olyan tesztadatokat is készítsünk, amellyel minden feltétel igaz és hamis értékű is lehet!

```
Eljárás:
  Ciklus amíg N<1 vagy N>100
    Ki: "ELSŐ"
  Ciklus vége
  Ciklus I=1-től N-ig
    Be: SOROZAT(I)
    Ki: "MÁSODIK"
  Ciklus vége
  M:=SOROZAT(1)
  Ciklus I=1-től N-ig
    Ha M<SOROZAT(I) akkor M=SOROZAT(I): Ki:
"HARMADIK"
  Ciklus vége
  Ha M<100 akkor Ki: M: Ki: "VÉGE"
Eljárás vége.
```

10.20

Írjunk olyan tesztadatokat, amellyel a következő program összes utasítása legalább egyszer végrehajtható! Olyan tesztadatokat is készítsünk, amellyel minden feltétel igaz és hamis értékű is lehet!

```
Eljárás:
  Be: N,A
  E:=1: U:=N
  Ciklus amíg A(E)≠0
    E:=E+1
  Ciklus vége
  Ciklus amíg A(U)≠0
    U:=U-1
  Ciklus vége
  Ciklus amíg E≤U
    Ciklus amíg E≤U és A(E)=0
      E:=E+1
    Ciklus vége
    Ha E≤U akkor Ki: E
    Ciklus amíg E≤U és A(U)≠0
      U:=U-1
    Ciklus vége
    Ha E≤U akkor Ki: U
  Ciklus vége
Eljárás vége.
```

10.21 

Írjunk tesztadatokat mindegyik fehér doboz módszerrel a logaritmikus keresés algoritmushoz!

10.22 

Írjunk tesztadatokat mindegyik fehér doboz módszerrel a szétválogatás helyben algoritmushoz!

10.23 

Készítsünk tesztadatokat a beillesztéses rendezés feladathoz úgy, hogy:

A. Minden utasítást legalább egyszer végrehajtsunk!

B. Minden feltétel lehessen igaz és hamis értékű is!

10.24 

Készítsünk tesztadatokat az utasításlefedés, illetve a részfeltétel-lefedés elve segítségével a visszalépéses keresés (backtrack) tételhez!

10.25 

Írjunk állításokat az összefuttatás tétel algoritmusába, valamint variáns függvényt a ciklus befejeződésének igazolásához!

10.26

Írjunk állításokat a visszalépéses keresés tétel (backtrack) algoritmusába, valamint variáns függvényt a külső ciklus befejeződésének igazolásához!

10.27

A következő algoritmus két szám maradékos osztását végzi el. Írjunk a ciklushoz invariáns állítást, variáns függvényt, és adjuk meg, hogy milyen tételeket kell bizonyítani a helyesség belátásához!

```
Eljárás:
  A:=0: B:=X
  Ciklus amíg B ≥ Y
    A:=A+1: B:=B-Y
  Ciklus vége
  H:=A: M:=B
Eljárás vége.
```

Előfeltétel: $X \geq 0$ és $Y \geq 0$

Utófeltétel: $X=H*Y+M$ és $0 \leq M < Y$

10.28

Egy számsorozatot rendezünk orgonasíp elrendezésbe. (A legnagyobb elem lesz középen, a következő tőle balra, a következő tőle jobbra, ...) Keressünk hibát a következő algoritmusban!

```
Eljárás:
  X:=N/2
  Ciklus I=1-től N-ig
    L:=1
    Ciklus J=I+1-től N-ig
      Ha A(L)<A(J) akkor L:=J
    Ciklus vége
    B(X):=A(L)
    Ha I páratlan akkor X:=X-I különben X:=X+I
  Ciklus vége
Eljárás vége.
```

10.29

Adott egy számsorozat, válogassuk ki a különböző elemeit! Keressünk hibát a következő algoritmusban!

```
Eljárás:
  K:=1
  Ciklus I=1-től N-1-ig
    J:=I+1
    Ciklus amíg J ≤ N és A(I) ≠ A(J)
      J:=J+1
    Ciklus vége
    Ha J>N akkor B(K):=A(I): K:=K+1
  Ciklus vége
  Ki: K, B()
Eljárás vége.
```

10.30 

Adott egy számsorozat, válogassuk ki a különböző elemeit! Keressünk hibát a következő algoritmusban!

```
Eljárás:
  K:=0
  Ciklus I=1-től N-ig
    J:=I+1
    Ciklus amíg J ≤ N és A(I) ≠ A(J)
      J:=J+1
    Ciklus vége
    Ha A(I) ≠ A(J) akkor K:=K+1: C(K):=A(I)
  Ciklus vége
Eljárás vége.
```

10.31 

Adott egy számsorozat, rendezzük át úgy, hogy a negatív elemei előzzék meg a pozitívakat! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  Ciklus I=1-től N-ig
    Ha A(I)>0 akkor P:=A(I)
                    Ciklus J=I+1-től N-ig
                      A(J+1):=A(J)
                    Ciklus vége
                    I:=I-1: A(N):=P
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.32 

Adott egy számsorozat, rendezzük át úgy, hogy a negatív elemei előzzék meg a pozitívakat! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  Ciklus I=1-től N-ig
    Ha A(I)<0 akkor B(I):=A(I)
              különben B(N+1-I):=A(I)
  Ciklus vége
Eljárás vége.
```

10.3 Hibakeresés

Hibakeresésről akkor beszélünk, ha már tudjuk, hogy van hiba a programban, csak még nem tudjuk, hogy milyen jellegű és hol található. Az első feladatokban algoritmusnyelven írt programok szerepelnek. Ezekről is tudjuk, hogy hibásak, de nem futtattuk őket, tesztadatok tehát nem állhatnak rendelkezésünkre. Ezen feladatokat próbáljuk először gondolkodva végigkövetni, bennük a hibákat megtalálni! Ha így nem sikerült, akkor kódoljuk, majd futtassuk le őket és a tesztadatok alapján keressük a hibát! Módszerek:

- Az indukciós módszer szerint rendezzük a rendelkezésünkre álló tesztadatokat.
- A dedukciós módszer lényege: a hiba lehetséges okai körének szűkítése.
- Visszalépéses módszer: a hiba előfordulásának helyétől visszafelé követi a program végrehajtását mindaddig, amíg az adatok hibásak.

Hibakeresési eszközök:

- Kiírás
- Töréspontok elhelyezése
- Nyomkövetés
- Változófigyelés
- Lépésenkénti végrehajtás
- Memóriakiírás
- Állapotellenőrzés
- Tartalmi ellenőrzés

Programozási típushibák:

- Gépelési, névválasztási hibák
- Általános vezérlési hibák
- Elágazás-szervezési hibák
- Ciklusszervezési hibák
- Bemeneti adatok hibái
- Változókkal kapcsolatos hibák
- Aritmetikai kifejezések hibái
- Tömbökkel kapcsolatos hibák
- Eljárások használatának, paraméterezésének hibái

10.33

Adott egy számsorozat, rendezzük át úgy, hogy negatív elemei előzzék meg a pozitívakat! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  L:=1: K:=1
  Ciklus I=1-től N-ig
    Ha A(I)< 0 akkor B(L):=A(I): L:=L+1
            különben C(K):=A(I): K:=K+1
  Ciklus vége
  Ciklus I=1-től L-ig
    A(I):=B(I)
  Ciklus vége
  Ciklus J=L+1-től K-ig
    A(J):=C(J)
  Ciklus vége
Eljárás vége.
```

10.34

Adott egy számsorozat, határozzuk meg szélső értékeinek számát! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  MAX:=A(1): MAXDB:=1: MIN:=A(1): MINDB:=1
  Ciklus I=2-től N-ig
    Ha A(I)>MAX akkor MAX:=A(I): MAXDB:=1
    Ha A(I)=MAX akkor MAXDB:=MAXDB+1
    Ha A(I)=MIN akkor MINDB:=MINDB+1
    Ha A(I)<MIN akkor MIN:=A(I): MINDB:=1
  Ciklus vége
  Ki: MAXDB+MINDB
Eljárás vége.
```

10.35

Adott egy természetes szám, határozzuk meg a prímosztóit! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  Ciklus I=2-től négyzetgyök(N)-ig
    Ha I osztója N-nek
      akkor J:=2
            Ciklus amíg J*J≤I és J nem osztó I-nek
              J:=J+1: Ha J*J>I akkor Ki: I
            Ciklus vége
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.36

A program számok mértani közepét számolja ki, az utolsó után 10 000-et kell beadni! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  S:=1: I:=0: Be: N
  Ciklus amíg N ≠ 10000
    S:=S*N: I:=I+1
  Ciklus vége
  Ha S<0 és nem Egész(N/2) akkor Ki: "Nem valós"
     különben A:=SGN(S)*ABS(S)**(1/I) : Ki: A
Eljárás vége.
```

10.37

Milyen hibákat találunk a következő programrészletben, amely egy mátrix sormaximumainak minimumát számolja?

```
Eljárás:
  R:=A(1,1)
  Ciklus I=1-től N-ig
    S:=A(I,1)
    Ciklus J=1-től K-ig
      Ha S<A(I,J) akkor S:=A(I,J)
    Ciklus vége
    Ha R>S akkor R:=S
  Ciklus vége
  Ki: R
Eljárás vége.
```

10.38

Milyen hibákat találunk a következő programban, amely egy vektor összes maximális értékű elemének az indexét írná ki, ha jól működne?

```
Eljárás:
  Be: N
  Ciklus I=1-től N-ig
    Be: A(I): L(I):=I
  Ciklus vége
  Ciklus I=1-től N-ig
    Ciklus J=I+1-től N-ig
      Ha A(I)<A(J) akkor X:=A(I): A(J):=A(I)
                         A(I):=X
                         X:=L(I): L(I):=L(J)
                         L(J):=X
      Elágazás vége
    Ciklus vége
  Ciklus vége
  Ki: L(1): Q:=2
  Ciklus amíg L(Q)=L(1)
    Ki: L(Q): Q:=Q+1
  Ciklus vége
Eljárás vége.
```

10.39

Milyen hibákat találunk a következő programrészletben, amely egy vektor maximális értékű elemeit gyűjti ki? (Dimenzionálási hibák nincsenek!)

```
Eljárás:
  S:=0: L:=S
  Ciklus J=1-től N-ig
    Ha A(J)>L akkor S:=S+1: L:=A(J): K(S):=J
    Ha A(J)=L akkor S:=S+1: K(S):=J
  Ciklus vége
Eljárás vége.
```

10.40

Keressünk hibákat a következő algoritmusban, amely egy vektor elemei közül a második legnagyobbat választja ki!

```
Eljárás:
  Be: N [N>1 és egész]
  Be: A()
  IM:=1: M:=A(IM)
  Ciklus I=2-től N-ig
    Ha A(I)>A(M) akkor IM:=A(I): M:=I
  Ciklus vége
```

```
  UVK:=1: KUL:=A(IM)-A(1)
  Ciklus I=2-től N-ig
    SEG:=M-A(I)
    Ha SEG<KUL és SEG ≠ >0 akkor UVK:=I: KUL:=SEG
  Ciklus vége
  Ha KUL>0 akkor Ki: A(UVK)
        különben Ki: "Mind egyenlok"
Eljárás vége.
```

10.41

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. (összesen N mérés volt) A mérést szárazföld fölött kezdtük és ott is fejeztük be. Ott van tenger, ahol a mérés értéke 0. Határozzuk meg a mért szakaszon a tenger szélességét! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
  E:=1
  Ciklus amíg A(E)>0
    E:=E+1
  Ciklus vége
  U:=N
  Ciklus amíg A(U)>0
    U:=U-1
  Ciklus vége
  Ki: (U-E)*5
Eljárás vége.
```

10.42

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. (összesen N mérés volt) A mérést szárazföld fölött kezdtük és ott is fejeztük be. Ott van tenger, ahol a mérés értéke 0. Határozzuk meg a mért szakaszon azon szigeteket, amelyeken nincs 100 méternél magasabb hely! Keressünk hibákat a következő algoritmusban!

```
Eljárás:
[E=Bal part helye, U=Jobb part helye, A(E)=0, A(U)=0
]
  Ciklus I=E-től U-ig
    Ha A(I-1)=0 és A(I)>0 akkor X:=I
    Ha A(I-1)>0 és A(I)=0
       akkor Ha H<100 akkor Ki: X, I
    Ha A(I)>H akkor A(I):=H
  Ciklus vége
Eljárás vége.
```

10.43

Repülőgépen utaztunk Európából Amerikába, s az út során X kilométerenként mértük a felszín tengerszint feletti magasságát. Feltételezésünk szerint a mérés tenger felett 0, sziget felett >0 értékű. Keressünk hibákat abban az algoritmusban, amely megadja azt a szigetet, amelytől a legtávolabb van szomszéd sziget! (A közelebbi szomszéd sziget legtávolabb van. Egy ilyet kell megadni!)

```
Szigetek eljárás:
  I:=1: DB:=0
  Ciklus amíg I≤N
    Ciklus amíg I≤N és (A(I)>0 vagy A(I+1)=0)
      I:=I+1
    Ciklus vége
    Ha I≤N akkor DB:=DB+1: SZKEZD(DB):=I+1
    Ciklus amíg A(I)=0 vagy A(I+1)>0
      I:=I+1
    Ciklus vége
    DB:=DB+1: SZVÉG(DB):=I
  Ciklus vége
  TÁV:=0
  Ciklus I=2-től DB-ig
    Ha SZVÉG(I-1)-SZKEZD(I)>TÁV és
       SZKEZD(I+1)-SZVÉG(I)>TÁV
         akkor TÁV:=SZVÉG(I-1)-SZKEZD(TÁV): H:=I
  Ciklus vége
  Ki: H, SZKEZD(H), SZVÉG(H)
Eljárás vége.
```

10.44

Repülőgépen utaztunk Európából Amerikába, s az út során X kilométerenként mértük a felszín tengerszint feletti magasságát. Feltételezésünk szerint a mérés tenger felett 0, sziget felett >0 értékű. A méréssorozatot tenger felett kezdtük és fejeztük be. Keressünk hibákat abban az algoritmusban, amely megadja azokat a szigeteket, ahol a legmagasabb csúcs található! (Csúcs minden olyan pont, amely magasabb a két szomszédjánál. Az adatok az A(N) vektorban találhatók.)

```
Szigetek eljárás:
  MAG :=-20000
  Ciklus I=2-től N-ig
    Ha A(I)>A(I-1) vagy A(I)>A(I+1)
       akkor Ha A(I)>MAG akkor MAG:=A(I)
  Ciklus vége
```

```
  DB:=0
  Ciklus I=2-től N-ig
    Ha A(I)=MAG akkor DB:=DB+1: X(DB):=I
  Ciklus vége
  Ciklus I=0-tól DB-ig
    E:=A(I)
    Ciklus amíg A(E)>0
      E:=E+1
    Ciklus vége
    V:=A(I)
    Ciklus amíg A(V)>0
      V:=V-1
    Ciklus vége
    Ki: "Sziget eleje:", E, "Sziget vége:", V
  Ciklus vége
Eljárás vége.
```

10.45

Egy H hosszúságú rudat bevonunk N réteggel egyenletesen (a végét is). D(I) jelenti az I. bevonás utáni átmérőt, D(0) a kezdeti átmérő. Keressük meg a hibát a következő algoritmusban, amely meghatározza az egyes felhasznált anyagok térfogatát!

```
Eljárás:
  V:=D(0)*D(0)/4*PI*H
  Ciklus I=1-től N-ig
    U:=D(I)*D(I)/4*PI*H: Ki: V-U: V:=VU: H:=H+D(I)
  Ciklus vége
Eljárás vége.
```

10.46

A program egy szám összes valódi nem prím osztóit írja ki. Keressünk hibákat a programban!

```
Eljárás:
  Be: N: I:=4
  Ciklus amíg I≤N/2 és I nem osztója N-nek
    Ciklus J=1-től Négyzetgyök(I)-ig
      Ha I osztója J-nek akkor Ki: I
    Ciklus vége
    I:=I+1
  Ciklus vége
Eljárás vége.
```

10.47

A program számok mértani közepét számolja ki, az utolsó után 10000-et kell beadni. Keressünk hibát a következő programban!

```
Eljárás:
  S:=1: I:=0: Be: N
  Ciklus amíg N<10000
    S:=S*N: I:=I+1: Be: N
  Ciklus vége
  Ha S<0 és nem Egész(N/2) akkor Ki: "Nem valós"
  különben A:=sgn(S)*abs(S)^(1/I): Ki: A
Eljárás vége.
```

10.48

A programrészlet egy vektor 0 elemeinek számát adja meg. Keressünk hibákat a következő programban!

```
Eljárás:
  I:=1: S:=0: D:=0
  Ciklus
    Ciklus amíg A(I)>0
      I:=I+1: S:=D+1
    Ciklus vége
  amíg I≥N
  Ciklus vége
  Ki: S
Eljárás vége.
```

10.49

A programrészlet egy mátrix sormaximumainak minimumát számolja. Keressünk hibákat a következő megoldásban!

```
Eljárás:
  R:=MTRX(1,1)
  Ciklus I:=1-től N-ig
    S:=MTRX(I, 1)
    Ciklus J:=1-től K-ig
      Ha S<MTRX(I, J) akkor S:=MTRX(I, J)
    Ciklus vége
    Ha R>S akkor R:=S
  Ciklus vége
  Ki: R
Eljárás vége.
```

10.50

A programrészlet hőmérsékleti adatokat dolgoz fel, jelzi, hogy az illető beteg, ha a mérés eredménye kisebb 35 foknál, vagy nagyobb 40 foknál, vagy pedig két mérés között a hőmérséklet több mint egy fokkal változott. Keressünk hibákat a következő programban!

```
Eljárás:
  Ciklus I:=1-től N-ig
    Be: A(I)
  Ciklus vége
  Ciklus I:=1-től N-ig
    Ha A(I)>40 vagy A(I)<35 vagy A(I)-A(I-1)>1
       akkor Ki:"BETEG"
  Ciklus vége
Eljárás vége.
```

10.51

A programrészlet A és B nem közös osztóit írja ki. Keressünk hibákat a következő megoldásban!

```
Eljárás:
  S:=0: D:= A div 2
  Ciklus I:=2-től D-ig
    Ha I osztója A-nak akkor S:=S+1
    Ha I osztója B-nek akkor S:=S+1
    Ha S=1 akkor Ki: I
  Ciklus vége
Eljárás vége.
```

10.52

A program két szám legnagyobb közös osztóját számolja ki. Keressünk hibákat a következő megoldásban!

```
Eljárás:
  Be: A, B
  X:=A: Y:=B
  Ciklus amíg X ≠ Y
    Ha X<Y akkor Y=Y-X
  Ciklus vége
  Ha Y<X akkor Y:=Y-X
  Ki: X
Eljárás vége.
```

10.53

A program egy vektor összes maximális értékű elemének az indexét írná ki, ha jól működne. Keressünk hibákat a programban!

```
Eljárás:
  Be: N
  K:=N
  Ciklus I:=1-től N-ig
    Be: A(I)
  Ciklus vége
  Ciklus
    K:=0
    Ciklus I=1-től N-ig
      Ha A(I)>A(I+1) akkor X:=A(I): A(I+1):=A(I)
                           A(I):=X: K:=1
      Elágazás vége
    Ciklus vége
    N:=N-1
  amíg K≤0
  Ciklus vége
  Ciklus I:=K-tól 1-ig -1-esével
    Ha A(I)=A(I-1) akkor Ki: I
  Ciklus vége
Eljárás vége.
```

10.54

A program egy vektor összes maximális értékű elemének az indexét írná ki, ha jól működne. Keressünk hibákat a programban!

```
Eljárás:
  Be: N, E
  Ciklus I:=2-től N-ig
    Be: M
    Ha M=E akkor Ki: I
         különben Ha M>E akkor M:=E
  Ciklus vége
Eljárás vége.
```

10.55

A program egy vektor összes maximális értékű elemének az indexét írja ki. Keressünk hibákat a programban!

```
Eljárás:
  Be: N
  Ciklus I=1-től N-ig
    Be: A(I): L(I):=I
  Ciklus vége
  Ciklus I=1-től N-ig
    Ciklus J=I+1-től N-ig
      Ha A(I)<A(J) akkor X:=A(I): A(J):=A(I): A(I):=X
                         X:=L(I): L(I):=L(J): L(J):=X
      Elágazás vége
    Ciklus vége
  Ciklus vége
  Ki: L(1): Q:=2
  Ciklus amíg  L(Q)=L(1)
    Ki: L(Q): Q:=Q+1
  Ciklus vége
Eljárás vége.
```

10.56

A program egy vektor összes maximális értékű elemének az indexét írná ki, ha jól működne. Keressünk hibákat a programban!

```
Eljárás:
  Be: N: K:=N
  Ciklus I=O-tól N-ig
    Be: A(I)
  Ciklus vége
  Ciklus
    K:=0
    Ciklus I=0 TO N-1
      Ha A(I)>A(I+1) akkor X:=A(I): A(I):=A(I+1)
                           A(I+1):=X: K:=1
      Elágazás vége
     Ciklus vége
    N:=N-1
  amíg K≤0
  Ciklus vége
  Ciklus I=K-tól 0-ig -1-esével
    Ha A(I)=A(I-1) akkor Ki: I
  Ciklus vége
Eljárás vége
```

10.57 

A program egy vektor összes maximális értékű elemének az indexét írná ki, ha jól működne. Keressünk hibákat a programban!

```
Eljárás:
  Be: N
  Ciklus I=1-től N-ig
    Be: A(I): B(I):=I
  Ciklus vége
  Ciklus K=N-1-től 1-ig -1-esével
    Ciklus B=1-től K-ig
      Ha A(B)<A(B+1) akkor X:=A(B): A(B):=A(B+1)
                           A(B+1):=X
                           Y:=B(B): B(B):=B(B+1)
                           B(B+1):=X
      Elágazás vége
    Ciklus vége
  Ciklus vége
  Ciklus F=1-től 10-ig
    Ki: B(F)
  Ciklus vége
Eljárás vége.
```

10.58 

A program egy vektor összes maximális értékű elemét gyűjtené ki, ha jól működne. (Dimenzionálási hibák nincsenek!) Keressünk hibákat a programban!

```
Eljárás:
  S:=0: L:=S
  Ciklus J=1-től N-ig
    Ha A(J)>L akkor S:=S+1: L:=A(J): K(S):=J
    Ha A(J)=L akkor S:=S+1: K(S):=J
  Ciklus vége
Eljárás vége.
```

10.59 

Milyen hibákat találunk a következő rendező eljárásban?

```
Rendezés:
  Ciklus I=N-től 1-ig -1-esével
    Ciklus J=1-től I-ig
      Ha A(J)>A(J+1) akkor X:=A(I): A(I+1):=A(I)
                           A(I):=A(I+1)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

10.60

A Balatonra egy négyzethálót fektettünk és minden pontjában megmértük a víz hőmérsékletét. Ha egy mérés parton volt, akkor az értéke=0, egyébként >0. A négyzetháló szélső pontjai parton voltak. Keressünk hibákat a következő programban, amely a partmenti víz átlaghőmérsékletét határozná meg, ha működne!

```
Eljárás:
  Ciklus I=2-től N-1-ig
    Ciklus J=2-től M-1-ig
      Ha X(I,J)≠0
          akkor Ciklus K=I-1-től I+1-ig
                  Ciklus L=J-1-től J+1-ig
                   Ha X(K,L)≠0 akkor D:=D+1:
S:=S+X(K,L)
                Ciklus vége
              Ciklus vége
      Elágazás vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

10.61

A Balatonra egy négyzethálót fektettünk és minden pontjában megmértük a víz hőmérsékletét. Ha egy mérés parton volt, akkor az értéke=0, egyébként>0. A négyzetháló szélső pontjai parton voltak. Keressünk hibákat a következő programban, amely a leghidegebb pont hőmérsékletét határozná meg, ha működne!

```
Eljárás:
  X:=10000
  Ciklus I=1-től N-ig
    Ciklus J=1-től M-ig
      Ha A(I,J)<X akkor X:=A(I,J)
    Ciklus vége
  Ciklus vége
  Ki: X
Eljárás vége.
```

10.62

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. (összesen N mérés volt) A mérést szárazföld fölött kezdtük és ott is fejeztük be. Ott van tenger, ahol a mérés értéke=0. Keressünk hibákat a következő programban, amely a tenger szélességét határozná meg a mért szakaszon!

```
Eljárás:
  E:=1
  Ciklus amíg A(E)>0
    E:=E+1
  Ciklus vége
  U:=N
  Ciklus amíg A(U)>0
    U:=U-1
  Ciklus vége
  Ki: (U-E)*5
Eljárás vége.
```

10.63

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. (összesen N mérés volt) A mérést szárazföld fölött kezdtük és ott is fejeztük be. Ott van tenger, ahol a mérés értéke=0. Keressünk hibákat a következő programban, amely a hegycsúcsok helyét és magasságát határozná meg a mért szakaszon, ha működne!

```
Eljárás:
  Ciklus I=1-től N-ig
    Ha A(I-1)<A(I) és A(I)>A(I+1) akkor Ki: I,A(I)
  Ciklus vége
Eljárás vége.
```

10.64

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. (összesen N mérés volt) A mérést szárazföld fölött kezdtük és ott is fejeztük be. Ott van tenger, ahol a mérés értéke=0. Keressünk hibákat a következő programban, amely a szigetek számát határozná meg a mért szakaszon, ha működne!

```
Eljárás:
  S:=0
  Ciklus I=2-től N-ig
    Ha A(I-1)=0 és A(I)>0 akkor S:=S+1
  Ciklus vége
  Ki: S
Eljárás vége.
```

10.65

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. (összesen N mérés volt) A mérést szárazföld fölött kezdtük és ott is fejeztük be. Ott van tenger, ahol a mérés értéke=0. Keressünk hibákat a következő prog-

ramban, amely azon szigeteket határozná meg a mért szakaszon, amelyeken nincs 100 méternél magasabb hely, ha működne!

```
Eljárás:
  Ciklus I=BALPART_HELYE-től JOBBPART_HELYE-ig
    Elágazás
      A(I)>H esetén A(I):=H
      A(I-1)=0 és A(I)>0 esetén X:=I
      A(I-1)>0 és A(I)=0 esetén Ha H<100 akkor Ki: X,
I
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.66

Egy H hosszúságú rudat bevonunk N réteggel egyenletesen (a végét is). D(I) jelenti az I. bevonás utáni átmérőt, D(0) a kezdeti átmérő. Keressünk hibákat a következő programban, amely az egyes felhasznált anyagok térfogatát határozza meg!

```
Eljárás:
  V:=D(0)*D(0)/4*PI*H
  Ciklus I=1-től N-ig
    U:=D(I)*D(I)/4*PI*H
    Ki: V-U: V:=VU: H:=H+D(I)
  Ciklus vége
Eljárás vége.
```

10.67

N napig mértük, naponta K alkalommal a hőmérsékletet. Keressünk hibákat az alábbi programban, amely a napi maximumok minimumát határozza meg!

```
Eljárás:
  L:=A(1,1)
  Ciklus I=1-től N-ig
    M:=A(I,1)
    Ciklus J=1-től K-ig
      Ha A(I,J)>M akkor M:=A(I,J)
    Ciklus vége
    Ha M<L akkor L:=M
  Ciklus vége
  Ki: L
Eljárás vége.
```

10.68

N napig mértük, naponta K alkalommal a hőmérsékletet. Keressünk hibákat a következő programban, amely azokat a napokat határozná meg, amelyek átlaghőmérséklete nagyobb volt az előző és a következő napinál is!

```
Eljárás:
  F:=0: E:=0
  Ciklus I=3-tól N-ig
    S:=0
    Ciklus J=1-től K-ig
      S:=S+A(I, J)
    Ciklus vége
    S:=S/N
    Ha F<E és E>S akkor Ki: I-1
    F:=E: E:=S
  Ciklus vége
Eljárás vége.
```

10.69

Keressünk hibákat a következő algoritmusban!
"Egy V0 térfogatú, M0 tömegű jégtábla úszik a vízen. N db tárggyal terheljük, amelyek tömegeit a T(N) tartalmazza, mindegyik R0 sűrűségű. Adjuk meg, hogy a tárgyak egyenkénti ráhelyezésével elsüllyed-e a jégtábla, s ha igen, akkor mikor!"

```
Eljárás:
  Ciklus I=1-től N-ig
    V:=V0+T(I)/R0
    Ha M+T(I)/V0<1 akkor Ki: "Elsűllyed",I
    M:=M+T(I)
  Ciklus vége
  Ki: "Nem sülyed el"
Eljárás vége.
```

10.70

Egy V0 térfogatú, M0 tömegű jégtábla úszik a vízen. N darab tárggyal terheljük, amelyek tömegeit a T(N) vektorban helyeztük el. Mindegyik tárgy R0 sűrűségű. Keressünk hibákat a következő programban, amely megadja, hogy elsüllyed-e a jégtábla, s ha igen, akkor mikor!

```
Eljárás:
  Ciklus I=1-től N-ig
    V:=V0+T(I)/R0
    Ha (M0+T(I)/V)<1
       akkor Ki: "Elsüllyed az ",I,". tárgy után"
    M:=M+T(I)
  Ciklus vége
  Ki: "Nem süllyed el!"
Eljárás vége.
```

10.71

Keressünk hibákat a következő algoritmusban (N rekeszben tárolunk fehér és színes üvegeket, vegyesen. Meg kell adni azt a minimális csereszámot, ahány fehér üveget színessel felcserélve elérhető az az állapot, amikor már csak egyetlen rekeszben van mindkét fajtából. Minden rekesz pontosan M db. üveget tartalmaz.):

```
Eljárás:
  Ciklus i=2-től n-ig
    j:=i-1: x:=rekesz[i]
    Ciklus amíg (rekesz[j]>x)
      rekesz[j+1]:=rekesz[j]: j:=j-1
    Ciklus vége
    rekesz[j]:=x
  Ciklus vége
  e:=1: u:=n: csere:=0
  Ciklus amíg e≤u
   Ha rekesz[e]<m-rekesz[u]
     akkor csere:=csere+rekesz[e]
           rekesz[u]:=rekesz[u]+rekesz[e]: e:=e+1
   különben ha rekesz[e]>rekesz[u]
     akkor csere:=csere+m-rekesz[u]
           rekesz[e]:=rekesz[e]-(m-rekesz[u])
   különben csere:=csere+rekesz[e]: e:=e+1: u:=u-1
   Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.72

Keressünk hibákat a következő algoritmusban (N rekeszben tárolunk fehér és színes üvegeket, vegyesen. Meg kell adni azt a minimális csereszámot, ahány fehér üveget színessel felcserélve elérhető az az állapot, amikor már csak egyetlen rekeszben van mindkét fajtából. Minden rekesz pontosan M db. üveget tartalmaz.):

```
Eljárás:
  Ciklus i=2-től n-ig
    j:=i-1: x:=rekesz[j]
    Ciklus amíg (j>0) and (rekesz[j]>x)
      rekesz[j]:=rekesz[j+1]: j:=j-1
    Ciklus vége
    rekesz[j+1]:=x
  Ciklus vége
  e:=1: u:=n: csere:=0
  Ciklus amíg e<u
   Ha rekesz[e]<rekesz[u]
     akkor csere:=csere+rekesz[u]
           rekesz[u]:=rekesz[u]+rekesz[e]: e:=e+1
   különben ha rekesz[e]>m-rekesz[u]
     akkor csere:=csere+m-rekesz[u]
           rekesz[e]:=rekesz[e]-rekesz[u]: u:=u-1
   különben csere:=csere+rekesz[e]: u:=u-1
   Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.73

A következő eljárás helyesen működik. Szövegek rendezett sorozatába beilleszt egy új szöveget. Milyen kezdőértékadást kell alkalmazni ahhoz, hogy az eljárás akkor is helyesen működjön, amikor a rendezett sorozatban még egyetlen elem sincs (N=0)?

```
Beillesztés (SZVG):
  N:=N+1: SZVG_SOR(N):=SZVG: B:=0: A:=SOR_SZ(B)
  Ciklus amíg A≠-1 és SZVG>SZVG_SOR(A)
    B:=A: A:=SOR_SZ(B)
  Ciklus vége
  SOR_SZ(B):=N: SOR_SZ(N):=A
Eljárás vége.
```

10.74

A következő programrészlet két növekvően rendezett számsorozat közös elemeit írja ki. Mi a feltétele annak, hogy egy elemet sem ír ki többször?

```
Eljárás:
  I:=1: J:=1
  Ciklus amíg I≤N és J≤M
    Elágazás
      A(I)=B(J) esetén: Ki: A(I)
      A(I)<B(J) esetén: I:=I+1
      A(I)<B(J) esetén: J:=J+1
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.75

Az alábbi program feladata, hogy meghatározza: hány helyen lesz olajfoltos két, egymáshoz kapcsolódó fogaskerék, ha az első kerék egyetlen fogára egy olajcseppet teszünk. (Feltesszük, hogy a szerkezet „elég sokáig" működik és az eredeti olajcsepp elég nagy ahhoz, hogy akárhány foltot okozzon.) Futtatáskor kiderült, hogy a program nem ad jó eredményt. Keressük meg a hibát, vagy a hibákat!

```
Eljárás:
  Ciklus
    Ki: "AZ ELSO KEREK FOGAINAK SZAMA "
    Be: EKSZ
  amíg EKSZ≥2 és Egész(EKSZ)
  Ciklus vége
  Ciklus
    X:=MKSZ: Y:=EKSZ
    Ciklus amíg X≠Y
      Ha X>Y akkor X:=A-X különben Y:=B-Y
      X:=MKSZ: Y:=EKSZ
    Ciklus vége
    Ki: "AZ OLAJOS FOGAK SZÁMA"
    Ki: "AZ ELSO KERÉKEN: ", X
    Ki: "A MASODIK KERÉKEN: ", Y
    Ki: "KÉR-E ÚJ SZÁMOLÁST (I/N)"
    Be: VALASZT
  amíg VALASZT≠"N"
  Ciklus vége
Eljárás vége.
```

10.4 Hatékonyabbra írás

Írjunk algoritmust az első N természetes szám négyzete összegének kiszámítására!

```
Eljárás:
  S:=0
  Ciklus I=1-től N-ig
    S:=S+I²
  Ciklus vége
Eljárás vége.
```

Az eljárás helyes eredményt ad, de nem elég hatékony. I^2 helyett I*I-t használva máris gyorsabbá tehetjük. A feladatot sokkal egyszerűbben is meg lehet oldani egy matematikai tétel felhasználásával:

```
Eljárás:
  S:=N*(N+1)*(2*N+1)/6
Eljárás vége.
```

Az ilyen és ehhez hasonló esetekben van szükség a hatékonyság vizsgálatára. Egész sor olyan módszer létezik, amelyek alkalmazásával a programot hatékonyabbá írhatjuk. Hatékonyabbra íráson a végrehajtási idő, a helyfoglalás (programszöveg és adatok), valamint a bonyolultság (elkészítési-, megértési idő, szükséges tudás) csökkentését értjük.

10.76

Írjuk át hatékonyabbá a következő programot, amely egy természetes szám legnagyobb osztóját adja meg. A legnagyobb osztó nem maga a szám.

```
Eljárás:
  I:=N
  Ciklus amíg I nem osztója N-nek
    I:=I-1
  Ciklus vége
  Ki: I
Eljárás vége.
```

10.77

Írjuk át hatékonyabbá a következő programot, amely két természetes szám legnagyobb közös osztóját számolja ki, felhasználva, hogy A és B közös osztói A-B-nek is osztói, ha A>B.

```
Eljárás:
  Ciklus amíg A≠B
    Ha A<B akkor B:=B-A különben A:=A-B
  Ciklus vége
  Ki: A
Eljárás vége.
```

10.78

Adott egy [A, B] intervallumon folytonos függvény, F(A)*F(B)<0. Keressünk az [A, B] intervallumban egy olyan helyet, amely a függvény gyökhelyétől E-nél kevesebbel tér el! Írjuk át hatékonyabbra a következő algoritmust!

```
Gyökkeresés eljárás:
  K:=A+E
  Ciklus amíg F(K)*F(K-E)>0
    K:=K+E
  Ciklus vége
  Gyökhely:=K
Eljárás vége.
```

10.79

Írjuk át hatékonyabbra az N-ig található ikerprímeket előállító algoritmust!

```
Ikerprímek eljárás:
  Ciklus I=2-től N-ig
    Ha prím(I) és prím(I+2) akkor Ki: I,I+2
  Ciklus vége
Eljárás vége.
```

10.80

Adott egy szimmetrikus mátrix (A(I, J)=A(J, I)). Írjuk át hatékonyabbá a legnagyobb értékű elem indexeit meghatározó eljárást!

```
Eljárás:
  MS:=1: MO:=1
  Ciklus I=1-től N-ig
    Ciklus J=1-től M-ig
      Ha A(I,J)>A(MS,MO) akkor MS:=I: MO:=J
    Ciklus vége
  Ciklus vége
  Ki: MS, MO, A(MS, MO)
Eljárás vége.
```

10.81

Írjuk át hatékonyabbá a következő eljárást, amely egy mátrixban egy X érték helyét adja meg. A mátrix sorainak elemei nagyság szerint rendezettek. Az egyes sorokhoz tartozó számintervallumok nem fedik át egymást. A sorok egymáshoz képest nincsenek sorrendben.

```
Eljárás:
  I:=1: TALÁLT:=HAMIS
  Ciklus amíg I≤N és nem TALÁLT
    J:=1
    Ciklus amíg J≤M és A(I,J)≠X
      J:=J+1
    Ciklus vége
    Ha J≤M akkor TALÁLT:=IGAZ különben I:=I+1
  Ciklus vége
  Ki: I, J
Eljárás vége.
```

10.82

Írjuk át a következő eljárást hatékonyabbá, amely egy Pascal-háromszög N. sorának elemeit határozza meg minden K-ra (K=0, 1, ..., N). Az N. sor K. elemének értéke:

$$\binom{n}{k} = \frac{n!}{k!*(n-k)!}$$

```
Eljárás:
  Ciklus I=0-tól N-ig
    S1:=1: S2:=1: S3:=1
    Ciklus I=1-től N-ig
      S1:=S1*I
    Ciklus vége
    Ciklus I=1-től K-ig
      S2:=S2*I
    Ciklus vége
    Ciklus I=1-től N-K-ig
      S3:=S3*I
    Ciklus vége
    Ki: S1/(S2*S3)
  Ciklus vége
Eljárás vége.
```

10.83

Írjuk át hatékonyabbá a következő eljárást, amely egy vektorban megkeresi egy X érték helyét. Tudjuk, hogy a vektor minden elemére igaz, hogy I<A(I)<2*I.

```
Eljárás:
  I:=1
  Ciklus amíg I≤N és A(I)≠X
    I:=I+1
  Ciklus vége
  Ha I≤N akkor Ki: I
Eljárás vége.
```

10.84

Írjuk át hatékonyabbra az első N (N>2) Fibonacci-számot előállító eljárást!

```
Fibonacci eljárás:
  Ciklus I=0-tól N-ig
    Elágazás
      I=0 esetén F(0):=1
      I=1 esetén F(1):=1
      egyéb esetben Fibonacci előállítás(I)
    Elágazás vége
  Ciklus vége
Eljárás vége.
Fibonacci előállítás:(I)
  F1:=1: F2:=1
  Ciklus J=2-től I-ig
    F:=F1+F2: F2:=F1: F1:=F
  Ciklus vége
  F(I):=F
Eljárás vége.
```

10.85

A K-Fibonacci-számokat a következőképpen képezzük:

$F(N):=F(N-1)+F(N-2)+...+F(N-K)$

Írjuk át hatékonyabbá a következő eljárást!

```
Eljárás:
  Ciklus I=0-tól K-1-ig
    F(I):=1
  Ciklus vége
  Ciklus I=K-tól N-ig
    F(I):=0
    Ciklus J=N-K-tól N-1-ig
      F(I):=F(I)+F(J)
    Ciklus vége
  Ciklus vége
  Ki: F(N)
Eljárás vége.
```

10.86

Írjuk át hatékonyabbá a következő eljárást, amely adott N mérésből meghatározza a tengerben levő szigetek kezdetét és végét. A mérések tengerszint feletti magasságokat tartalmaznak. A 0 értékű mérések tengert jelentenek.

```
Eljárás:
  Ciklus I=1-től N-ig
    Ha I=1 és A(I)>0 vagy I>1 és A(I-1)=0 és A(I)>0
         akkor Ki: "KEZDET:", I
    Ha I=N és A(I)>0 vagy I<N és A(I)>0 és A(I+1)=0
         akkor Ki: "VEG:", I
  Ciklus vége
Eljárás vége.
```

10.87

Írjuk át gyorsabbra a következő eljárást! Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. A méréssorozatot szárazföld fölött kezdtük és fejeztük be. Volt közben tenger is. Ott van tenger, ahol a mérés értéke: 0. Határozzuk meg minden egyes szigeten a legmagasabb pontokat!

```
Eljárás:
  S:=0
  Ciklus I=2-től N-1-ig
    Ha A(I-1)=0 és A(I)>0 akkor S:=S+1: K(S):=I
    Ha A(I)>0 és A(I+1)=0 akkor V(S):=I
  Ciklus vége
  Ciklus I=1-től S-1-ig
    M:=K(I)
    Ciklus J=K(I)+1-től V(I)-ig
      Ha A(J)>A(M) akkor M:=J
    Ciklus vége
    Ki: K(I), V(I), M, A(M)
  Ciklus vége
Eljárás vége.
```

10.88

Írjuk át gyorsabbra a következő eljárást! Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. A méréssorozatot szárazföld fölött kezdtük és fejeztük be. Volt közben tenger is. Ott van tenger, ahol a mérés értéke: 0. Határozzuk meg az X méternél magasabb szigetek kezdeteit és végeit!

```
Eljárás:
  E:=Első 0 helye: U:= Utolsó 0 helye: I:=E
  Ciklus amíg I≤U
    Ciklus amíg A(I)<F és I≤U
      I:=I+1
    Ciklus vége
    Ha I≤U akkor K:=I-1
                 Ciklus amíg A(K)>0
                   K:=K-1
                 Ciklus vége
                 V:=I+1
                 Ciklus amíg A(V)>0
                   V:=V+1
                 Ciklus vége
                 Ki: K, V
                 I:=V
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.89

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. A méréssorozatot szárazföld fölött kezdtük és fejeztük be. Volt közben tenger is. Ott van tenger, ahol a mérés értéke: 0. Adjuk meg az egyes szigetek hosszait! Írjuk át a következő algoritmust kevesebb helyet foglalóra!

```
Eljárás:
  DB:=0
  Ciklus I=E+1-től V-1-ig
    Ha MAG(I)>0 és MAG(I-1)=0 akkor
         DB:=DB+1: KEZDET(DB):=I
    Elágazás vége
    Ha MAG(I)>0 és MAG(I+1)=0 akkor VÉG(DB):=I
  Ciklus vége
  Ciklus I=1-től DB-ig
    HOSSZ(I):=VÉG(I)-KEZDET(I)+1
  Ciklus vége
Eljárás vége.
```

10.90

Öt kilométerenként megmértük a felszín tengerszint feletti magasságát. A méréssorozatot szárazföld fölött kezdtük és fejeztük be. Volt közben tenger is. Ott van tenger, ahol a mérés értéke: 0. A tenger E-től V-ig tart. Adjuk meg

azon szigetek kezdeteit és végeit, ahol a legmagasabb pont van! Írjuk át hatékonyabbra a következő megoldást!

```
Szigetek eljárás:
  MAX:=E
  Ciklus I=E+1-től V-ig
    Ha MAG(I)>MAG(MAX) akkor MAX:=I
  Ciklus vége
  DB:=0
  Ciklus I=E-től V-ig
    Ha MAG(I)=MAG(MAX) akkor DB:=DB+1: X(DB):=I
  Ciklus vége
  SZDB:=0
  Ciklus I=1-től DB-ig
    K:=X(I)-1
    Ciklus amíg MAG(K)>0
      K:=K-1
    Ciklus vége
    V:=X(I)+1
    Ciklus amíg MAG(V)>0
      V:=V-1
    Ciklus vége
    Ha SZDB=0 akkor SZDB:=1: SZK(SZDB):=K:
SZV(SZDB):=V
    különben ha SZV(SZDB)≠V akkor
                    SZDB:=SZDB+1: SZK(SZDB):=K
                    SZV(SZDB):=V
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.91

Írjuk át hatékonyabbra (végrehajtási idő) a következő feladatot megoldó algoritmust! Két rádióadó adásidőit mértük. Az első adáskezdeteit az EK(N) aásvégeit az EV(N) vektorban tároljuk, a másodikét pedig az MK(M), illetve az MV(M) vektorokban. Mindkettő az adáskezdetek szerint sorba van rendezve. Állapítsuk meg, hogy a két adássorozatot adhatta-e ugyanaz az adó! Az eredményt az eljárás végén az `Adhatta` nevű változó tartalmazza!

```
Adhatta-e eljárás:
 Adhatta:=Igaz
 Ciklus I=1-től N-ig
  Ciklus J=1-től M-ig
   Ha EK(I)≥MK(J) és EK(I)<MV(J) vagy
     EV(I)>MK(J) és EV(I) ≤MV(J) vagy
     MK(J)>EK(I) és MK(J)<EV(I) akkor Adhatta:=Hamis
  Ciklus vége
 Ciklus vége
Eljárás vége.
```

10.92

Írjuk át hatékonyabbra a következő algoritmust! (Nem tudjuk, mit csinál.)

```
Eljárás:
  F:=0
  Ciklus I=1-től N-ig
    Elágazás
      A(I)>0 esetén A(I):=A(I)*N/2
      A(I)=0 esetén F:=I*I*I
      A(I)<0 esetén A(I):=A(I)-F
      A(I)=0 esetén A(I):=F^2
    Elágazás vége
  Ciklus vége
  Ciklus I=1-től N-ig
    Ha A(I)<0 akkor A(I):=-A(I)
  Ciklus vége
  Ciklus I=1-től N-ig
    Ha A(I)<F és F>0 akkor A(I):=F
  Ciklus vége
Eljárás vége.
```

10.93

A Balatonra egy négyzethálót fektettünk és minden rácspontban megmértük a víz hőmérsékletét. A parton a mért érték 0, a vízben >0. MAX tartalmazza a viz maximális hőmérsékletét. Írjuk át gyorsabbra a parthoz legközelebbi maximumhelyet kereső eljárást!

```
Keresés eljárás:
  TMIN:=+∞
  Ciklus I=1-től N-ig
    Ciklus J=1-től M-ig
      Ciklus K=1-től N-ig
        Ciklus L=1-től M-ig
          Ha X(I,J)=MAX és X(K,L)=0 és
             TÁV(X(I,J),X(K,L))<TMIN akkor
                      TMIN:=TÁV(X(I,J),X(K,L))
                      MINX:=I: MINY:=J
        Ciklus vége
      Ciklus vége
    Ciklus vége
  Ciklus vége
  Ki: MINX, MINY, TMIN
Eljárás vége.
```

10.94

Egy ember testhőmérsékletét 10 percenként mérjük. Akkor mondjuk betegnek, ha a hőmérséklet 34 foknál kevesebb vagy 38 foknál több, vagy pedig két mérés között 1 foknál többel változott. Írjuk át hatékonyabbra azt az algoritmust, amely megszámolja, hogy hányszor volt beteg az illető!

```
Beteg eljárás:
  DB:=0
  Ciklus I=1-től N-ig
    Ha H(I)<34 vagy H(I)>38 akkor DB:=DB+1
    Ha I>1 és ABS(H(I)-H(I-1))>1 akkor DB:=DB+1
  Ciklus vége
Eljárás vége.
```

10.95

Egy kutyaszépségversenyen N kutya indult, mindegyik M kategóriában. Az egyes kategóriák maximális pontszámát a MAX(M), az értékeléshez szükséges minimális pontszámot a MIN(M) vektor tartalmazza. Írjuk át hatékonyabbra azt az algoritmust, amely megadja azokat a kutyákat, amelyek egy kategóriában sem estek ki, de nem is nyertek sehol!

```
Kiválogatás eljárás:
  DB:=0
  Ciklus I=1-től N-ig
    J:=1
    Ciklus amíg J≤M és KU(I, J)≥MIN(J)
      J:=J+1
    Ciklus vége
    Ha J>M akkor DB:=DB+1: K(DB):=I
  Ciklus vége
  Ciklus J=1-től M-ig
    L:=MAX(J): VAN:=Hamis
    Ciklus amíg nem VAN
      Ciklus I:=1-től N-ig
        Ha KU(I,J)=L
           akkor VAN:=Igaz
                 P:=1
                 Ciklus amíg P≤DB és K(P)≠I
                   P:=P+1
                 Ciklus vége
                 Ha P≤DB kkor Ciklus amíg P<DB
                                K(P):=K(P+1): P:=P+1
                              Ciklus vége
                              DB:=DB-1
                 Elágazás vége
        Elágazás vége
      Ciklus vége
    Ciklus vége
  Ciklus vége
  Ki: K() [1-től DB-ig]
Eljárás vége.
```

10.96

Egy nyúl N évig élhet. Megadjuk minden évre a nyulak azévi számát, azt, hogy ezek hányszorosa éri meg a következő évet, valamint azt, hogy egy X éves nyúlnak mennyi utóda születik abban az évben. Kövessük nyomon a nyulak éves változását! Írjuk át kevesebb helyet foglalóra a következő algoritmust! [X(J):= ennyi J éves nyúl van]

```
Nyulak eljárás:
  Ciklus
    Ciklus I=1-től N-ig
      Ha I=1 akkor Y(1):=0
                   Ciklus J=1-től N-ig
                     Y(1):=Y(1)+UTODSZAM(J)*X(J)
                   Ciklus vége
           különben Y(I):=ELETBENMARAD(I-1)*X(I-1)
      Elágazás vége
    Ciklus vége
    Ciklus I=1-től N-ig
      X(I):=Y(I)
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

10.97

Egy NxN-es mátrixban számokat tárolunk, csak 0 vagy 1 értékűek lehetnek. Véletlenszerűen kiválasztunk egy 1 értékű számot (azaz megadjuk a helyét). Határozzuk meg, hogy a szomszédságában hány darab 1-es szám található! Írjuk át rövidebb futási idejűre a következő megoldást!

```
Szomszédszám(I, J) eljárás:
  Ciklus K=I-1-től I+1-ig
    Ciklus L=J-1-től J+1-ig
      Ha K≥1 és K≤N és L≥1 és L≤N és (K≠I vagy
           L≠J) akkor Ha A(K,L)=1 akkor S:=S+1
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

10.98

Egy méréssorozatban a hibákat úgy szeretnénk javítani, hogy minden értéket helyettesítünk önmaga, valamint a két szomszédos mérés súlyozott átlagával. Írjuk át időben gyorsabbra, valamint kevesebb helyet foglalóra a következő megoldást!

```
Hibajavítás eljárás:
  Ciklus I=1-től N-ig
    Elágazás
      I=1 esetén Y(I):=(4*X(I)+X(I+1))/5
      I=N esetén Y(I):=(X(I-1)+4*X(I))/5
      egyéb esetben Y(I):=(X(I-1)+4*X(I)+X(I+1))/6
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.99

Egy étteremben minden nap kiszámolják, hogy mennyi az aznapi bevétel. Számítsuk ki, hogy mennyi a munkanapi bevételek átlaga, valamint, hogy mennyi a munkaszünetes napok bevételeinek átlaga (ilyen a szombat és a vasárnap)! Írjuk át hatékonyabbra a következő megoldást!

```
Átlagok(N) eljárás:
  MATL:=0: M:=0: MSZATL:=0: MSZ:=0
  Ciklus I=1-től N-ig
    Ha (I MOD 7)=0 vagy (I MOD 7)=6
       akkor MSZ:=MSZ+1: MSZATL:=MSZATL+BEVÉTEL(I)
    különben M:=M+1: MATL:=MATL+BEVÉTEL(I)
  Ciklus vége
  MATL:=MATL/M: MSZATL:=MSZATL/MSZ
Eljárás vége.
```

10.100

Olvassunk be hónap sorszámokat, majd adjuk meg, hogy hány téli, tavaszi, nyári, őszi hónap volt! Írjuk át a következő megoldást időben gyorsabbra!

```
Eljárás:
  Ciklus I=1-től N-ig
    Be: H
    Elágazás
     H=1 vagy H=2 vagy H=12 esetén DB(1):=DB(1)+1
     H=3 vagy H=4 vagy H=5 esetén DB(2):=DB(2)+1
     H=6 vagy H=7 vagy H=8 esetén DB(3):=DB(3)+1
     H=9 vagy H=10 vagy H=11 esetén DB(4):=DB(4)+1
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.101

A XX. század néhány egymást követő évében naponta megmértük a napi középhőmérsékletet. Írjuk át hatékonyabbra azt az algoritmust, amely eldönti,

hogy a szökőévek átlaga nagyobb-e a nem szökőévek átlagánál! (Legalább 4 évet vizsgáltunk!)

```
Eljárás:
  SZ:=0: SZDB:=0: NSZ:=0: NSZDB:=0
  Ciklus I=K-tól V-ig
    Ha (I MOD 4)=0 akkor NAP:=366
                   különben NAP:=365
    ÖSSZ:=0
    Ciklus J=1-től NAP-ig
      ÖSSZ:=ÖSSZ+A(I, J)
    Ciklus vége
    ÖSSZ:=ÖSSZ/NAP
    Ha I MOD 4=0 akkor SZ:=SZ+ÖSSZ: SZDB:=SZDB+1
                 különben NSZ:=NSZ+ÖSSZ: NSZDB:=NSZDB+1
  Ciklus vége
  NAGYOBB:= (SZ/SZDB > NSZ/NSZDB)
Eljárás vége.
```

10.102

Adott N darab mérés, adjuk meg a legmagasabb csúcsot! Írjuk át a következő megoldást hatékonyabbra!

```
Legmagasabb csúcs:
  DB:=0
  Ciklus I=2-től N-1-ig
    Ha A(I-1)<A(I) és A(I)>A(I+1)
            akkor DB:=DB+1: CSÚCS(DB):=I
  Ciklus vége
  ACSM:=0
  Ciklus J=1-től DB-ig
    Ha A(CS×CS(J)) > ACSM
            akkor ACSM:=A(CSÚCS(J)): MAX:=J
  Ciklus vége
  MAXCS×CS:=CS×CS(MAX)
Eljárás vége.
```

10.103

Európából Amerikába repültünk repülőgéppel, s az út során X kilométerenként mértük a felszín tengerszint feletti magasságát. Elképzelésünk szerint ott van tenger, ahol a mérés 0, s ott föld, ahol a mérés >0. Írjuk át hatékonyabbá a szigetek kezdeteit és végeit megadó eljárást!

```
Eljárás:
  I:=2 : DB:=0
  Ciklus amíg I<N
    Ciklus amíg I<N és nem (A(I)>0 és A(I-1)=0)
      I:=I+1
    Ciklus vége
    Ha I<N akkor DB:=DB+1: KEZD(DB):=I
    Ciklus amíg I<N és nem (A(I)>0 és A(I+1)=0)
      I:=I+1
    Ciklus vége
    Ha I<N akkor VÉG(DB):=I
  Ciklus vége
Eljárás vége.
```

10.104

Adott egy szöveg, adjuk meg a szöveg szavainak számát! Írjuk át hatékonyabbra a következő megoldást!

```
Eljárás:
  DB:=0
  Ciklus I=1-től N-1-ig
    Ha SZOVEG(I)=" " és SZOVEG(I+1)≠" "
         vagy I=1 és SZOVEG(1)≠" " akkor DB:=DB+1
  Ciklus vége
Eljárás vége.
```

10.105

Adott egy számsorozat orgonasíp elrendezésben. Írjuk át hatékonyabbra azt az algoritmust, amely az egymást követő elemek egymástól való eltérését írja ki!

```
Eljárás:
  Ciklus I=1-től N-1-ig
    Ha A(I+1) > A(I) akkor Ki: A(I+1)-A(I)
                    különben Ki: A(I)-A(I+1)
  Ciklus vége
Eljárás vége.
```

10.106

Adott egy számsorozat. A páros sorszámú elemeinek adjuk meg a négyzetét, a páratlanoknak a -1-szeresét! Írjuk át hatékonyabbra a következő megoldást!

```
Eljárás:
  Ciklus I=1-től N-ig
    Ha I páros akkor B(I):=A(I)*A(I)
            különben B(I):=-A(I)
  Ciklus vége
Eljárás vége.
```

10.107

Adott két rendezett sorozat. Emberek személyi azonosító kódjait tartalmazzák születési idő szerint sorbarendezve. (Az azonosító kód 2-7. számjegye tartalmazza a születési év utolsó két számát, a születési hónapot és a születési napot.) Írjuk át időben rövidebbre azt az algoritmust, amely a két sorozat összefuttatását végzi!

```
Eljárás:
  A(N+1, ):=(9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9)
  B(M+1, ):=(9, 9, 9, 9, 9, 9, 9, 9, 9, 9, 9)
  I:=1 : J:=1 : K:=0
  Ciklus amíg I<N+1 vagy J<M+1
    K:=K+1
    II:=A(I,2)*100000+A(I,3)*10000+A(I,4)
        *1000+A(I,5)*100+A(I,6)*10+A(I,7)
    JJ:=B(J,2)*100000+B(J,3)*10000+B(J,4)
        *1000+B(J,5)*100+B(J,6)*10+B(J,7)
    Elágazás
      II<JJ esetén C(K, ):=A(I, ): I:=I+1
      II=JJ esetén C(K, ):=A(I, ): I:=I+1: J:=J+1
      II>JJ esetén C(K, ):=B(J, ): J:=J+1
    Elágazás vége
  Ciklus vége
Eljárás vége.
```

10.108

Egy táblázatban „*" és „ " jelek találhatók. A táblázat elemeit az algoritmusban látható szabály szerint kell átalakítani! Írjuk át időben gyorsabbra az eljárást!

```
Eljárás:
 Ciklus I=1-től N-ig
  Ciklus J=1-től M-ig
    S:= szomszédos csillagok száma(I, J)
```

```
      Elágazás
       T(I,J)="*" és (S<2 vagy S>3) esetén T(I,J):=" "
       T(I, J)=" " és S=3 esetén T(I,J):="*"
       T(I,J)="*" és (S=2 vagy S=3) esetén T(I,J):="*"
      Elágazás vége
    Ciklus vége
  Ciklus vége
Eljárás vége.
```

10.109

Egy síkon pontokat ábrázoló programhoz szükséges véletlenszerűen választani a 4 síknegyed között. Írjuk át időben rövidebbre a következő megoldást!

```
Eljárás:
  Q:=RND(0)
  Ha Q<0.25 akkor L:=1
  különben ha Q<0.5 akkor L:=2
  különben ha Q<0.75 akkor L:=3
  különben L:=4
  Elágazás vége
Eljárás vége.
```

10.110

Egy síkon pontokat ábrázoló programhoz szükséges véletlenszerűen választani a 4 síknegyedből pontot. Írjuk át időben rövidebbre az alábbi megoldást!

```
Eljárás:
  X:=RND(0)*A
  Y:=RND(0)*B
  L:=RND(4)
  Ha L=2 akkor X:=-X
  különben ha L=3 akkor X:=-X: Y:=-Y
  különben ha L=4 akkor Y:=-Y
  Elágazás vége
Eljárás vége.
```

10.111

Írjuk át gyorsabbra a következő programrészletet!

```
Ciklus X=1-től N-ig
  Ciklus Y=1-től N-ig
    B(X):=0: F(X,Y):=0
  Ciklus vége
Ciklus vége
```

10.112

Írjuk át gyorsabbra, helyben rövidebbre a következő programrészletet!

```
Ha I=1 és (J=1 vagy J=M) vagy I=N és (J=1 vagy J=M)
   akkor Ha (F*8/3)-INT(F*8/3) < 0.5
            akkor F:=INT(F*8/3)
         különben F:=INT(F*8/3+1)
         Elágazás vége
különben ha I=1 vagy I=N vagy J=1 vagy J=M
   akkor Ha (F*8/5)-INT(F*8/5) < 0.5
            akkor F:=INT(F*8/5)
         különben F:=INT(F*8/5+1)
         Elágazás vége
Elágazás vége
```

10.113

Írjuk át kevesebb helyet foglalóra a következő programrészletet!

```
Ciklus
  Ki: "P(A)": Be: P(1)
amíg P(1)<1
Ciklus vége
Ciklus
  Ki: "P(B)": Be: P(2)
amíg P(2)<1
Ciklus vége
Ciklus
  Ki: "P(C)": Be: P(3)
amíg P(3)<1
Ciklus vége
Ciklus
  Ki: "P(D)": Be: P(4)
amíg P(4)<1
Ciklus vége
Ciklus
  Ki: "P(E)": Be: P(5)
amíg P(5)<1
Ciklus vége
```

10.114

Írjuk át kevesebb helyet foglalóra a következő programrészletet!

```
Ciklus I=1-től N-ig
  Ciklus J=1-től M-ig
    K:=RND(0)
    Ha K<P(1) akkor T(I,J):="A": R(1):=R(1)+1
    különben ha K<P(1)+P(2) akkor T(I,J):="B"
                                 R(2):=R(2)+1
    különben ha K<P(1)+P(2)+P(3) akkor T(I,J):="C"
                                      R(3):=R(3)+1
    különben ha K<P(1)+P(2)+P(3)+P(4) akkor
                                      T(I,J):="D"
                                      R(4):=R(4)+1
    különben T(I,J):="E": R(5):=R(5)+1
    Elágazás vége
  Ciklus vége
Ciklus vége
```

10.115

Írjuk át gyorsabbra a következő programrészletet!

```
S(X,Y):=S
Ha A(X,Y)="*" és (S(X,Y)=2 vagy S(X,Y)=3)
   akkor A(X,Y)="*"
Ha A(X,Y)=" " és S(X,Y)=3 akkor A(X,Y)="*"
Ha A(X,Y)="*" és (S(X,Y)>3 vagy S(X,Y)<2)
   akkor A(X,Y)=" "
```

10.116

Írjuk át gyorsabbra a következő programrészletet!

```
I:=0
Ciklus
  I:=I+1: J:=1
  Ciklus amíg S(I)≠3 és Q(J)≠3 és J≤7
    J:=J+1
  Ciklus vége
amíg I>9 vagy S(I)=3 vagy Q(J)=3
Ciklus vége
Ha Q(J)=3 akkor Q(J):=0
Ha S(I)=3 akkor S(I):=0
```

10.117

Írjuk át kevesebb helyet foglalóra a következő programrészletet!

```
R:=RND(0)
Ha R<P1 akkor R(1):="A": R(2)="A"
              R(3)="A": R(4):="A"
különben ha R<P1+P2 akkor R(1):="A": R(2):="A"
                           R(3):="A": R(4):="B"
különben ha R<P1+P2+P3 akkor R(1):="A": R(2):="A"
                              R(3):="B": R(4):="B"
különben ha R<P1+P2+P3+P4 akkor R(1):="A": R(2):="B"
                                 R(3):="B": R(4):="B"
különben R(1):="B": R(2):="B": R(3):="B": R(4):="B"
Elágazás vége
```

10.118

Írjuk át gyorsabbra, kevesebb helyet foglalóra a következő programrészletet!

```
Q:=RND(0)
Ha Q<0.25 akkor L:=1
különben ha Q<0.5 akkor L:=2
különben ha Q<0.75 akkor L:=3
különben L:=4
Elágazás vége
```

10.119

Írjuk át a következő programrészletet úgy, hogy gyorsabb legyen!

```
Ciklus I=1-től N-ig
  B4:=0
  Ciklus J=1-től 4-ig
    Ha Rész(SZÖVEG_1(I),J)="B" akkor B4:=B4+1
  Ciklus vége
  Ciklus K=0-tól 4-ig
    Ha B4=K akkor SZÖVEG_2(I):=SZÖVEG_3(K)
  Ciklus vége
Ciklus vége
```

10.120

Írjuk át a következő programrészletet úgy, hogy gyorsabb legyen!

```
S:=0
Ciklus I=1-től N-ig
  S:=S+A(I)/N
Ciklus vége
P:=0
Ciklus I=1-től N-ig
  P:=P+(A(I)-S)^2/N
Ciklus vége
```

10.121

A következő programrészlet két növekvően rendezett számsorozat közös elemeit írja ki. (Egyikben sem fordul elő ugyanaz a szám kétszer.) Írjuk át a programot gyorsabbra!

```
I:=1: J:=1
Ciklus amíg I≤N és J≤M
  Ha A(I)=B(J) akkor Ki: A(I)
  Ha A(I)<B(J) akkor I:=I+1 különben J:=J+1
Ciklus vége
```

10.122

Írjuk át hatékonyabbra a következő programrészletet!

```
G:=9.81: Be: M, HO
Ciklus H=HO-tól 0-ig -1-esével
  EH(H):=M*G*H: EM(H):=M*G*HO-M*G*H
Ciklus vége
```

10.123

Írjuk át gyorsabbra a következő algoritmust!

```
G:=9.81: Be: M,H0
Ciklus H=0-tól H0-ig
  T(H):=SQR(2*H/G)
Ciklus vége
Ciklus H=0-tól H0-ig
  V(H):=SQR(2*G*H)
Ciklus vége
Ciklus H=0-tól H0-ig
  EM(H):=1/2*M*V(H)^2
Ciklus vége
```

10.124

Írjuk át hatékonyabbra a következő programrészletet!

```
G=9.81: Be: M, T0
Ciklus T=0-tól T0-ig
  V(T):=G*T
Ciklus vége
Ciklus T=0-tól T0-ig
  S(T):=1/2*G*T^2
Ciklus vége
Ciklus T=0-tól T0-ig
  E(T):=1/2*M*(G*T)^2
Ciklus vége
```

10.125

Írjuk át hatékonyabbra a következő programrészletet!

```
Be: X: PX:=0
Ciklus I=0-tól N-ig
  PX:=PX+A(I)*X^I
Ciklus vége
```

10.126

Írjuk át ha ékonyabbra a következő programrészletet!

```
Be: X, Y
Ha X>=0 és Y>=0 akkkor Ki: "1. síknegyed"
Ha X<0 és Y>=0 akkor Ki: "2. síknegyed"
Ha X<0 és Y<0 akkor Ki: "3. síknegyed"
Ha X>=0 és Y<0 akkor Ki: "4. síknegyed"
```

10.127

Írjuk át ha ékonyabbra a következő programrészletet!

```
Be: A, B
Ha A<B akkor MAX:=B különben MAX:=A
Ha B≤A akkor MIN:=B különben MIN:=A
Ha A<B akkor X:=A: A:=B: B:=X
```

10.128

Írjuk át ha ékonyabbra a következő programrészletet!

```
Ha A(I)<B(I) akkor CC(I):=AA(I): I:=I+1
   különben CC(I):=AA(I)-BB(I): I:=I+1: J:=J+1
```

10.129

Írjuk át ha ékonyabbra a következő programrészletet!

```
Ciklus
  Ciklus
    Be: A
  amíg A≥0 és A≤5 és Egész(A)
   Ciklus vége
```

```
    Elágazás A szerint
      1: X1:=X1+1
      2: X2:=X2+1
      3: X3:=X3+1
      4: X4:=X4+1
      5: X5:=X5+1
    Elágazás vége
  amíg A≤0
  Ciklus vége
  Ki: X1, X2, X3, X4, X5
```

10.130

Adott egy háromszög három oldala. Határozzuk meg a háromszög területét! Írjuk hatékonyabbá az alábbi megoldást!

```
Eljárás:
  X1:=(A+B+C)/2-A: X2:=(A+B+C)/2-B
  X3:=(A+B+C)/2-C
  TERULET:=Négyzetgyök((A+B+C)/2*X1*X2*X3)
Eljárás vége.
```

10.131

Írjuk át hatékonyabbra!

```
Eljárás:
  Ciklus
    Be: ELSO
  amíg ELSO<0 vagy ELSO>1
  Ciklus vége
  Ciklus
    Be: MASODIK
  amíg MASODIK<0
  Ciklus vége
  Ciklus
    Be: HARMADIK
  amíg HARMADIK<0 vagy HARMADIK>1
  Ciklus vége
  Ciklus
    Be: NEGYEDIK
  amíg NEGYEDIK<0
  Ciklus vége
Eljárás vége.
```

10.132

Határozzuk meg egy N fős osztályban az informatika, a testnevelés, a rajz és az ének év végi jegyeinek átlagát! Írjuk át hatékonyabbra az alábbi algoritmust!

```
Eljárás:
  INATL:=0: TNATL:=0: RAATL:=0: EZATL:=0
  Ciklus I=1-től N-ig
    INATL:=INATL+IN(I)/N
  Ciklus vége
  Ciklus I=1-től N-ig
    TNATL:=TNATL+TN(I)/N
  Ciklus vége
  Ciklus I=1-től N-ig
    RAATL:=RAATL+RA(I)/N
  Ciklus vége
  Ciklus I=1-től N-ig
    EZATL:=EZATL+EZ(I)/N
  Ciklus vége
  Ki: INATL, TNATL, RAALT, EZATL
Eljárás vége.
```

10.133

Írjuk át hatékonyabbra!

```
Eljárás:
  Ciklus I=1-től N-ig
    Ha (A(I)>50 vagy Páros(A(I))) és
       (A(I)>50 vagy A(I)≤200) akkor B(I):=A(I)
                               különben B(I):=0
  Ciklus vége
Eljárás vége.
```

10.134

Adott egy N elemű sorozat. Adjuk meg azon H hosszúságú részsorozatait, amelyek összege legalább SUM! Írjuk át hatékonyabbá az alábbi algoritmust!

```
Eljárás:
  Ciklus I=1-től N-H+1-ig
    Ciklus I=0-tól H-ig
      S:=S+SOROZAT(I+J)
    Ciklus vége
    Ha S>SUM akkor DB:=DB+1: KEZD(DB):=I
  Ciklus vége
Eljárás vége.
```

10.135

Az áramszolgáltató vállalat a villanyórákról a következőket tartja nyilván: a tulajdonos neve, lakcíme (irányítószám, város, utca, házszám, emelet, lakás), körzetazonosító, óra típusa (éjszakai, nappali, ipari), egységár, előző óraállás, aktuális óraállás, fizetendő összeg (egy tulajdonosnak többféle órája is lehet). Adjunk helytakarékos tárolást és írjunk olyan eljárást, amely bekéri egy fogyasztó adatait és elküldi neki a villanyszámlát!

10.136

Egy vállalat tárolja a tőle rendelő vállalat adatait a következő formában: Megrendelés száma, dátum, vevő neve, vevő címe, vevő számlaszáma, áru megnevezése, egységár, mennyiség, szállítási határidő. Hogyan lehet N db megrendelés esetén az adatokat helytakarékosabban tárolni? Írjuk meg azt az eljárást, amely a választott adatszerkezet alapján egy ilyen táblázatot ír ki!

11. FEJEZET

JÁTÉKOK

TÁBLÁS JÁTÉKOK
EGYÉB LOGIKAI JÁTÉKOK AZONOS SZEREPLŐKKEL
AMIKOR A JÁTÉKOSOK FELADATA ELTÉR
MATEMATIKAI JÁTÉKOK

11. JÁTÉKOK

A tanulók számára valószínűleg legvonzóbb fejezet elsősorban azért került a könyv végére, mert játékprogramot nem könnyű írni: jó, ha előtte már minél több programozási gyakorlatot szerez az olvasó. A professzionális szintű kalandprogramok írása túlmutat könyvünk keretein, túlegyszerűsített „utánérzéseket" viszont nem akarunk feladni.

A felsorolt példák tehát olyan játékok, amelyeknél a megoldáshoz (győzelemhez) logikus gondolkodásra van szükség, és persze nem árt, ha valaki már gép nélkül is próbálkozott a játékkal. Főleg az első három alfejezetben ún. stratégiai játékokkal foglalkozunk. Ezekben a játékosoknak meghatározó szerepük van a játék alakulására – szemben például a szerencsejátékokkal.

Az ilyen játékprogramok –funkciójuk szerint – háromfélék lehetnek:

- A legegyszerűbb esetben a gép csak szemléltet, szimulálja a játékeszközt: pl. sakktábla helyett a képernyőn mozgathatók a figurák.
- Valamivel fejlettebb az a változat, amelyben a program nem engedi meg a szabálytalan lépéseket. Esetleg tárolja is a játék menetét, s ezzel lehetővé teszi a korábbi állások visszahívását, a játék elemzését.
- Az igazi játékprogram már versenyre is kelhet az emberrel: a játék egyik résztvevője a számítógép lehet. Az ilyen feladat megoldásában a legfontosabb a nyerő (vagy azt minél jobban megközelítő) stratégia meghatározása, algoritmusának kidolgozása.

Abban a reményben, hogy lesznek, akik a legutóbbi változattal is megpróbálkoznak, szóljunk pár szót a nyerő stratégiáról is. Akkor mondhatjuk, hogy egy játéknak van nyerő stratégiája, ha bárhogyan (akármilyen jól is) játszik az ellenfél, semmiképpen sem veszít az, aki ezt a stratégiát követi. A stratégia az esetenkénti döntéseket (mit lépjek?) gyűjti össze, és minden elvileg előforduló helyzetre tartalmaz utasítást. Egy részleges stratégia az esetek csak bizonyos részére tartalmaz utasítást – ilyenkor már csak kisebb eséllyel bízhatunk, hogy elkerüljük a vereséget.

Sok játéknak (NIM, szám kitalálósok, egyszerűbb amőbák stb.) egyszerűen megfogalmazható, explicit nyerő stratégiája van, így ezek könnyen algoritmizálhatók. Más esetekben (awari, malom, Othello, dáma stb.) eléggé összetett út vezet a biztos győzelemhez (döntetlenhez), ilyenkor eleinte elég néhány jó ötletet figyelembe venni (részleges stratégia), majd egyre több ilyet beépítve, egyre jobban fog játszani a programunk. Végül például a sakk, a GO és az amőba annyira bo-

nyolultak, hogy stratégiáik elméletét csak nagyon érdeklődőknek ajánljuk: nem túlzás azt mondani, hogy nincs (nem ismert) nyerő stratégiájuk.

A feladatok nehézségét elsősorban az határozza meg, mennyire bonyolult játéknak, mennyire aktív változatáról van szó. újból hangsúlyozzuk, hogy érdemes játszani a saját programunk ellen, rengeteget tanulhatunk belőle: szinte biztos, hogy a játék további javításokhoz vezet a nyerő stratégián.

Összefoglalva: a fejezet célja az, hogy szórakoztató, játékos formában ösztönözzön a fegyelmezett, logikus gondolkodásra. Ehhez kívánunk az Olvasónak hasznos időtöltést, eredményes „játszást".

Javasolt irodalom:

1. Vargha B.: Játékkoktél.
Minerva, 1967
2. Mosonyi K.: Matematikai játékok (szakköri füzet).
Tankönyvkiadó, 1977
3. J. Nievergelt - J.C. Farrar - E.M. Reingold: Matematikai problémák megoldásainak számítógépes módszerei.
Műszaki Könyvkiadó, 1977
4. R. Szendrei J.: A játék matematikája.
Tankönyvkiadó, 1978
5. Csirmaz L.: Logikai játékok.
Középiskolai Matematikai Lapok, 1982
6. M. Eigen - R. Winkler: A játék.
Gondolat , 1981
7. Z. Novak: A malomtól a goig, 50 táblás játék.
Gondolat, 1982
8. Csákány A. - Vajda F.: Játékok számítógéppel.
Műszaki Könyvkiadó, 1985

11.1 Táblás játékok

Igyekeztünk érinteni a legtöbb közismert táblás játékot, természetesen reménytelen dolog teljességet kívánni. A passzív (csak adminisztráló) programok jó ujjgyakorlatok lehetnek. Ezek hasznosságát növelheti, ha további szolgáltatásokat nyújtanak: pl. képesek egy állást megjegyezni, majd az állás könnyen visszahívható – ez elemzésekhez hasznos. Ugyancsak kellemes, ha figyelmeztetnek nagyon durva hibákra.

Ezen részben olyan játékok lesznek, melyek (több változata) egy speciális játéktérhez (tábla, figurák) kötődnek, és két, lényegében egyforma szerepű játékos játssza. A játékosok kizárólag úgy különböztethetők meg, hogy kezdő, illetve nem kezdő (második), minden más szempontból teljesen azonos a szerepük. Ennek megfelelően követendő stratégiájuk is lényegében azonos. Általában ez a stratégia csak egyikük (leggyakrabban a kezdő) számára hoz győzelmet, a másik kénytelen beérni a szoros vereséggel. Ha a kezdés joga nem jelent ilyen előnyt (hátrányt), akkor az optimális stratégia csupán az ellenfél hibájának köszönhetően vezet győzelemre.

Az alábbiakban sokszor csak magát a játékot (szabályait) ismertetjük, ilyenkor a feladat mindig a (lehetőleg) nyerő algoritmus kigondolása, és esetleg programozása. Ha csak demonstrálásra gondolunk, azt mindig jelezzük, és közvetlenül utána ki is tűzzük az aktív változatot.

A felsorolt játékok döntő többsége kapcsolódik valamilyen kereskedelmi forgalomban kapható táblához (készlethez), így ezek kereskedelmi neveit is említjük, és szabályaik, további változataik bővebben megtalálhatók a mellékelt leírásokban.

11.1

(Tic-tac-toe, tic-tac-ought) Képzeljünk el egy 3x3-as négyzetrácsot mint játéktáblát. Két játékos teszi felváltva a saját jelét a táblára. Az nyer, akinek sikerül három (saját) jelet egy vonalba elhelyezni – akár vízszintesen, akár függőlegesen, akár átlósan. Írjunk programot, mely adminisztrálja két gyerek játékát, persze figyelmeztet a szabálytalanságokra is!

11.2

Készítsük el az előző játéknak azt a változatát, amikor ellenfélként játszhatunk is a géppel. Igyekezzünk olyan programot írni, amely minél nehezebben győzhető le! Nem túl nehéz a tökéletesen játszó változat sem!

11.3

Módosítsuk az előző játékot úgy, hogy a játékosoknak csak 3-3 saját jele van, és ha ezek elfogytak, tologathatnak szomszédos mezőre, de átlósan nem szabad! Győz, akinek sikerül három jelét egyvonalban elhelyezni.

11.4

3x3-as táblán játsza két személy, kétféle (0 vagy 1) jel bármelyikét írhatják a mezőkbe (nincs saját jel). Egyikük (előre meg kell állapodni, hogy melyikük) nyer, ha van két egyforma sor vagy oszlop, vagy ha 3 darab 0 került egy vonalba. Ha kilenc jel elhelyezése után egyik fenti feltétel sem teljesül, másikuk a győztes.

11.5 

5x5-ös rácsra ketten felváltva tesznek kétféle jelet, egyikük mindig csak egyet, a másik viszont kettőt tehet. A cél öt saját jelet egy vonalba (átlósan nem számít) gyűjteni. Ha ez senkinek sem sikerül, az eredmény döntetlen.

11.6 

(Egyszerűsített amőba) A játéktér legyen egy 20x20-as rács, ennek mezőire felváltva tesz két játékos jeleket. Az nyer, aki négy saját szimbólumot tud egy vonalba, egymás mellé helyezni. Próbáljuk rögtön a (jól) játszó változatot megírni!

11.7 

(Amőba játék, vagy GOMOKO) Az előző feladatot nehezítjük: öt jelet kell egy vonalba egymás mellé gyűjteni. Nagyon gyakorlott játékosok (és nagyon jó programok) esetén a 20x20-as korlátozás feloldható. Sajnos a bevezetőben tett megjegyzés itt is aktuális: csak részleges stratégiával kísérletezzünk!

11.8 

Módosítsuk az előző feladatot úgy, hogy átlósan elhelyezett ötösök ne számítsanak!

11.9 

(Csőamőba) Képzeljünk el tíz (12-15-20) függőleges üvegcsövet egymás mellett! Ketten felváltva potyogtatnak kétféle színű golyókat a csövekbe. Az nyer, aki négy (esetleg öt) egyforma színűt tud elhelyezni egymás mellé, egymás alá, vagy ferdén egymás mellé.

11.10 

6x6-os rácson játssza két személy, különböző korongjaikkal. Felváltva tesznek, és az nyer, akinek sikerül egy tetszőleges méretű négyzetnek mind a négy csúcsát elfoglalni. Hány ilyen négyzet van egyáltalán?

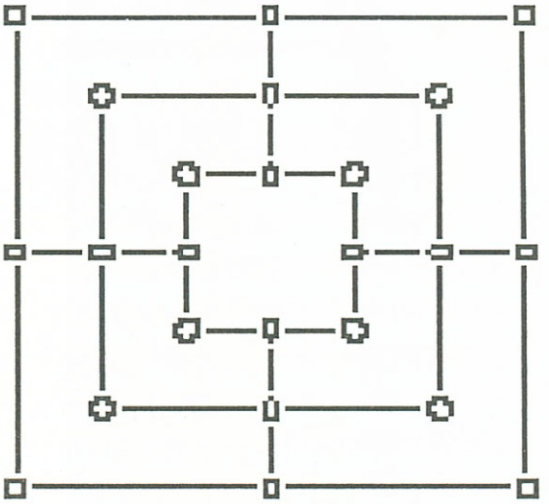
11.11 

Próbálkozzunk meg a malomjáték aktív változatával!

11.12 

(Térbeli malom) Egy 4x4-es rács minden mezőjén áll egy rúd, melyek mindegyikére pontosan 4 korong fér. A játékosok kétféle színű korongot tesznek felváltva, és a cél négy korong egy vonalba gyűjtése. Adminisztráljuk a játékot, figyeljünk a szabálytalan lépésekre! Itt a térbeli megjelenítés nem könnyű!

11.13



(Malom) Az alábbi speciális rácson két játékos felváltva helyezheti el kétféle színű korongjait, összesen 9-9 darabot. Ha valamelyiküknek sikerül hármat egyvonalban olyan mezőkre tenni, amelyek vonallal össze vannak kötve, akkor ő „malmot" csinált, és egyet levehet az ellenfél korongjai közül. Amikor mind a 9-9 korongot letették, attól fogva felváltva tologathatják saját figuráikat egy szomszédos üres mezőre. Ha valakinek már csak három korongja maradt, tetszőlegesen ugrálhat üres mezőre. Cél: elfogyasztani az ellenfél figuráit. Először csak a táblát szerkesszük meg, és programunk figyelje a szabálytalanságokat!

11.14

(LOGI, térbeli amőba) Egy 4x4x4-es térbeli rács pontjaiba tehet felváltva két játékos kétféle színű korongokat. A cél négy egyforma korongok elhelyezése valamilyen irányban. Szimuláljunk egy ilyen intelligens játékszert: adminisztrálja két személy játékát, sőt jelezze a szabálytalan lépéseket is!

11.15

Próbáljuk az előző két feladat aktív változatát elkészíteni: írjunk minél jobban játszó programot!

11.16

Ketten felváltva lépegetnek egyetlen bábuval egy sakktábla bal alsó sarkából kiindulva. Mindenkor egyet mozdulhatnak jobbra, fel, vagy átlósan. Az nyer, aki a közös bábut a jobb felső sarokba betolja.

11.17

(Amazon) Ketten játsszák sakktáblán. Királynőket (vezéreket) tesznek felváltva a táblára, de mindig úgy, hogy az új vezér egyetlen régivel se legyen ütésben. Aki már nem talál helyet, az veszít.

11.18

Ketten játsszák egy olyan óralapon, ahol a tizenkettes helyén 0 van. Kétféle korongjaikat felváltva teszik a még szabad számokra. Minden lépésnél egy korong tetszőleges (szabad) helyre kerül, egy másik korongot pedig arra a számra kell tenni, amelyet az előző szám és a mutató által jelzett szám összegeként kapunk (ha ez nagyobb 11-nél, akkor 12-t ki kell belőle vonni). Ha itt saját korongja

van, újabbat nem tehet, de ha ellenfélé, akkor azt levéve, elfoglalhatja a számot. Minden második (összegző) lépésnél az összegre forgatjuk a mutatót. Az nyer, akinek több pontot (számot) sikerül elfoglalnia: több korongja van a számlapon. Keressünk algoritmust a minél jobb játék szolgálatában!

11.19

Módosítsuk az előző játékot! A második letevéseket az utolsó foglalás és a mutató állásának különbsége határozza meg, természetesen negatív különbségeknél hozzá kell adni tizenegyet.

11.20

(Awari) Egymással szemben, két sorban kétszer hat gödör van, mindkét végükön egy-egy nagyobbal (tál). Minden gödörben 3-3 kavics van. A saját oldalunk valamelyik gödréből felvesszük a kavicsokat, és az óramutató járásával ellenkező irányba haladva egyesével átpakoljuk (a tálba is, ha sorra kerül). Ha éppen tálba kerül az utolsó, jutalmul még egy ürítésre kapunk lehetőséget. Ha olyan gödörbe kerül az utolsó, amelyik üres, akkor a vele szemben lévő gödröt saját táljába ürítheti. Az nyer, akinek a táljába több kavics gyűlik.

11.21

Ismert az előző játék azon változata, amikor kiinduláskor 4-4 (6-6) kavics van a gödrökben. Egy teljesen általános változat kedvéért ezek a bemenő adatok legyenek a játék elején megadhatók, vagy akár véletlenszerűek. Milyen mértékben változik a nyerő stratégia?

11.22

(Wari) A 11.20 játék újabb változata, de nincs tálka, csak 2x6 gödör. Két további ponton is különbözik az awaritól. Egyrészt, ha az átpakolás során 11-nél több követ veszünk fel, akkor a lerakás során a tizenkettedik lépésnél is ki kell hagyni azt a gödröt, ahonnan felvettük a köveket (ez persze nem fordul elő túl gyakran). Másrészt rabolni (zsákmányolni) is másképpen lehet: ha a lerakást az ellenfél oldalán fejeztük be, olyan gödörben, ahol két vagy három kő volt, akkor ezeket a köveket megszerezzük. Az nyer, aki többet zsákmányol.

11.23

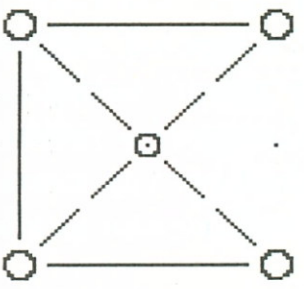
(Oua) Az előző játéktól csak a rablásban különbözik. Ha a letevés az ellenfél utolsó gödrénél ér véget, és ott két vagy három kő volt, az elvehető. Ha egy kirablott gödör előtt (ellenséges gödörben) is 2-3 kő volt, az is elvehető. Ha egy következőben nem 2-3 van, az ezt megelőzők már semmiképpen sem rabolhatók.

11.24

(Mankola) Gödrönként 6-6 kővel indul az előzőkhöz hasonló játék. A játékosok felváltva üríthetik (tál itt sincs!) a saját oldalukon lévő gödröket, és minden ürítés után zsákmányolják az összes olyan gödör tartalmát, ahol sok páros számú kő található.

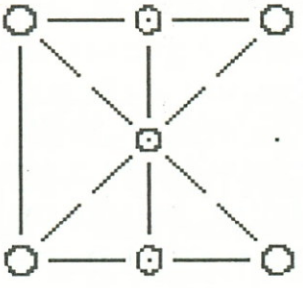
11.25

Két játékos 2-2 korongját felváltva lerakhatja, majd felváltva tologathatja a vonalak mentén. Az győz, aki az ellenfelét úgy körül tudja venni, hogy az semerre se tudjon mozdulni. Készítsünk aktív programot a játékra!



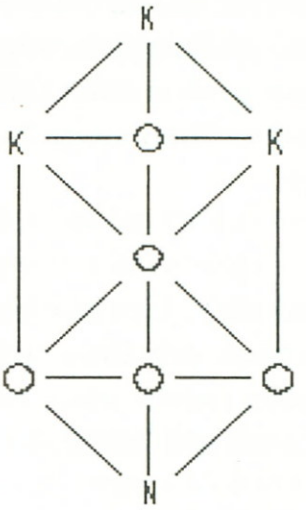
11.26

Az előző játék 3-3 figurával és változatlan szabályokkal ilyen táblán is játszható.



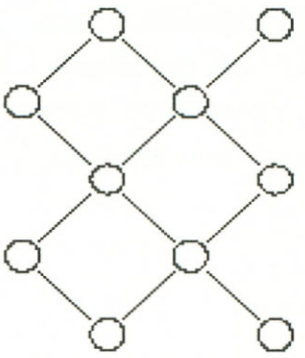
11.27

Készítsünk programot az alábbi játékra: A kutyák (K) csak lefelé lépkedhetnek (ferdén is), a nyúl (N) minden irányba. A játékosok felváltva lépnek. A kutyák győznek, ha körülfogják a nyulat, míg a nyuszi célja a kutyák fölé kerülni.



11.28

A legalsó, illetve legfelső két mezőn áll két-két különböző figura (antilop ellen sakál). Felváltva lépnek egy-egy bábuval bármely egyenes mentén akármennyit. Kétféle figura nem kerülhet egy egyenesre; aki emiatt nem tud lépni, az kimarad. El kell foglalni az ellenfél kiindulását.

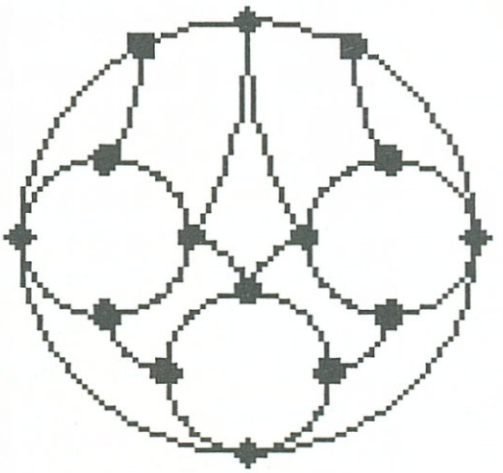


11.29

Sakktáblán játszható az előző feladatok következő változata: a hajtók (vadászok, kopók, ragadozók) a tábla alsó sorának négy egyszínű mezejéről indulnak, és csak felfelé haladhatnak. A menekülők (nyuszi vagy más állat) a túlsó sor egyik azonos színű mezőjéről indul. A lépések mindig ferdén történnek (mindig ugyanolyan színű mezőre), és a cél az előzőkhöz hasonlóan bekerítés, illetve mögékerülés. Írjunk programot, amivel ilyet játszhatunk!

11.30

A mellékelt táblán ketten játszanak. Egyikük legalul, másikuk legfelül indul, s utóbbinak el kell fognia a másikat: oda kell lépnie, ahol az áll. Csak szomszédos (összekötött) pontra lehet lépni. Ha hat lépésen belül nem sikerült az elfogás, a menekülő nyert.



11.31

(Fordítós) Van 10 négyzetünk, egyik oldaluk fehér, másik fekete. Kiinduláskénti sorba vannak rakva, véletlenszerű színekkel felfelé. Két játékos felváltva fordíthat négyzeteket: egyet fehérről feketére, vagy ha akar még egy továbbit: a feketére fordítottól balra lévőt (ez akármilyen színű lehet). Az nyer, aki az utolsó fehéret is befeketíti.

11.32

(Othello, reversi, színfordítós) Ketten játsszák egy NxN méretű rácson, felváltva helyezik le kétféle színű korongjaikat. Minden letevést úgy kell végrehajtani, hogy az újonnan és egy régebben letett saját korong között (legalább egy irányban) egyvonalban lévő mezők mindegyikén ellenséges korong legyen. A lépés által sajátra változtatunk minden olyan szomszédos ellenséges korongot, amely az utoljára letett és egy régebbi saját között van (bármilyen irányban). Az nyer, akinek több mezőt sikerül elfoglalnia. A kiinduló állásban a tábla közepén 2-2 korong áll, az azonos színűek átlósan szomszédosak. Némi ismerkedés után kiderül, hogy a korongok változását nem is könnyű adminisztrálni (időnként nyolc irányt is kell figyelni!), így érdemes megírni a passzív változatot: két személy játékát a képernyőn kövesse a program, jelezze persze a szabálytalan lépéseket!

11.33

Némi játszadozás után megpróbálkozhatunk az előző játék aktív változatával is.

11.34

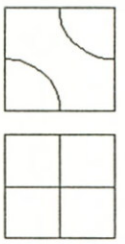
Ismeretes az előző játéknak az a változata is, amikor csak egyetlen (általunk választott) irányban változtathatjuk meg az ellenséges korongokat. Ez látszólag egyszerűbb, de sajnos a választási lehetőség bizony bonyolítja a helyzetet.

11.35

Egy 5x5-ös táblára 12-12 bábut helyezünk, a középső mező kimarad. A játékosok felváltva tolják saját bábujukat szomszédos (átlós érintkezés nem számít szomszédnak!) üres mezőre. Aki nem tud lépni, az veszít.

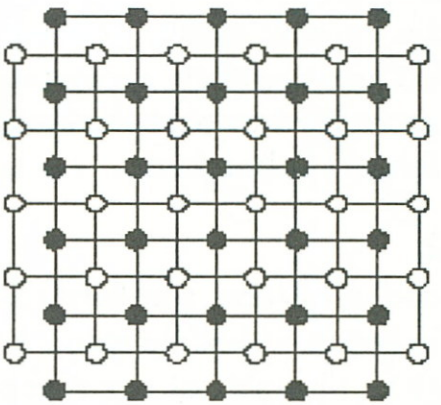
11.36

NxN-es négyzetrácson játszható, valamelyik sarokból az átelleneset kell elérni a következőképpen: kétféle lapocskákat tesz a két játékos a mezőkre, de mindig kapcsolódni kell a lapocskán lévő vonalak valamelyikével egy már letett lap valamelyik vonalához. Aki eléri a túlsó sarkot, az nyer. Eleinte N= 4, 5, 6 esetekkel kísérletezzünk!



11.37

Két játékos (Sötét és Világos) felváltva húz be függőleges és vízszintes szakaszokat a saját szomszédos pontjaik közé. A behúzott vonalak sehol sem keresztezhetik egymást. Az nyer, akinek előbb sikerül összekötni a rács két szemközti oldalát. A rács mérete a gyakorlattal növelhető.



11.38

9x9-es négyzetrácson játssza két személy. Felváltva jelölnek meg négyzeteket, de mindig csak olyat, amelyiknek még nincs szomszédja megjelölve (átlós „szomszéd" nem szomszéd). Ha elfogytak a még megjelölhető üres mezők, a maradt üresekbe azt a számot írják, ahány jelölt szomszédja van (1, 2, 3 vagy 4) lehet. Előzetesen megállapodtak egy-egy különböző számban 1 és 4 között, és az nyer, aki a saját számából többet lát a rácsban. Nem érdektelen a passzív változat sem, de próbálkozzunk az aktív (játszó) programmal is!

11.39

(Fókusz) Egy sakktábla minden sarkából 3-3 (L alakban lévő) mezőt elhagyunk. Két játékosnak van 18 fehér, illetve 18 fekete korongja a tábla közepén, 6x6-os elrendezésben: soronként nézve két-két egyforma egymás mellett, oszloponként pedig egyenként váltogatják a színüket. A játékosok felváltva lépnek, átlósan nem, de a korongok egymásra tehetők. Minden lépésnél megfogjuk egy olyan halom tetejét (állhat egyetlen korongból is), aminek legtetején saját korongunk van, és egyetlen irányba annyit lép, ahány korongot felvettünk, majd ezeket letesszük az érkezés helyén lévő korongok tetejébe. Sehol nem lehet több öt korongnál, ha mégis így alakul, alulról annyit kell levenni, hogy csak

öt maradjon. A levett ellenséges korongok kiesnek a játékból, a sajátokat bármikor bárhová fel lehet tenni, de ez is lépésnek számít: utána a másik játékos következik.

11.40

(Csipke) 20x20 négyzetből álló táblán játsszák. A játékosok felváltva írnak be jeleket, de csak olyan mezőre, amelynek a szomszédjában nincs csillag (átlósan nem számít a szomszédság). Ha már nem tudunk tenni, összeszámoljuk, hány összefüggő részre bomlott a tábla – itt figyelembe vesszük az átlós szomszédságot. Szerepcserével játszanak még egyet, és az nyer (a két játék alapján), aki több részre szaggatta a táblát.

11.41

Kerítsünk körül egy 15x15-ös négyzetrácsot. Két játékos felváltva húzza be valamelyik kis négyzet egy oldalát. Ha valaki egy kis négyzet negyedik (utolsó) oldalát rajzolja, akkor jutalmul még egy oldalt behúzhat (ha ez is negyedik oldal, akkor újabb jutalmat kap stb.). A bezárt négyzetekbe mindenki saját betűjét írja, és a végén az nyer, akinek több betűje van a táblán.

11.42

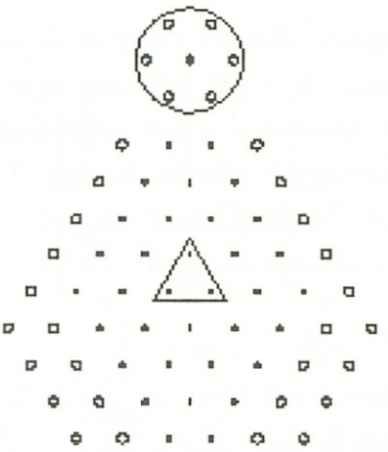
„Felezzünk" el egy sakktáblát átlósan, és tekintsük az átlót is tartalmazó részt. Így egy 36 mezőből álló táblánk van, amihez kell még 18-18 korong. Mindkét játékoshoz hozzárendeljük a tábla egy-egy egyenes (8 mezős) oldalát, ahonnan kezdve felváltva rakosgatják le korongjaikat az ellenfél oldalával párhuzamosan haladva. Vagy a kiinduló vonalra lehet tenni, vagy már előzőleg megkezdett vonalat folytathatunk. Ha ellenséges figura áll az utunkban, át kell ugrani, és ezért hibapont jár. Akinek kevesebb hibapontja gyűlik össze, az nyer.

11.43

Sokféle dámajáték ismert, az „50 táblás játék" című könyv legalább tízfélét felsorol. Egy NxN-es sakktábla alsó, illetve felső 2-3 sorában, azonos színű mezőkön állnak a két játékos kétféle színű korongjai. A korongokkal csak előre lehet mozogni, egyesével, átlósan; az azonos színű mezőkön maradva. Ha a korong útjában ellenséges korong áll, azt átugorhatja, aminek következtében az átugrottat azonnal le kell venni. Az ilyen ugrások folytathatók: ha ugrás után a táblára érési pont előtt újabb ellenség van, az is átugorható ugyanebben a lépésben stb. Az ugrások között (90 fokkal) irányt is lehet váltani, de mindig csak előre lehet mozogni. Próbáljuk elkészíteni mindkét változatot!

11.44

(ROTARY) Az itt látható hatszögletű táblán játsszák, 13 -13 golyóval, az ottani kezdőállással, és fontos szerepe van a hatlyukú korongnak (pörgettyű) is. Minden lépés abból áll, hogy a játékos leteheti a pörgettyűt a tábla olyan részére, ahol a korong lyukaiban több saját golyót láthat, mint ellenségest. A pörgettyű közepe nyílván üres lyuk fölé kell kerüljön. Ezután a pörgettyű tetszőleges (60-120-...-300 fokos) elforgatásával tetszés szerint átrendezheti (átforgathatja) ezeket a golyókat. Az nyer, aki a tábla közepén lévő háromszög mindhárom helyére saját golyót juttat. Igazi játék hiányában hasznos lehet a tábla szimulálása a képernyőn, és bizony nem is könnyű. Először próbálkozzunk a passzív változattal!



11.45

Gondolkozzunk egy kicsit a ROTARY játék nyerő stratégiáján, és próbáljunk minél jobban játszó algoritmust készíteni!

11.46

(Hex) Szabályos hatszögek méhsejtszerűen kapcsolódva rombuszszerűen alkotnak 10x11-es rácsot (11 hatszög egymás mellé, majd 11 újabb egymás mellett, de előzőhöz képest annyival eltolva jobbra, hogy illeszkedjenek). Két játékos felváltva rakja le saját jelét, és céljuk, hogy a rombusz ellentétes oldalait saját jeleikkel összefüggően összekössék. Akinek ez sikerül, az a győztes.

11.47

Jóval nehezebb a feladat, ha 10x10 avagy 11x11-es táblán próbálkozunk az előző játékkal!

11.48

Egy ilyen gyűjteményből nyilván nem hiányozhat a sakk. Ugyanakkor senkit nem akarunk arra buzdítani, hogy előképzettség nélkül sakkozó program írásába fogjon. Már az is komoly teljesítmény, ha valaki egy igazi sakktáblát szimulál: tetszőleges lépés végrehajtódik a képernyőn.

11.49

(Hexapawn, hat gyalog) Csak igen egyszerű ujjgyakorlat egy 3x3 mezős „sakktáblára". A két szélső sorban 3-3 gyalog áll, melyek egymás felé mozoghatnak a sakk szabályainak megfelelően, az ütés is úgy történik. Az győz, aki egy figuráját az ellenfél kezdővonalára viszi, vagy az ellenfél minden figuráját leüti.

Győzhetünk úgy is, ha az ellenfél már nem tud lépni. Ennek a játéknak az aktív változata ízelítőt adhat egy sakkprogram bonyolultságából.

11.50

Próbáljunk olyan programot írni, ami kevés (3-6) figurás végjátékot elemezget! Vizsgálja végig az összes lehetséges lépést, hagyja figyelmen kívül a rosszakat (ütésveszély, sakk stb.), és válasszon minél jobban a többi közül!

11.51

Véletlenszerűen elhelyezve 6-8-20 GO követ egy táblára (a képernyőn), jó gyakorlás fejben kiértékelni a létrejött állást. Próbáljuk a kövek elhelyezését nem teljesen függetlenül, inkább „ügyesen" csinálni!

11.2 Egyéb logikai játékok azonos szereplőkkel

A címből csak részben tudjuk meg, hogy mi kapcsolja össze az alfejezet játékait. Majdnem mindegyik befejeződik véges számú lépés után, van egy győztes és egy vesztes, döntetlen, egy-két kivételtől eltekintve, nem alakulhat ki. A játék kimenetele nem csupán a játékosok ügyességétől függ, egyikük (sok esetben a kezdő) győzni is tud, függetlenül attól, hogy ellenfele hogyan lép. Legtöbbjüknél nemcsak azt bizonyították be, hogy az egyik játékosnak nyerő stratégiája van, hanem maga a nyerő stratégia is ismert, sőt több is igaz, van valami közös ezekben a stratégiákban. Nem akarjuk előre elárulni, hogy mi lenne az, mert a feladatok megoldásának fontos része, és egyben a legérdekesebb is, a nyerő stratégia megkeresése. Ha valaki elég sokat megoldott már közülük, rájöhet arra, hogy mi a közös bennük. Ezután már könnyebben birkózhat meg a többivel. Sok segítséget találhatunk az ajánlott könyvekben, folyóiratokban.

A fejezet csoportosítását több szempont szerint végeztük, elkerülhetetlen volt az ütközés. Ezért találkozunk itt matematikai játékokkal is, a táblás játékok között pedig vannak olyanok, amelyek szorosan kapcsolódnak ehhez az alfejezethez.

11.52

Két játékos, A és B a SZÁMLÉTRA játékot játssza. Legyen A a kezdő, 1-től 10-ig bármilyen egész számot mondhat, B hozzáad valamennyit, majd A ad hozzá, és így folytatódik a játék. A hozzáadott szám legalább 1 és legfeljebb 10. Az nyer, aki a 100-et kimondja. Melyik játékosnak van nyerő stratégiája? Írjunk programot, amely jól játszik mindkét játékos szerepében! Mi választhassuk meg, hogy ki kezd!

11.53 

Hogyan módosul az előző program, ha A és B a SZÁMLÉTRA játékot visszafelé játssza úgy, hogy 100-tól indulnak, és felváltva kivonnak egy-egy számot az előző eredményből? A szám amelyet kivonnak legalább 1 és legfeljebb 10. A győztes az, aki eléri a 0-át.

11.54 

Módosítsuk a SZÁMLÉTRA játékot úgy, hogy a „nyeri" szót cseréljük „veszíti el"-re, a többi szabály változatlan marad!

11.55 

Készítsünk olyan programot, amely eldönti, hogy melyik játékosnak van nyerő stratégiája, ha 100 helyett tetszőleges 10-nél nagyobb pozitív egész számot választunk!

11.56 

Legyen a hozzáadott szám legalább 1 és legfeljebb K, amelyre teljesül, hogy K pozitív egész szám és K<N! Döntsük el, most ki nyer megfelelő játék esetén! Adjuk meg a nyerő stratégia algoritmusát!

11.57 

Próbáljunk olyan programot írni, amelyik jól játssza a SZÁMLÉTRA játékot az előző feladatokban leírt feltételeknek eleget tevő, tetszőleges K és N estén is!

11.58 

Tovább nehezítjük a játékot, ha a hozzáadott (vagy elvett) szám alsó határát is változtathatjuk. Legyen ez L, ahol L<K pozitív egész szám. Írjunk olyan programot, amely megkérdezi L, K és N értékét, ha a feltételeknek nem felel meg valamelyik, azt jelzi. Választhassunk, hogy ki kezd! Jól játsszon mindkét játékos szerepében.

11.59 

Két játékos 100 db forintossal a következő játékot játssza: a kezdő lerak egy egyforintost, és azután felváltva tesznek erre legalább 1-et, de legfeljebb annyit, mint amennyit addig leraktak. Aki a századik forintot odateszi, az nyer. Melyik játékos győz, ha ügyesen játszik? Sorsolás döntse el, hogy ki kezd! Készítsünk olyan programot, amelyik mindig győz, ha az lehetséges!

11.60 

Hogyan módosul a program, ha a kezdő 2, 3 vagy 4 pénzérmét tehet le először?

11.61 

Írjunk programot, amely 100 db helyett egy tetszőlegesen választott pozitív egész N darabszám esetén eldönti, hogy az első vagy a második játékos győz-e, ha ügyesen játszik! Adjuk meg a nyerő stratégia algoritmusát! Írjuk meg a játék általános esetben is jól játszó változatát!

11.62 

Úgy is módosíthatjuk a szabályokat, hogy a játékosok legalább 1-et, de legfeljebb 2-szer (vagy 3-, 4-, ...n-szer) annyit tehetnek, mint amennyit addig összesen kiraktak.

11.63 

András és Béla úgy játszanak, hogy megállnak egymástól pontosan 100 lépésre, majd felváltva egymás felé mennek. Egy alkalommal legalább 2, de legfeljebb 9 lépést tesznek meg. Az nyer, aki a másik helyére tud lépni, vagy átlép azon. Melyik játékosnak van nyerő stratégiája? Hogyan lépjen, ha nyerni akar?

11.64 

András és Béla úgy játszanak, hogy megállnak egymástól 100 lépésre, majd felváltva egymás felé mennek a következő szabály szerint: Mindig szabad 1-et „gyorsítani" vagy „lassítani", vagy ugyanolyan sebességgel haladni, mint az előző saját lépéskor, de megállni nem szabad. Az nyer, aki a másik helyére tud lépni vagy átlépi őt. Írjunk programot, amely segít Andrásnak, hogy megnyerje a játékot! Ellenfele minden lépése után kiírja, hogy mennyit lépjen.

11.65 

Anna és Balázs a következő játékot játsszák: felváltva mondanak egy-egy egész számot 1-től 5-ig, és azt hozzáadják az addig elhangzott számok összegéhez, azzal a megszorítással, hogy az utoljára elhangzott számot nem ismételheti meg a soron következő játékos. Az a vesztes, akinél először lesz nagyobb a szám 39-nél. Olyan programot készítsünk, amely győzni tud, ha az lehetséges! Mi szeretnénk eldönteni, hogy ki kezd.

11.66 

Próbáljunk programot írni az általános esetre! 39 helyett vegyünk egy tetszőleges pozitív egész számot!

11.67 

Az asztalon van 25 szál gyufa. Két játékos felváltva elvesz belőle 1, 2 vagy 3 szálat. Az nyer, aki utoljára tudott venni a halomból. Készítsünk jól játszó programot!

11.68 

Egy kupacban 25 kavics van. A és B a következő játékot játsszák: felváltva vesznek a kupacból 1, 2 vagy 3 kavicsot, egészen addig, amíg a kupac el nem fogy. Az nyer, aki utoljára vesz el egyszerre két kavicsot.

11.69 

Keressünk optimális stratégiát mindkét játékos számára, ha az előző feladatban szereplő halomban N db kavics van! Írjunk programot, amely jól játszik mindkét játékos szerepében!

11.70 

Adott két kupac tárgy (kavics, gyufa stb.). Két játékos felváltva vesz el belőlük, de mindig csak az egyik halomhoz lehet nyúlni (onnan akármennyi elvehető, de legalább egy). Az veszít, aki már nem tud venni. Írjunk erre is tökéletesen játszó programot!

11.71 

Cseréljük ki a „veszít" szót „nyer"-re! Hogyan változik a programunk?

11.72 

Próbáljuk általánosítani az előző feladatot tetszőleges számú halomra! Ezt a változatot nevezik (klasszikus) NIM játéknak.

11.73 

Megengedhető, hogy bárki a már elvett dolgaiból akárhányat visszategyen egyetlen halomba. Hogyan módosul az így is jól játszó program?

11.74 

(Tak-tix) Négyzet alakban korongokat helyezünk el. A játékosok felváltva húznak, bármelyik sorból vagy oszlopból, csak az a kikötés, hogy összefüggő csoportot kell elvenni, a levett korongok között nem lehet hézag. Az veszít, aki az utolsót elveszi. Készítsünk programot, amely jól játssza ezt a játékot!

11.75 

A sakktáblán fekete és fehér korongokat helyezünk el tetszőlegesen minden oszlopban, a feketéket az egyik, a fehéreket a másik oldalon. A játékosok felváltva lépnek egymással szemben. Saját koronggal előre lehet lépni akárhány lépést az üres helyeken. Nem szabad átugrani az ellenfél korongját. Így, ha két korong egymással szembekerül, akkor egyik sem mozdulhat. Az nyer, aki utoljára tud lépni. Próbáljunk jól játszó programot írni!

11.76 

Az előbbi játék másik változatában megengedik a visszafelé lépést is. Hogyan módosul a programunk?

11.77 

(Csien Szü Dzü) Két játékos felváltva vesz el két halomból néhány kavicsot. Ha a játékos csak az egyik halomból vesz ki kavicsot, annyit vehet, amennyit akar, akár az egészet is. Ha mindkét halomból vesz, akkor mindkettőből ugyanannyit kell kivennie. A játékot az nyeri, aki az utolsó kavicsot felveszi. Készítsünk algoritmust a kezdő számára „vesztes helyzetek" megkeresésére!

11.78 

Írjunk programot, amely az előző játékot úgy játssza, hogy mindig győz, ha az lehetséges!

11.79 

A NIM játékot változtassuk meg úgy, hogy ne lehessen akármennyit elvenni egy halomból, hanem csak M és N közötti mennyiséget!

11.80 

Egy jobbra tetszőlegesen meghosszabbítható sáv mezőkre van osztva. Elhelyezünk a mezőkön néhány követ. Két játékos felváltva lép úgy, hogy egy követ közelebb visz a sáv baloldali végéhez. Egyszerre léphetünk többet is, de elfoglalt mezőre lépni, vagy követ átugrani nem lehet. Az győz, aki az utolsó követ a helyére húzza. Próbáljunk jól játszó programot írni!

11.81 

Két játékos felváltva korongokat helyez egy téglalap alakú játéktáblára. A korongok semmilyen mértékben nem kerülhetnek egymásra. Az veszít, aki nem tud többet tenni: nincs már korongnyi hely. Melyiküknek van nyerő stratégiája? Írjunk programot, amely optimálisan játszik akár az egyik, akár a másik játékos szerepében!

11.82 

Játszhatjuk az előző játékot korongok helyett dominókkal is.

11.83 

Készítsünk programot, amely partnerként játsza a következő játékot: ketten felváltva letépik egy margaréta egy vagy két szirmát. Az nyer, akinek az utolsó szirom jut.

11.84 

(Batnum, harapás) Korongokat helyezünk el a játéktáblán téglalap alakban, lehet négyzet alakban is. A játékosok felváltva lépnek úgy, hogy kijelölnek egy korongot. Ez a korong, továbbá minden alatta levő és minden tőle jobbra levő korong eltűnik, mintha csokoládéból haraptak volna ki egy darabot. Az veszít, akinek a bal felső sarokban levő korong jut. Készítsünk olyan programot, amelyben a számítógép nem aktív résztvevő, csak adminisztrálja a játékosok lépéseit! Megjeleníti a játék állását, a hibás lépéseket nem engedi meg, figyeli a játék végét, kiírja a végeredményt.

11.85 

Egyszerűsítsük az előző feladatot úgy, hogy a korongok elrendezése csak 2xn-es legyen! Írjunk erre az esetre aktívan játszó programot!

11.86 

Változtassuk meg a programot úgy, hogy akkor is jól játsszon, ha a korongokat négyzet alakban helyezzük el!

11.87 

(Osztójáték) Ketten a következő játékot játsszák: felváltva 1000 (pozitív) osztóit mondják, azzal a feltétellel, hogy olyan osztót nem mondhatnak, amelyik egy korábban mondott osztónak is osztója. 500 után nem mondhatnak többet pl. 2-t, 25-t, de 200-at igen. Az veszít, aki így magát az 1000-et kénytelen kimondani. Melyik játékosnak van nyerő stratégiája?

11.88 

Az előző játékban 1000 helyett válasszunk egy olyan számot, amelynek prímtényezős felbontása $p*q^n$! Készítsünk optimálisan játszó programot!

11.89 

Hogyan módosul a program, ha a választott szám p^n*q^n alakú?

11.90 

Próbálkozzunk aktívan játszó program készítésével akkor, ha a választott szám prímtényezős felbontása p^m*q^n alakú!

11.91 

Tetszőlegesen választott N pozitív egész szám esetén jól játszó programot írni már nagyon nehéz feladat, de akik elég sokat foglalkoztak játékelmélettel is, megbirkózhatnak vele.

11.92

(Aliquot) Adott természetes számból két játékos felváltva kivonja a szám egy osztóját (a számot magát kivéve). Az ellenfél a megmaradt számból von ki egy osztót. Aki nem tud már kivonni, vesztett. Adjuk meg a nyerő stratégia algoritmusát! Írjunk programot, amely mindig győz, hacsak lehet!

11.93

Ketten a következő játékot játsszák. Az alábbi rendszerbe

$$a_1x_1 + \ldots + a_nx_n = c$$
$$b_1x_1 + \ldots + b_nx_n = d$$

felváltva egymás után az a_i, b_i, x_i (i=1, 2..., n) és c, d betűk helyébe tetszés szerinti valós számot írnak. Minden lépésben tetszőlegesen döntik el, hogy melyik még szabad betű értékét adják meg. 0-t azonban csak akkor írhatnak egy betű helyébe, ha van olyan sor, amelyikben ez a betű szerepel, és rajta kívül ebben a sorban már minden más betű értéke meg van választva. A kezdő azt szeretné elérni, hogy mind a két egyenlőség érvényes legyen, különben a második nyer. Írjunk tökéletesen játszó programot!

11.94

Adott egy téglalap. Oldalainak hossza pozitív egész szám. A játékosok felváltva levágnak belőle egy vagy több négyzetet. Ha az ellenfél már nem tud miből vágni, elvesztette a játékot. Készítsünk programot, amely mindkét játékos szerepében a lehető legjobban játszik!

11.95

Adott egy halomban 6 db kavics. Az első játékos a kavicsokat két halomba osztja úgy, hogy ne legyen egyenlő a halmokban a kavicsok száma. A második ugyanúgy szétosztja az egyiket a halmok közül, majd ismét az első következik. Ha valamelyik nem tud szétosztani a szabálynak megfelelően, vesztett.

11.96

Az előző feladathoz próbáljunk jól játszó programot készíteni, ha a kavicsok száma N!

11.97

20 egymás utáni számozott mezőből álló táblán játssza két játékos. A tábla első 18 mezőjén véletlenszerűen köveket helyeznek el. Minden helyről sorsolással döntik el, hogy ott legyen kő vagy ne. Az utolsó kettő üresen marad. A játékosok felváltva lépnek. András kezd. Minden alkalommal előre vihet egy követ az útjában legelső üres helyre. Béla mindig hátrább tehet egy mezővel egy olyan követ, amelyik mögött üres mező van. András győz, ha a 20. Mezőre követ tesz. Béla győz, ha egy kőelrendeződés megismétlődik.

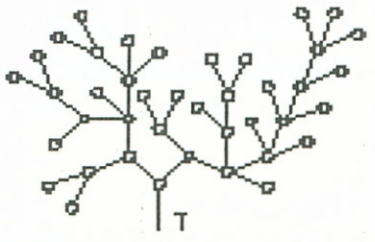
11.98

Játszhatjuk a játékot más hosszúságú táblán is. Készítsünk programot, amely eldönti, hogy nyerhet-e András az adott kőelrendezés esetén! Írjuk meg a program aktív változatát!

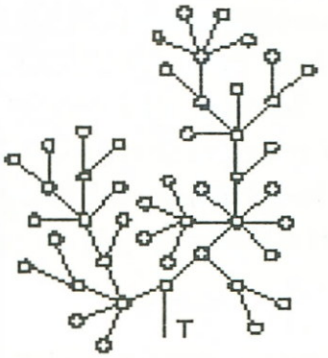
11.99

Írjunk programot, amellyel a következő játékot játszhatjuk! Az ábrán látható „fából" ketten felváltva vesznek el egy-egy tetszőleges „ágat". Az így „levágott ágnak" mindig leesik a folytatása, minden további elágazása is. A megmaradó rész mindig az, amely a T törzset tartalmazza. Aki ezt a törzset veszi el az veszít.

1.

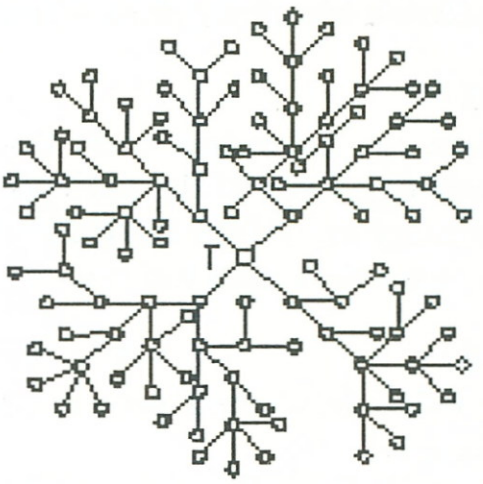


2.



11.100

Két játékos felváltva vesz el az ábrán látható gráf élei közül egyet-egyet. Ha így az ábra két össze nem függő részre bomlik, akkor mindig leveszik ezek közül azt is, amelyik nem tartalmazza a T pontot. Az nyer, aki az utolsó élt veszi el. Készítsünk olyan programot, amely a játékot optimálisan játssza!



11.101

Adott a síkon n ($n \geq 3$) pont úgy, hogy semelyik három ne legyen egy egyenesen. Két játékos felváltva összeköt két-két pontot egy szakasszal úgy, hogy azok ne messék egymást. Az nyer, aki utolsóként tud szakaszt húzni. Próbáljunk jól játszó programot írni!

11.102 

A Shannon-féle kapcsolójátékot egy olyan gráfon játsszák, amelynek két csúcsa ki van tüntetve. Két játékos (NYIT és ZÁR) lép felváltva. NYIT minden lépésben elhagy egy élt a gráfból, ZÁR pedig minden lépésben elfoglal egy élt, amit NYIT a továbbiakban már nem törölhet. A játék akkor végződik ZÁR győzelmével, ha meg tud őrizni egy olyan utat, amely összeköti a két kitüntetett csúcsot, ha ez nem sikerül, akkor NYIT nyer. Próbálkozzunk jól játszó program írásával!

11.103 

Adott hat pont, egy szabályos hatszög hat csúcsa. Két játékos játszik, mindegyiknek más színű ceruzája van. Felváltva kötnek össze egy-egy pontpárt. Aki először kénytelen úgy húzni, hogy saját színéből háromszög jöjjön létre, az veszít. Csak azok a háromszögek számítanak, amelyeknek minden csúcsa a hat pont közül kerül ki. Írjunk mindkét játékos szerepében jól játszó programot!

11.3 Amikor a játékosok feladata eltér

Lehet, hogy az Olvasó indokolatlannak találja a „JÁTÉKOK" első három alfejezetének szétválasztását. Mi úgy gondoltuk, célszerű ez az elkülönítés. Ebben a részben a legtöbb játék (lehet, hogy a fejtörő pontosabb elnevezés) olyan lesz, ami kifejezetten számítógépesnek mondható: lényegében a gépek elterjedésével lettek népszerűek. Most is található némelyik játéknál „játéktér", de ez többnyire csak egy darab papír. A játékosok szerepe csupán a bonyolult változatokban szimmetrikus: általában a feladatsorok végén. Egyszerűbb esetekben egyikük ELREJT vagy GONDOL – az ő szerepe általában (eleinte) könnyebb. A másiknak kell ügyes kérdésekkel minél gyorsabban meg-(vagy ki)TALÁLni. Ezen kívül néhány egyszemélyes játékot is ide soroltunk.

11.104 

A gép GONDOL egy (véletlen) számot 1 és 100 között és a játékosnak (ellenfélnek) ki kell találnia azt. Minden tippre a gép (a program) megmondja, hogy a gondolt szám kisebb vagy nagyobb a tippnél, esetleg egyenlő vele. Ezt a passzív programot kell megírni, majd lehet vele játszani néhányat. Próbáljuk meg minél kevesebb próbálkozással megtalálni a gondolt számot!

11.105 

Kicsit nehezebb az az aktív változat, amikor a gép TALÁLgat, és mi válaszolunk.

11.106 

Némi játszadozás után talán sikerül észrevenni, hogy megfelelő kérdezősködés esetén –a szerencsétől eltekintve – létezik egy minimális kérdésszám, ahány kérdéssel bármilyen számot kitalálhatunk. Próbálkozzunk meg azzal a változattal: amikor programunk minimális kérdésből is KITALÁLja az ellenfél által gondolt számot!

11.107 

Feltételezve, hogy ellenfelünk ismeri a fenti optimális kérdezősködést, írjunk most olyan változatot, amelyik nem hagy semmit a véletlenre: nem GONDOL véletlenszerűen, hanem „menekül" a kérdések elől. Nagyon kell vigyázni, nehogy ellentmondásba keveredjünk előző válaszainkkal. Ily módon most már versenyeztethet is programunk különböző személyeket: szerencse kizárva.

11.108 

Feltételezve, hogy ellenfelünk korrektül követi a menekülős stratégiát, programozzuk be a következő mutatványt: a játék elején találomra „kiírunk" egy számot az adott határok között, majd ügyes TALÁLGATÁSSAL „rákényszerítjük" az ellenfelünket az előre kiválasztott számra.

11.109 

Tegyük fel, hogy pontosan ismerjük a partner menekülős stratégiáját! Milyen számokat tudunk előre kiválasztani, hogy azután MINIMÁLIS számú TALÁLGATÁSSAL ki is kényszerítsük az ellenfélből? Van-e olyan szám, ami eleve kudarcra ítélt?

11.110 

Visszatérve az eredeti (véletlen) számkitalálóshoz, egy más típusú nehezítés, ha a GONDOLT számunk valóban menekül: minden találgatás után egy adott módon megváltozik. Például minden alkalommal egy hozzáadódik; vagy minden „kisebb" válasznál egy hozzáadódik, „nagyobb" válasznál eggyel csökken; lehet ugyanez fordítva is stb. Az első három feladat mintájára ehhez is elkészíthető néhány változat.

11.111 

EGYIK gondol egy számot 1 és 30 között, amit MÁSIKnak kell kitalálni. Ha MÁSIK tippje nagyobb a gondolt számnál, akkor EGYIK azonnal megnöveli számát a különbség kétszeresével, és attól kezdve arra gondol. Ha a tipp kisebb a számnál, akkor csak a különbséggel növel. MÁSIK akkor nyer, ha sikerül kitalálni a (növekvő) számot, mielőtt az elérné a százat. Szimuláljuk géppel EGYIK szerepét!

11.112 

Készítsük el az előző feladat aktív változatát: találgasson minél hatékonyabban a program!

11.113 

Az eddigiek alapján próbáljuk megállapítani, mely számokat nehéz, illetve melyeket könnyű kitalálni!

11.114 

Egy egyszerű (egydimenziós) tűz-víz játékot kapunk a következőképpen: a tippre nem „kisebb-nagyobb" a válasz, hanem „forró" – ha szomszédos a tipp; meleg – ha közel vagyunk; illetve „langyos", „hideg", „jeges" egyre távolabbat jelentenek. Írjuk meg a GONDOL-TALÁLGAT változatot egyaránt!

11.115 

Az előző feladatban lehet 1-7 (vagy 1-9) csillaggal jelezhetjük, mekkora a különbség a tipp és a gondolt szám között. Minél több csillagot kapunk, annál közelebb vagyunk.

11.116 

Újabb változat, ha két számmal kérdezünk, és ilyenkor a lehetséges válaszok: „kisebb", „közte van", „nagyobb". Próbáljunk az alfejezet első hat feladata mintájára ilyen feltételek mellett is minél több változatot végiggondolni!

11.117 

A képernyőn megjelenő rács egyik mezőjében valami kincs van elrejtve. A kurzor (vagy más) billentyűkkel mozoghatunk a rácson, és minden „lövésünk" után válaszul egy számot kapunk: annyi lépésnyire vagyunk a kincstől, ami a függőleges és vízszintes irányú eltérések összegét jelenti. Írjuk meg a DUG változatot: amikor a program csak adminisztrál (válaszol a tippjeinkre)!

11.118 

Próbáljuk meg kiküszöbölni itt is a véletlen szerepét: ne véletlenszerűen dugjon a program, hanem próbáljon menekülni a tippek elől! Ha elkészült ez a változat, játsszunk vele néhányat, és próbáljuk minél kevesebb próbálgatással megtalálni a kincset!

11.119 

Cseréljünk szerepet! Írjuk meg a fentiek KERESŐ változatát! Programunk figyeljen az esetleges ellentmondásokra: menekülésünkbe könnyen csúszhat hiba, ettől ne szálljon el a program, hanem figyelmeztessen rá!

11.120

Nehezítés, ha egy NxN-es táblán az N-nél nagyobb „távolságokat" csak > jel jelzi. Hogyan változik ettől a kérdések és válaszok optimális stratégiája? Próbáljuk a vegyes változatot elkészíteni!

11.121

Az eddigi négyzetrácson a távolság helyett iránnyal is jelezhető, merre van a kincs. Először négy irányt feltételezve, próbáljuk elkészíteni a két programváltozatot egyszerre!

11.122

Nem sokkal nehezebb, ha négy helyett nyolc irányt veszünk figyelembe.

11.123

Az eddigiek mintájára próbálkozzunk a menekülős változattal! Programunk igyekezzen minél jobban nehezíteni a megtalálást!

11.124

Nagyobbaknak való változat, ha a tippre a Pitagoraszi távolság jelenik meg. Minden kísérletünkre a (passzív) program mondja meg a kincs és a tipp igazi távolságát!

11.125

Próbálkozzunk meg az aktív (kereső) programmal Pitagoraszi távolságot feltételezve is!

11.126

Jelentős nehezítés a keresőnek, ha a távolság értéke egészre kerekítve jelenik meg.

11.127

Ugyancsak izgalmasabb a játék, ha több kincs is van, ilyenkor a még meg nem találtak távolságai sorban jelentik a választ.

11.128

Az előző feladat további nehezítése, ha a távolságok összekeverednek valamelyik tipptől kezdve.

11.129

A kincs egyenletes sebességgel mozoghat is egy előre meghatározott irányba.

11.130

Újabb távolságfogalom lehet a már említett „forró" – „meleg" – „langyos" stb. válasz is.

11.131

A válasz lehet egy (egység)vektor két koordinátája is, ami a kincs irányába mutat.

11.132

A kereső játékok mindegyike átfogalmazható térbelire, persze a megoldás (és a játék is) jóval nehezebb.

11.133

Tekintsünk N különböző egész számot összekeverve, egy sorozatban! A következő módszerrel kell sorbarendezni: egy elemi lépés során a sorozat baloldaláról egy darabot megfordítunk. Például az „5" művelet az 123456789 sorozatból az 543216789 sorozatot adja. Próbáljunk ilyen műveletek segítségével rendezni tetszőleges sorozatot!

11.134

Próbáljuk az előző feladatban a rendezést minél kevesebb lépésben megoldani! Mennyire függ a lépésszám a rendezendő sorozattól?

11.135

Készítsünk programot, amely 5 betűs szavak kitalálását teszi lehetővé. Mi a program által „gondolt" szóra találgatunk, a program pedig a miénkre. Minden találgatás után megmondjuk a találatok számát.

11.136

(Master mind, gondolkozz) Hatféle színű golyócskából négyet kiválaszt az egyik játékos, a másiknak pedig ki kell találni, melyik helyen milyen szín van. Utóbbi kirak különböző lehetőségeket, amire válaszul megtudja, hányat talált el összesen, és hányat úgy, hogy a pozíciók is egyeznek. (PSKZ és PZFL esetben 2 (PZ) illetve 1 (P) találat.) A cél: minél kevesebb lépésben kitalálni az eldugott golyók színeit, pozíció szerint. Írjunk programot, amely egyszerre GONDOL és TALÁLGAT!

11.137

Valamivel nehezebb, ha több egyforma színt is lehet dugni, mert ilyenkor nő az esetek száma.

11.138 

A számkitalálós és kereső játékokhoz hasonlóan, készítsünk optimálisan kereső (kérdező) programot, amelyik véletlentől függetlenül minimális lépésben találja meg a megoldást!

11.139 

Az előző feladatot figyelembevéve, próbáljuk a szokásos nehezítést! Ne dugjunk előre, hanem a találgató kérdéseitől függően „meneküljünk"!

11.140 

(Csuka Ruma) Sorban egymás mellett gödrök vannak: négy kisebb egyenként 2-2 kaviccsal, és a sor végén egy nagyobb üres ötödik. A kis gödrök valamelyikéből kivesszük a benne levő kavicsokat, és innentől kezdve sorban a nagy felé haladva egyenként átrakjuk a többi gödörbe. Ha marad, teszünk a nagyba is, sőt folytatjuk elölről a kis gödrökkel, amíg el nem fogynak a kivett kövek. Egyetlen megszorítás, hogy üres gödröcskébe nem kerülhet az utolsó kő! Ezután egy másik (nem üres) kis gödröt választunk ... Nyerünk, ha sikerül a fenti szabályok szerint minden kavicsot átpakolni a nagy lyukba, illetve sikertelenek vagyunk, ha valamelyik lépésben az utolsó kavics üres gödörbe jut.

11.141 

(Memory) Csináljunk memóriajátékot! A képernyőn le van fordítva KxM kártya. Két kártyát, amit mi kérünk, felfordít a program. Ha ez pár (két egyforma), akkor megnyertük őket, ellenkező esetben visszafordulnak a helyükre. Ezután újakat kérhetünk. Az győz, aki több párt nyer.

11.142 

(Bűvös lámpácskák) KxN lámpa helyezkedik el rácsszerűen, mindegyik eloltva (kikapcsolva). Mindegyiket fel kell gyújtani az alábbiak szerint: Bármelyiket kijelölve, vele együtt minden szomszédja ellenkező állapotú lesz (megfordul). Általában 4 szomszéd van, de a széleken csak három, a sarokban kettő. Először készítsük el a „táblát" a képernyőre!

11.143 

Érdekesebb a játék, ha a tábla szemközti oldalai „összeérnek" (akár labda-, akár úszógumiszerűen képzelhetjük el), ilyenkor minden lámpának pontosan négy szomszédja lesz. Készítsünk ehhez táblát!

11.144 

Próbáljunk stratégiát keresni a bűvös lámpákra: hogyan célszerű oltogatni a lámpácskákat, hogy minél hamarabb valamennyien átváltozzanak! Érdemes többféle méretű táblát megvizsgálni, hol a síkbeli, hol a „felhajlított szélű" az egyszerűbb (gyorsabb).

11.145 

(Raktár) Egy 4x5-ös keret egyik rövid oldalán középen van egy két egységnyi nyílás. A keretben van négy darab egységnyi lapocska, öt darab dupla lap (ebből négy álló, egy fekvő), és egy 2x2-es nagy lap. Utóbbit kell „kihozni" a keretből egymás melletti tologatással, de közben kisebbet nem szabad kivinni. Több kiinduló állás is elképzelhető. Szimuláljuk a játékot: a képernyőn jelenítsük meg a keretet a lapokkal, gombnyomásra tologasson a gép, és figyeljen a szabálytalanságokra!

11.146 

Az előző játéknak próbáljuk meg az aktív változatát: néhány (2-4) kiinduláshoz generáljuk az optimális lépéssorozatot, aminek segítségével jó néhány esetben képes lesz folytatni a programunk.

11.147 

Egészítsük ki (helyesen) az alábbi nyitott mondatot!
Ebben a mondatban pontosan ... darab 0, ... darab 1, ..., és ... darab 9-es van.

11.148 

Egészítsük ki (betűvel kiírt) számnevekkel az alábbiakat!
Ebben a mondatban pontosan ... darab a, ... darab b, ..., ... darab z betű van.

11.4 Matematikai játékok

Játékos matematikai feladatokat, matematikai fejtörőket gyűjtöttünk egybe. A könnyebbek inkább a matematika megkedveltetésében, az érdeklődés felkeltésében segíthetnek, míg a nehezebbekhez már ajánlatos némi tapasztalat, jártasság. Másként fogalmazva, a matematika eleinte cél, később pedig eszköz. A fejezet bevezetőjében említett könyvek nagy része természetesen ezen alfejezethez is ajánlható.

11.149 

Írjunk programot, amely NxN-es (N páratlan, először legyen 5) bűvös négyzetet generál! 1-től N-ig az összes számot be kell írni egy NxN-es rácsba úgy, hogy minden sorban, minden oszlopban és az átlókban ugyanaz legyen a számok összege.

11.150 

Tekintsük a A1, A2, A3, A4, B1, B2, B3, B4, C1, C2, C3, C4, D1, D2, D3, D4 16 darab szimbólumot! Helyezzük őket el egy 4x4-es rácson úgy, hogy minden sorban, minden oszlopban különböző betűk és különböző számok álljanak!

11.151

Próbáljuk a fentieket NxN-es rácsra általánosítani!

11.152

Van három rúd, és az egyiken 3 (4, 5) különböző nagyságú korong: bármelyik korong alatt csak nála nagyobb van. Úgy kell átpakolni a korongokat egy másik rúdra – a harmadikat is felhasználva –, hogy az említett piramis struktúra minden rúdon minden időben érvényesüljön. A korongokat csak egyesével lehet mozgatni.

11.153

(Hanoi tornyai) Próbáljuk a fenti feladatot tetszőleges (N) számú korongra általánosítani! (Történeti érdekesség, hogy több ázsiai országban ahhoz az eseményhez kötötték a világ végét, amikorra 64 kőkorongot sikerül ezzel a módszerrel „átpakolni". Erre gondolva N ne nagyon legyen 20-nál nagyobb!)

11.154

Az előző feladat további általánosítása, amikor egy már elkezdett átpakolást folytat, és helyesen befejez a gép. Figyelni kell, melyik rúd legyen a cél – nem mindig mindegy! Ennél egyszerűbb valamivel, ha csak azt mondja meg programunk, hány lépés szükséges még.

11.155

Jól ismert az alábbi trükk: néhány kártyára számok vannak írva (pl. 1-100), bizonyos számok esetleg több lapon is szerepelnek. Ha valaki gondol egy számra, és átadja azokat a kártyákat, amelyeken a szám előfordul, mi (a kártyák első számai alapján) azonnal megmondjuk a gondolt számot. Gépesítsük a mutatványt!

11.156

Az előző feladatban kulcsfontosságúak a lapok bizonyos (általában legelső) számai. Próbáljuk ezeket egészen más jellegűekre kicserélni úgy, hogy a mutatvány változatlanul könnyű legyen!

11.157

Van egy $a_1<a_2<a_3...<a_N$ súlysorozatunk. Milyen terheket tudunk ezekkel kétkarú mérlegen mérni, ha

- csak egyik serpenyőbe teszünk súlyokat,
- mindkét oldalon lehetnek súlyok.

Mi van, ha a súlysorozatban egyenlőséget is megengedünk?

11.158

Van 12 golyónk, közülük az egyik más súlyú, mint a többi. Próbáljuk egy kétkarú mérleggel (súlyok nélkül) minimális számú méréssel kideríteni, melyik a hibás golyó?

11.159

Általánosítsuk az előző feladatot N golyóra! N növekedtével hogyan változik a szükséges mérések száma?

11.160

Van tizenöt golyónk, amelyek közül kettő radioaktív. A Geiger-Müller dobozba betéve közülük akárhányat, megtudjuk, van-e azok között aktív (hogy hány van, azt már nem tudhatjuk). Hogyan válasszuk ki a vizsgálandó golyókat, hogy minél hamarabb megtaláljuk mindkét aktívat?

11.161

Egy boltban 1624 Ft értékben 33 és 47 forintos tollak vannak. Vajon melyikből mennyi?

11.162

Egy programmal próbálgassuk végig – minél rendszeresebben és ügyesebben – , milyen helyettesítéssel lesz igaz:

```
ABC  *  BC
-----
BCD
 EBC
------
FABC
```

Az azonos betűk azonos számjegyeket, a különbözők különbözőket jelölnek.

11.163

Az előző feladat klasszikus változatában SEND+MORE=MONEY egyenletet kell megoldani. Ennek 4-5 különböző sebességű változata is létezik, köztük 10-100-szoros sebességkülönbséggel!

11.164

Próbáljunk olyan programot készíteni, amely minél általánosabb körben old meg az előzőhöz hasonló betűszámtan feladatokat!

11.165 

Farey-törteknek nevezzük a közönséges törtek (racionális számok) olyan elrendezését, ahol sorra vesszük az összes 1/N, 2/N, ... (N-1)/N alakú számot, majd növelve N-et folytatjuk mindezt. Írjunk programot, amely nagyság szerint előállítja és kiírja ezeket a törteket!

11.166 

Készítsünk programot, amely 1-999 között tetszőleges számot betűvel kiír – minél intelligensebben! Ugyanígy, ha betűvel írjuk be, akkor számmal kapjuk vissza!

11.167 

Készítsünk öröknaptár programot!

11.168 

Készítsünk programot számok gyakoriságának megtippelésére! Jelenjen meg 100-300 kétjegyű szám 5-20 másodpercre, aminek eltelte után tippelni kell a legnagyobbra, a leggyakrabban előfordulóra, a hiányzókra stb.!

11.169 

Adott N féle tulajdonság, melyek mindegyike egyforma eséllyel jellemzi egy halmaz elemeit. Kiválasztva a halmazból K elemet, mi lesz az esélye annak, hogy van benne két azonos tulajdonságú elem? (A feladat születésnap-paradoxon néven közismert, ha azonos tulajdonságnak a közös születésnapot tekintjük.)

11.170 

A tizenötös játék tetszőleges állását vizsgáljuk meg: végrehajtható-e az átrendezés?

11.171 

Töltsünk ki egy TOTO vagy LOTTO szelvényt, majd szimuláljunk egy húzást, és ellenőrizzük a találatokat!

11.172 

Töltsünk ki néhány LOTTO szelvényt úgy, hogy BIZTOS kettes találatunk legyen: bárhogy húzzák is a számokat, valamelyik szelvényen legyen (legalább egy) kettesünk!

11.173 

Az előző feladatban inkább biztos hármas legyen a cél! Érdemes 90 helyett kevesebb (12-30) számból tippelni. Vajon miért?

11.174

Csináljunk TOTÓ-kulcsot: adott biztos, két-, háromesélyes meccsekhez töltsünk ki úgy szelvényeket, hogy biztosan legyen telitalálatunk!

11.175

Az előző feladatban elégedjünk meg biztos 12, (11) találattal!

11.176

Készítsünk programot, amely 100 (200-1000) lottóhúzást szimulál! Készítsünk statisztikát a kihúzott számok gyakoriságáról! Próbáljuk minél látványosabban ábrázolni!

11.177

Dobjunk fel egy pénzdarabot többször, és készítsünk statisztikát az előforduló fej/írás gyakoriságokról! Figyeljük meg, hogyan alakul: mikor melyik vezet a „versenyben"?

11.178

A fej/írások sorozatában keressünk minél hosszabb egyforma (csupa fejből vagy csupa írásból álló) részsorozatot!

11.179

Ketten játszanak: felváltva dobnak pénzt, és sorban írják a kimeneteleket. A játékhoz mindig csak az utolsó hármat veszik figyelembe, és az első kap pontot, ha FIF adódik, míg a második kap, ha FFF jön ki. Addig játszanak, míg valamelyikük 50 pontot el nem ér. Vajon egyforma-e a két játékos esélye?

11.180

Programozzuk be az ismert feladványt: egy kecskét, egy farkast és egy fej káposztát kell átvinni a folyón! A kecskét semelyik másikkal nem lehet magára hagyni, és a csónakban csak két dolgot vihet a révész. Írjunk programot, amely állandóan figyelemmel kíséri a helyzetet, és figyelmeztet a veszélyre!

11.181

Ha van N tárgyunk (számunk), és ezeket véletlen sorrendben szeretnénk kiíratni, akkor legegyszerűbb N darab véletlen sorszámot kisorsolni, és ezek sorrendjében kiíratni az elemeket. Ennek a vázolt eljárásnak van egy hibája - tapasztalni fogjuk. Próbáljuk ezt kiküszöbölni, kérdés mi lesz annak a hátránya?

11.182

Egy jobb (gyorsabb) keverés, ha generálunk két véletlen egészet, és a második sorszámút az első helyére tesszük, majd megjegyezzük a második sorszámú helyen lévő értéket. Újabb véletlen sorszámot generálva, ide rakjuk az előbb meg-

jegyzett számot, és ami ott volt, azt jegyezzük meg. Kb. hány ilyen cserét kell végrehajtani, hogy „jól" összekeveredjen az eredeti számhalmaz?

11.183

Ha az a feladat, hogy kb. N darabot (véletlenszerűen) kiválasszunk, akkor igen célszerű választani egy N-nél kisebb A véletlen sorszámot, és egy N-nel relatív prím egészet (R). Ezután A-ból kiindulva R-esével végiglépkedünk a számokon, ez lesz a keverés. Ha N-nél nagyobb sorszám adódna, természetesen levonunk belőle N-et, és úgy folytatjuk a lépkedést. Ez az eljárás nagyon gyors, és gyakran „eléggé" véletlen is.

11.184

A két utolsó eljárást ötvözve kaphatjuk az igazi keverést. Kérdés, hogy mit jelent az ötvözés?

12. FEJEZET

GRAFIKA ÉS HANG

GYAKOROLTATÓ GRAFIKA
GEOMETRIAI TRANSZFORMÁCIÓK
FÜGGVÉNYÁBRÁZOLÁS
REKURZÍV ÁBRÁK
TÉRBELI ÁBRÁZOLÁS
HANG

12. GRAFIKA ÉS HANG

Az immár évtizedes kétség – „programot írjunk, avagy használjunk" – feloldását kísérli meg a fejezet. Mindkét fenti célra nagyon alkalmas témakörről van szó:

- pedagógiailag hasznos, motivált;
- látványos és objektív visszajelzés és
- nagyon sokszintű közelítési mód jellemzi.

Ugyanakkor már egy jó évtizede DIGITART néven egy új művészeti ág bontakozik: Dürer, Escher, Vasarely és mások hagyományos nyomdokait követve grafikához, montázsokhoz a művészek segítségül hívják a számítógépet. Ebbe a kategóriába egyaránt beletartozik a videofelvételek manipulálása, érdekes (függvénnyel megadott) görbeseregek kinyomtatása, és irányítottan véletlenszerű rajzolatok válogatása. Minthogy ehhez némi művészi érzék, speciális látásmód is szükséges, célunk nem lehet ilyen alkotások oktatása, de szeretnénk ezt a világot is érzékeltetni.

Véletlen látványnak nevezzük, amikor egy grafikus ábra paraméterei véletlen számok. Legegyszerűbb esetben futtatunk egy ilyen programot, és feljegyezzük, milyen paraméterek mellett „néz jól ki". Haladóbb változat, amikor programunk – pontosan az eddigi tapasztalatok alapján – nem teljesen „összevissza" válogat, hanem igyekszünk „esztétikusan irányítani" a véletlenszám generálását. Ilyen algoritmusok segítségével fontos fogalmak (oszthatóság, közös osztó,) mélyíthetők el. A legfontosabb, hogy felismerjük a látványt „esztétikusan befolyásoló, vezérlő" összefüggéseket az egyes paraméterek között. Mint említettük, ez valóban motivált, szinte játékos bevezetés a problémamegoldás témakörhöz.

Általában a grafikus utasítások a nyelvek géptől leginkább függő részei. Az alábbiakban olyan lehetőséget képzelünk el, ahol egyetlen utasítással lehet vonalat húzni, kört rajzolni. Sok feladatnál (például forgatások) jelentős könnyítés, ha polárkoordinátás megjelenítés is lehetséges; azaz, ha Logo (teknőc) grafika is rendelkezésre áll. A színezés, és a hang még ennél is változatosabb, ezek ügyében először okvetlenül át kell tanulmányozni a gép, illetve a gép kézikönyvét. Nem győzzük eléggé hangsúlyozni a módszeres kísérletezést, mint tanulási módot, ez itt mindennél fontosabb és gyümölcsözőbb!

Megjegyezzük, hogy a mellékelt ábrák nem feltétlenül mérethűek, a fő hangsúly mindig az arányokon, kapcsolatokon van: nalami ilyesmit kell a képernyőre „varázsolni".

Az első fejezet teljesen hagyományos módon (grafikus) programozást oktat, gyakoroltat. A második, harmadik rész inkább a matematikával való rokonságon van a hangsúly, tehát a programok futtatása is hasznos. A negyedik fejezet a programozás egyik legizgalmasabb részét (a rekurziót) támogatja.

Az két utolsó fejezet a teljesség kedvéért kerül be: nem lehet kihagyni. E két terület szélsebesen fejlődik, szinte függetlenül az előzőktől. Jóval kevesebb tapasztalattal, inkább a jövőre kacsingatva említettük.

Javasolt irodalom:

1. Székely V.: Számítógépes grafika alapjai PC-n.
ComputerBooks, 1993
2. Hargitai - Kaszanyiczki.: Grafikák IBM PC-n.
LSI,1993
3. Turcsányiné Szabó M. - D. Senftleben: A Logo programozási nyelv.
Műszaki Könyvkiadó, 1986
4. Votisky Zsuzsa (szerk.) : Etűdök személyi számítógépekre.
Gondolat, 1984
5. Major Z. - Valovits I.: A BASIC feladatok tükrében.
Tankönyvkiadó, 1986
6. DIGITART művészeti kiállítás katalógus.
Szépművészeti Múzeum.
7. Kepes J.: Mikroszámítógépes grafika.
Műszaki Könyvkiadó, 1987

12.1 Gyakoroltató grafika

Már említettük, hogy a képernyőre való rajzolás kiváló gyakorlási lehetőség arra, hogy a tanulók programozási alapfogalmakkal (azonosító, ciklus, elágazás, véletlenszám, tömb stb.) alaposan megismerkedjenek. Ennek az alfejezetnek határozottan ez a célja, tulajdonképpen a könyv második fejezete (Elemi programozás) kiegészítésének is tekinthető.

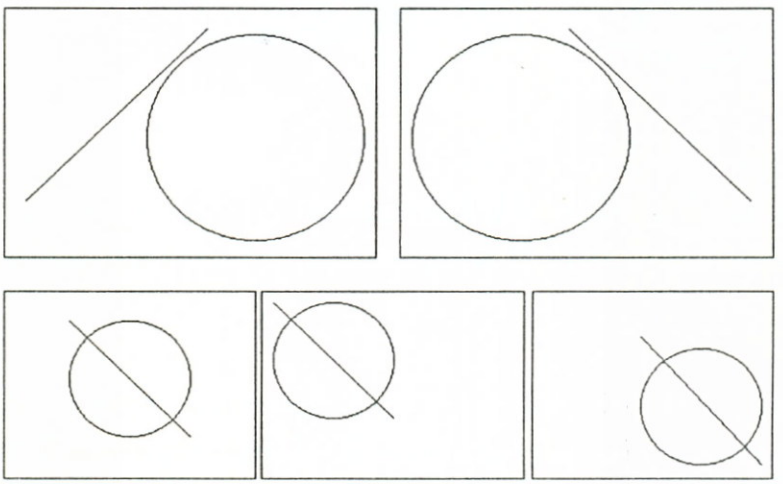
A gyakorlás konkrét célja a koordinátageometria elmélyítése mellett 5–20 soros programocskák írása, amelyekben azonosítók, ciklusok és tömbök (esetleg feltételek) is szerepelnek.

Ismét hangsúlyozzuk, hogy minimális elvárás a géppel szemben, hogy szakaszt, kört és téglalapot rajzoljon egyetlen utasítással (általában LINE, CIRCLE, RECTANGLE, BAR). Ezek persze kiválthatók segédprogramokkal, de ez a megoldás nehézkes és általában lassú. Némi trigonometriával megoldhatók a különböző forgatások is, de mint jeleztük, komoly előny a Logo-grafika, az ellipszis és az ívek lehetősége. Az ilyen igényt mindig jelezzük.

Viszonylag kevés feladat vonatkozik a színekre. Ennek a könyvbeli megjelenítés nehézségén túl az is az oka, hogy nagyon nehéz általános feladatokat megfogalmazni. Rengeteg próbálkozásra van szükség, amire persze (szorgalmi feladatként) buzdítunk is.

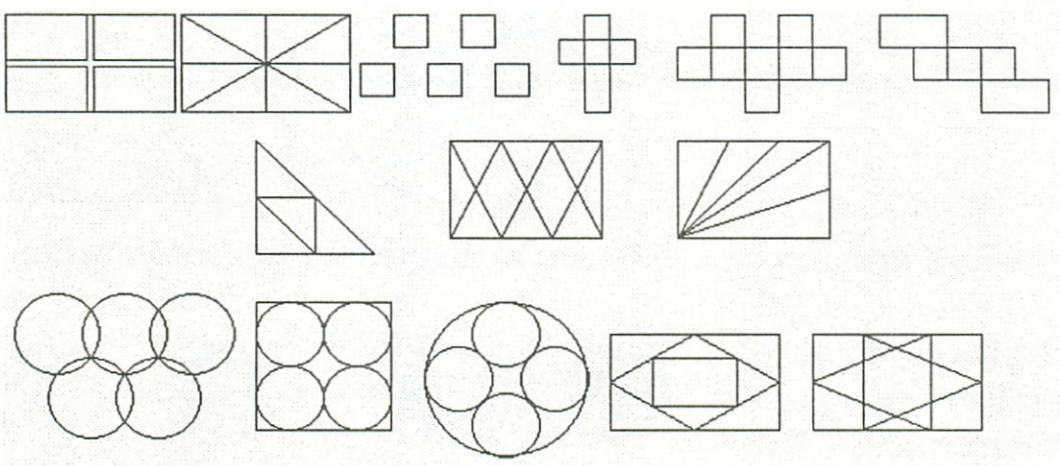
12.1

Készítsünk 1-2 soros programokat, az alábbi ábrák rajzolására!



12.2

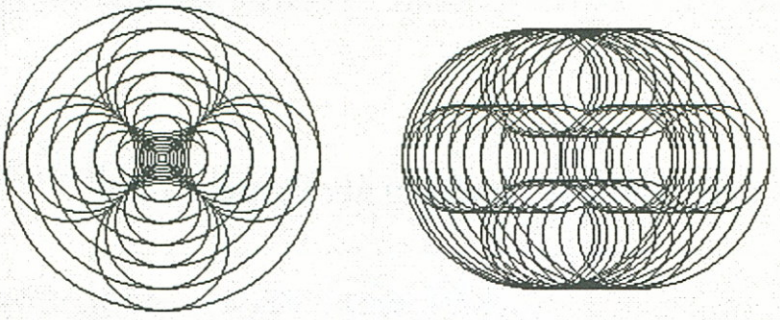
Az alábbi rajzok mindegyikét egy-egy 3-7 soros programocska készítette Alakzatok elhelyezését, a képernyőn való tájékozódást gyakorolhatjuk velük. Írjuk meg a programokat! Az utolsó sor rajzaihoz már gondolkodni is kell egy kicsit, sőt némelyiket érdemes papíron próbálgatni, számolgatni.



12.3

Változókkal lehet „mozgatni" az előző két feladat ábráit. Lehetnek olyan változók, amelyek meghatározzák az ábra helyét (tologatás), méretét, arányait (nagyítás) stb. Változtassunk minél több dolgot! Ezzel is tájékozódást, illetve geo-

metriai összefüggéseket gyakorolhatunk. Ha nem töröljük minden futás után a képernyőt, akkor sokkal érdekesebb ábrák is kialakulhatnak.

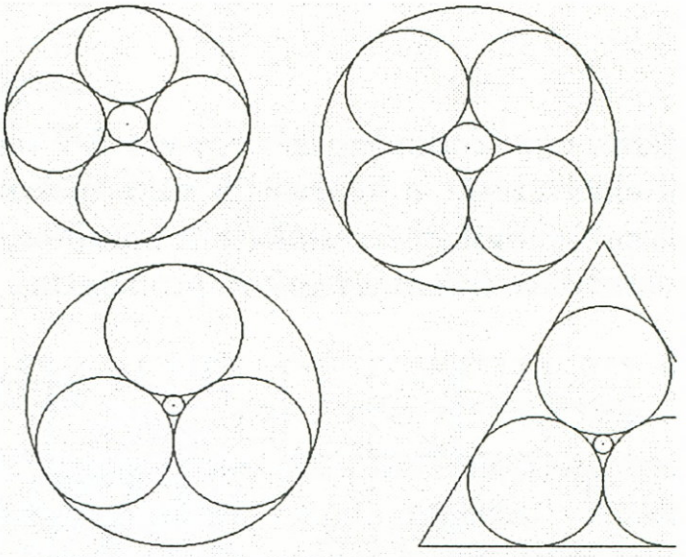


12.4

A 12.1. példa is „kínálja" a változók használatát. Írjunk egyetlen programot, amely minden sor valamennyi ábráját megrajzolja, csak a változó helyes értékét kell beállítani!

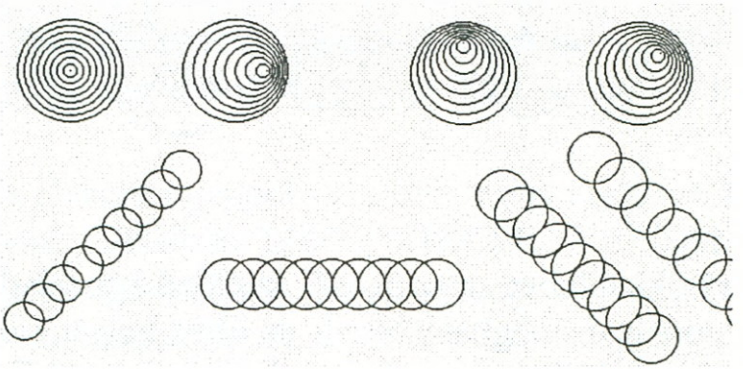
12.5

A változók használatának még magasabb szintje, amikor némi geometriai tudást alkalmazva készítünk tetszőleges ábrákat. Őrizzük meg eközben az érintés fogalmát, sőt próbáljuk az affinitást is megfelelőre állítani.

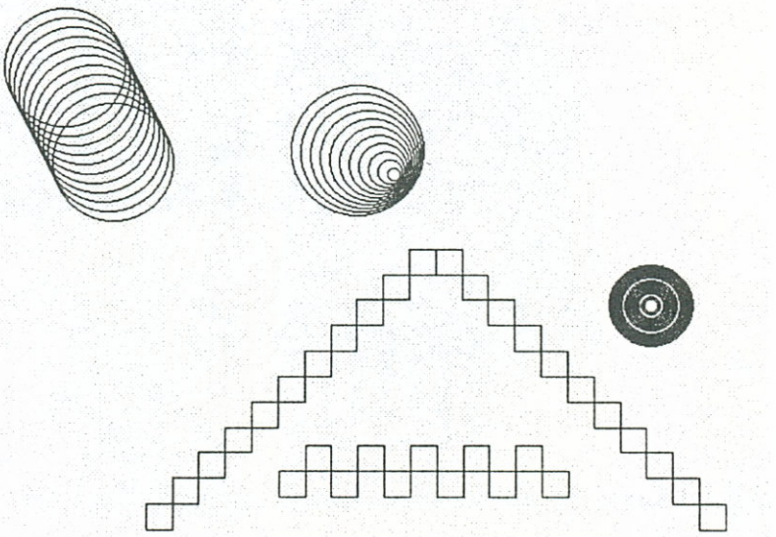


12.6

Ciklusok segítségével sokkal gyorsabban alakíthatunk ki érdekes ábrákat. Ehhez persze az azonosítók is nélkülözhetetlenek.



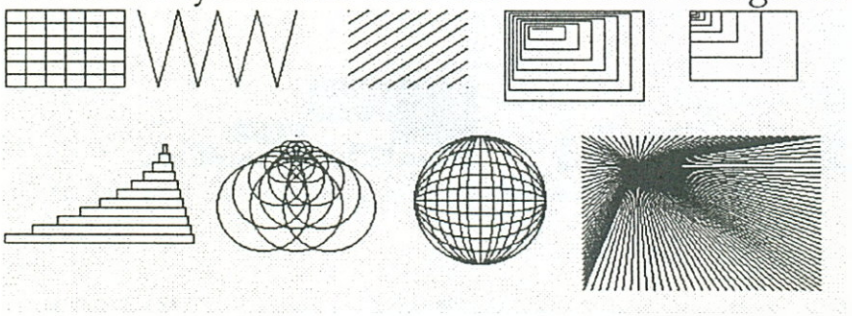
12.7



Készítsük el az alábbi rajzokhoz tartozó programokat! Oldjuk meg változatonként minél kevesebb (3-5) utasítással, azaz okvetlenül használjunk ciklusokat!

12.8

Itt néhány további – picit bonyolultabb – rajz látható. A feladat: a képernyőre „varázsolni" őket! Reménytelennek tűnő esetben most is segíthet a próbálgatás.



12.9

Helyezzünk 10-10 kört és téglalapot véletlenszerűen a képernyőre! Nem lóg ki egyik sem?

12.10

Néhány eddigi feladatból próbáljunk véletlen látványt csinálni: kapjanak RND-vel értéket a változók!

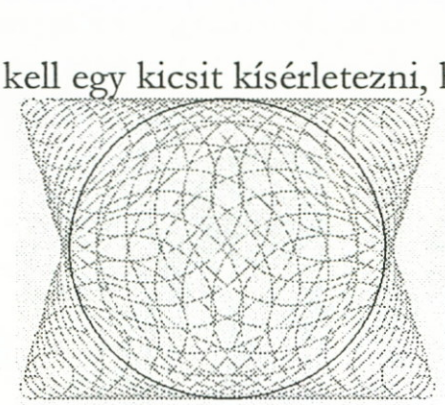
12.11

Készítsünk védelmet az ellen, hogy a véletlen alakzatok „kilógjanak" a képernyőről!

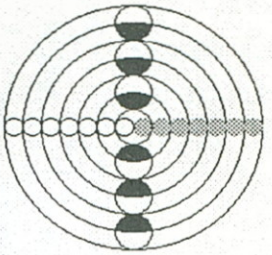
12.12

Húzzunk véletlenszerűen 100-500 vízszintes (függőleges, átlóval párhuzamos stb.) vonalat a képernyőre! Mennyire lesz sötét (világos) a képernyő?

12.13

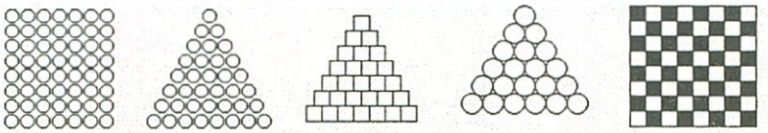


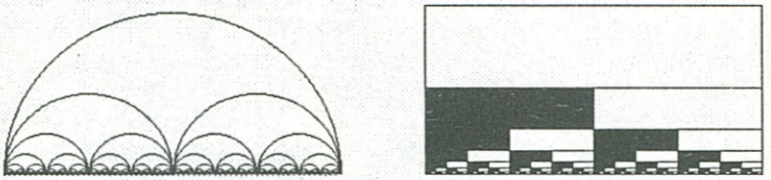
Valószínűleg kell egy kicsit kísérletezni, hogy az alábbi ábrák előálljanak:



12.14

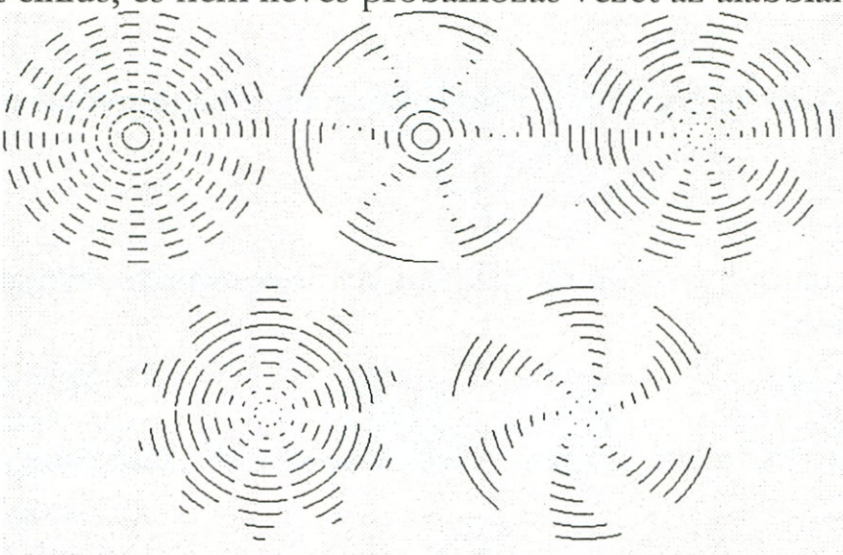
Az alábbi néhány ábrát okvetlenül egymásba ágyazott (skatulyázott) ciklusokkal szerkesszük!





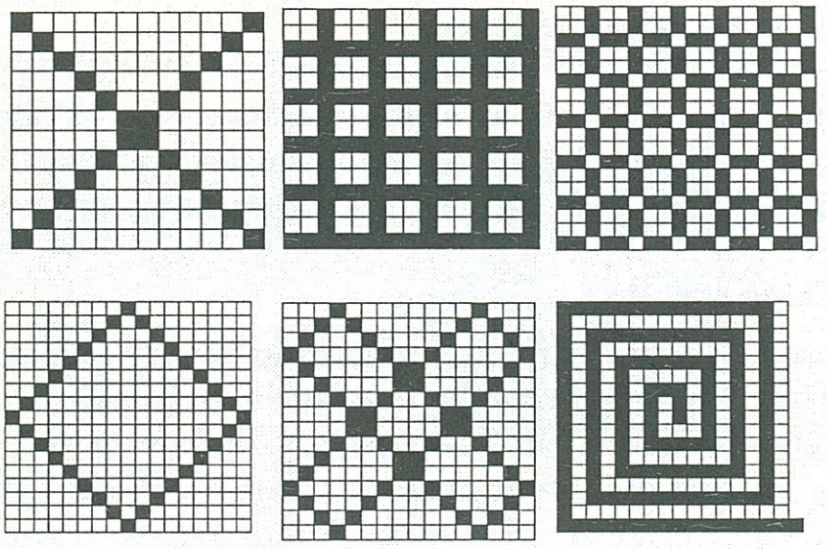
12.15

Ugyancsak két ciklus, és nem kevés próbálkozás vezet az alábbiakhoz:



12.16

Először vegyük észre az alábbi ábrákban a színezés szabályszerűségeit, majd reprodukáljuk azokat:



12.17

Próbáljunk véletlenszerűen színezni az előzőkhöz hasonló ábrákat! Itt figyelni kell, hogy a tartomány külsejét (véletlenül, tévedésből) satírozni hosszadalmas, és nem is látványos!

12.18

Az igazi teljesítmény, ha „véletlenszerűen színezünk szabályosan"! Például egy rácsban minden N-edik mezőt, ahol N véletlenszám; vagy véletlen színeket választunk; egy pont körül véletlen sugár mentén színezünk stb.

12.19

Írjunk programot, melynek segítségével könnyen tudunk egyszerű rajzokat készíteni! Rajzoljon pontonként a gép: a kurzormozgató billentyűk hatására négy irányban hagyjunk nyomot a képernyőn!

12.20

Módosítsuk az előző programot úgy, hogy a kurzor 8 irányba mozoghasson, azaz finomabb rajzokat is készíthessen!

12.21

Írhatunk olyan programot is, hogy egy-egy billentyű segítségével rajzolható legyen szakasz, kör, téglalap!

12.22

Írjunk programot, amely megrajzolja egy egyszerű dal kottáját violinkulcsban!

12.23

Írjunk programot, melynek segítségével a képernyő egy kis területe a számítógéppel megjegyeztethető, és adott (más) helyre kirajzolható!

12.2 Geometriai transzformációk

Az iskolai oktatásban eléggé elkülönül az elemi (szerkesztéses, bizonyításos) geometria és a koordinátageometria. Időben is különállnak és a feladatok, eredmények is jócskán különböznek. Jelen alfejezet elsődleges célja, hogy egy kis hidat építsen e két anyagrész közé. Sokan mások is észrevették már, hogy erre a számítógép grafikus lehetőségei kiválóan alkalmasak.

Az esetleg nem túl vonzó ismeretek intenzív gyakoroltatásának unalmát azzal próbáljuk ellensúlyozni, hogy gyakran generáltatunk látványos rajzokat a képernyőre. Ezeket persze ki-ki tetszése szerint változtathatja, tovább színesítheti, újabb véletlen elemeket szőhet a feladatokba. Az itt gyakorolt ismeretek segítségével az előző alfejezetben készített véletlen látványok messze felülmúlhatók: némi ízelítőt szeretnénk adni a számítógépes (grafikai) művészetből. (Itt jegyezzük meg, hogy M. C. Escher, majd V. Vasarely grafikáira nagy hatással voltak kiváló arab és más keleti geométerek tervezte mozaikok.)

Még egyszer hangsúlyozzuk a kísérletezés szerepét. Az ábrák előállításához gyakran szükségesek bonyolultabb összefüggések. A vizuális próbálkozások jelentősége pontosan az, hogy módszeres próbálgatás révén szinte rávezetnek az összefüggések felismerésére. Ezután már az összefüggés megfogalmazása (az algoritmizálás) is könnyebb.

Az alfejezet első felében az ismert transzformációkat gyűjtöttük össze, a második felében ezek alkalmazásával generálunk érdekesebb látványokat. A többszörös eltolás (elforgatás) itt mindig azt jelenti, hogy az eredeti síkidomot egy távolság (szög) többszöröseivel transzformáljuk sorban egymás után, és így egybevágó alakzatok egy sorozatához jutunk. A gép még egyenes vonalakat is ritkán képes húzni, szinte mindent törött szakaszokkal közelít. Az ebből adódó úgynevezett interferenciajelenséget használjuk ki az utolsó részben: végül egészen mást látunk, mint amit rajzolunk. A sűrűn egymás mellett elhelyezkedő töréspontok az eredetinek szinte ellentmondó rajzolatokat hoznak létre.

12.24

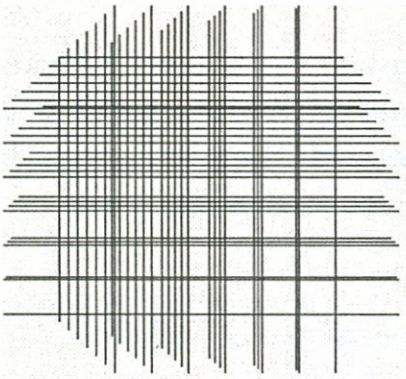
Az előző fejezetben már sikerült alakzatokat vízszintesen és függőlegesen eltolni a képernyőn. Próbáljunk egyszerűbb formákat (pont, kör, szakasz stb.) tetszőleges irányba eltolni!

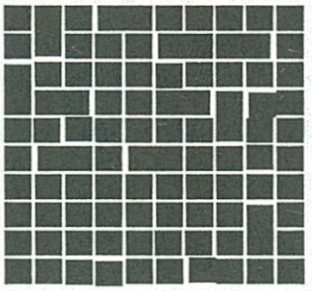
12.25

Próbáljunk bonyolultabb alakzatokat (sokszög, ív, ellipszis stb.) tetszőlegesen eltolni! Hogyan lehet jellemezni azokat a síkidomokat, amelyeket könnyebb, illetve, amelyeket nehezebb eltolni?

12.26

Érdekes lehet, ha egyenes szakaszokat ritkuló vagy sűrűsödő módon sokszor (egyre jobban) eltolunk.





12.27

Még izgalmasabb az előző feladat, ha több (nem feltétlenül merőleges) irányban toljuk el a szakaszokat: a besűrűsödések sokkal több változatosságot kínálnak.

12.28

Az előző feladathoz készítsünk védelmet a képernyőn kívülre írás ellen: előre állapítsuk meg, mi a maximális mértékű eltolás!

12.29

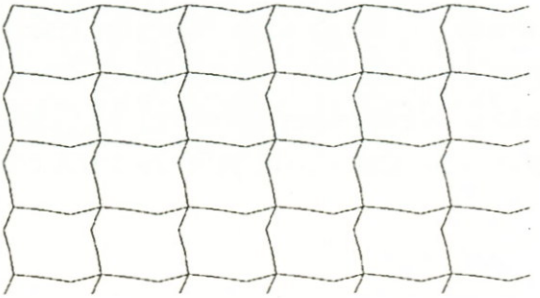
Egy négyzetrács elemeit véletlenszerűen mozgatva egy picit, érdekes „mutációkat" generálhatunk. (Digitart, Budapest 1986. Molnár Vera és Nemes Kálmán ötletei.)

12.30

Próbáljuk meg szabályos sokszögekkel beborítani a képernyőt! A sokszögek hézagmentesen illeszkedjenek, és ne is fedjék át egymást, az ilyen lefedést szokták parkettázásnak is nevezni. Hány oldalú sokszögekkel oldható ez meg? Írjunk programot, amely ilyen szabályos sokszögekkel parkettáz!

12.31

Téglalap jellegű sokszögekkel a következőképpen parkettázhatunk. Képzeletben beborítjuk a képernyőt egy N*M-es hálóval. A háló négy sarkának bármelyikéből kiindulva N*M darab L alakot (esetleg elforgatva) fedezhetünk fel. Ha mindegyik L alakot megfelelő töröttvonallal helyettesítjük, amely átmegy az L végpontjain és töréspontján, akkor egy elég egzotikus alakzattal sikerül parkettáznunk. Vigyázzunk,



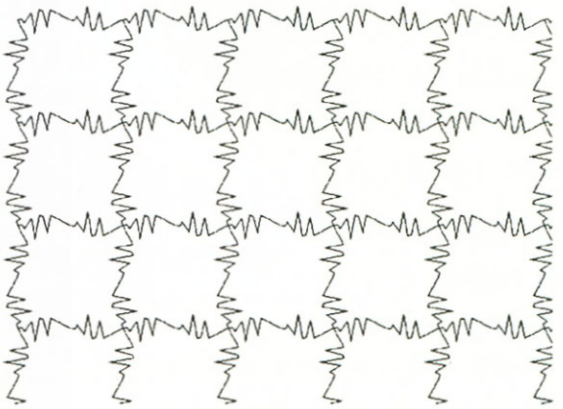
hogy a helyettesítő törött vonal ne nagyon messe önmagát!

12.32

Homorú-domború parketták váltogatják egymást, ha a fenti négyzetrács minden második helyén a mező teljes kerületét (zárt) töröttvonallal helyettesítjük. Itt még inkább kell vigyázni, hogy csak egyszer „záruljon be" a vonal.

12.33

Rengeteget javít a látványon, ha a parkettákban különböző szimmetriák érvényesülnek. Igyekezzünk az oldalakat (töröttvonalakat) tükrözéssel, forgatással egymásból létrehozni!



12.34

Az interferencia jelenségekből ad ízelítőt, ha az előző három feladat bármelyikében eltologatjuk a rácsot. Miután egyszer helyettesítettük a szakaszokat töröttvonallal, ismételjük meg ezt még néhányszor, egyre jobban eltolva a rácsot. 3-4 eltolás után rá se lehet ismerni az eredeti parkettákra! Persze nem mindegy, mennyivel toljuk el!

12.35

Bármelyik parkettázás továbbfejleszthető, ha a rács egymás melletti soraiban (oszlopaiban) egyetlen dolgot megváltoztatunk – véletlenszerűen. Tehát például az első oszlop mezői egyeznek, a második oszlopban is csupa egyforma parketta van, de mindegyik egyetlen szakaszban (pontban) különbözik a balra mellette lévőtől. A harmadik oszlop ismét egyetlen jellemzőben tér el a másodiktól, stb. Érdekes átalakulás jöhet létre az első és az utolsó oszlop között.

12.36

Az előző program futása során több változás fér el, ha valamelyik sarokból indulunk ki, és az átló mentén haladunk. Eleinte egyszerűsítjük a parkettát (elhagyunk pontokat, vonalakat), majd a mellékátlótól kezdve újabbakat veszünk hozzá. Némi szerencsével M.C.Escher „Metamorfózisok" című sorozatára emlékeztető alakzatok jelenhetnek meg a képernyőn.

12.37

Jóval nehezebb, ha nem téglalapok, hanem paralelogrammák alkotta rács segítségével parkettázunk.

12.38

Megpróbálkozhatunk olyan rácsból kiindulni, amelyet (szabályos) háromszögek vagy hatszögek alkotnak. Itt nemcsak számolgatni kell egy keveset, hanem még nagyobb szerepet kap a kísérletezés: hogyan néz ki jól az ábra.

12.39

A képernyő baloldalán lévő tetszőleges alakzatot tükrözzük át a másik felére!

12.40

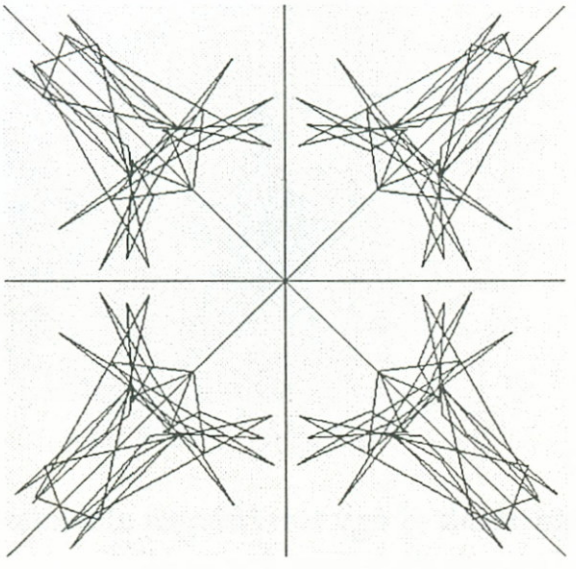
A képernyő valamelyik negyedében lévő idomot tükrözzük a többi három negyedbe is!

12.41

Ha a vízszintes és függőleges tengelyek két felezőjét is tekintetbe vesszük, minden idomnak hét további megfelelőjét tudjuk előállítani. Írjunk ilyen programot!

12.42

Az eddigiek alapján már egy egész látványos kaleidoszkópot tudunk készíteni: véletlenszerűen generált alakzatokat tükrözéssel 2-4-8-szorozva érdekesen fog alakulni a képernyő. Az igazi teljesítmény a véletlen idomok ügyes megválasztása, illetve a transzformáció gyors végrehajtása. Okvetlenül négyzetalakú (vagy „körszerű") „képernyőben" gondolkodjunk!



12.43

Meghúzva a téglalap alakú képernyő két átlóját, minden pontot többszöröznI(!) tudunk az átlókon való többszöri tükrözéssel (mindkét átlóra felváltva tükrözünk). Hány képpont jöhet létre? Mennyire lesz ez kaleidoszkópszerű (azaz szimmetrikusan szabályos)? Minden képpont rajta lesz-e a képernyőn?

12.44

Az előző feladaton okulva csak egy kisebb (belső) téglalap pontjait tükrözzük, így minden képpont a képernyőre kerül! Keressünk „üres" foltokat, és gondolkodjunk el, mi lehet ezek létezésének az oka?

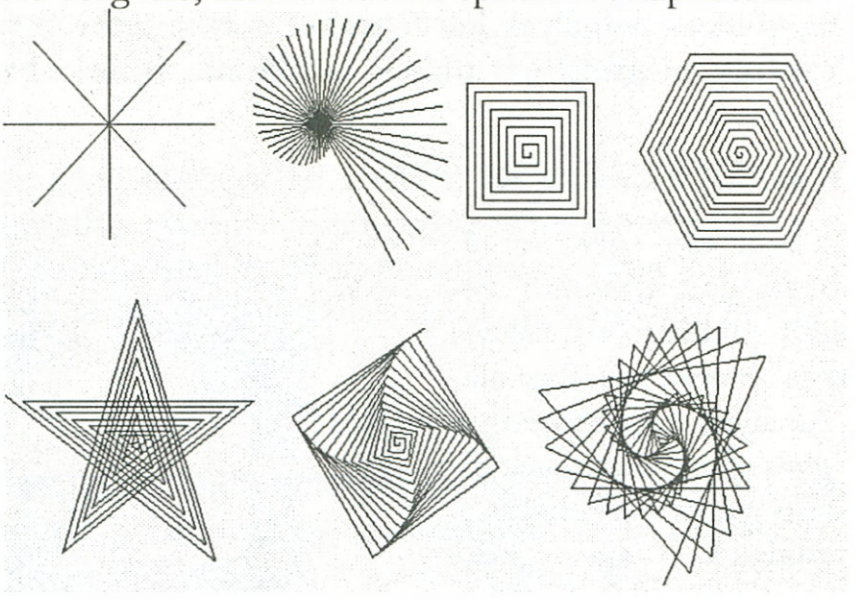
12.45

Tükrözzünk egy alakzatot tetszőleges irányú egyenesre!

A Logo (teknőc) grafika fantasztikus karrierjét elsősorban annak köszönheti, hogy nagyon könnyű megvalósítani a forgatást. A polárkoordináta-rendszer nagyon támogatja ezt a transzformációt: a „fordulj és menj" filozófia nagyon látványos ábrákat eredményes egészen kicsik számára is könnyen megvalósíthatóan. A BASIC és Pascal nyelvekben (amelyek Descartes koordinátákat használnak), ez nem ennyire egyszerű: trigonometrikus függvények szükségesek.

12.46

Néhány egyszerű forgatás, illetve érdekes spirálokat kaphatunk:



12.47

Forgassunk el egy tetszőleges alakzatot egy belső pontja körül! Milyen alakzatokat, hogyan érdemes forgatni?

12.48

Ha külső pontja körül forgatunk el egy idomot, akkor sokkal jobban kell vigyázni, nehogy lelógjon az ábra a képernyőről! Tetszőleges pont és tetszőleges alakzat esetén próbáljuk előre megállapítani, kifér-e vagy sem!

12.49

Szabályos sokszögek előállítása is forgatásra vezethető vissza. Szerkesszünk szabályos N szöget!

12.50

Tegyünk egy kis téglalapot a bal felső sarokba! Úgy toljuk végig a képernyő tetején, hogy közben forgassuk is egy irányba 10-50 fokkal! Utána az alatta lévő sorban egy már kicsit elforgatott téglalapot forgassunk (és toljunk) tovább, és így tovább! A legalsó sorban lesznek végül a legjobban elforgatott idomok. Ha szerencsésen (ügyesen) választottuk meg a forgatások és eltolások mértékét, érdekes (szinte térbeli) ábrákat kaphatunk.

12.51

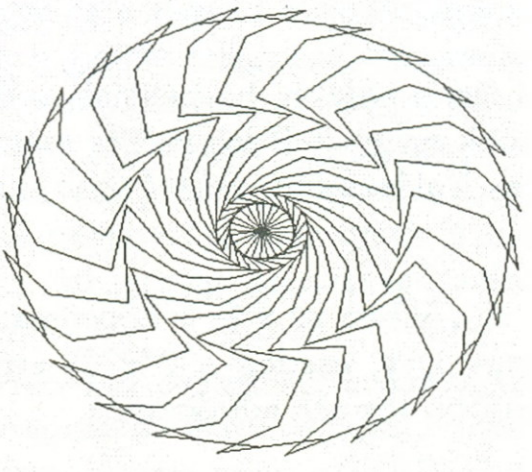
Forgassunk el egy síkidomot kétszeresen, sokszor egymás után! Egyrészt a középpontja, másrészt egy külső pont körül forgassunk (keringessünk), de másmás szöggel! Próbáljuk keskeny alakzatokkal, illetve például négyzettel! Változtassuk a kétféle elfordulás szögeinek arányát!

12.52

Tovább fokozható az előző program hatása, ha mozog az a pont, amely körül forgatunk. Izgalmas játék a kialakult képernyő alapján kitalálni, milyen pályát is járt be a forgatás centruma.

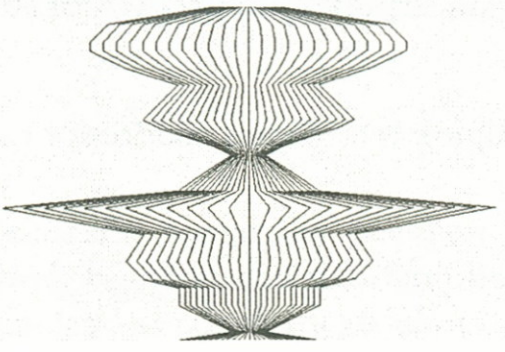
12.53

Rajzoljunk egy töröttvonalat a képernyő közepére, amely nem metszi önmagát! Forgassuk körbe valamelyik végpontja körül!



12.54

Legyen egy töröttvonal a képernyő egyik oldalán. 8-20 lépésben közelítsük a töröttvonal minden pontját a képernyő függőleges szimmetriatengelyéhez, majd tükrözzük az egészet a képernyő függőleges középvonalára (szimmetriatengelyére)! Érdekes, kehelyszerű alakzatok jelenhetnek meg.



12.55

Az előző feladatban a közelítést kétféleképpen csinálhatjuk: Mindig osztjuk a távolságot egy egynél nagyobb számmal, vagy mindig csökkentjük egy alkalmas számmal. Melyik megoldás az élethűbb, és miért?

12.56

Körgyűrűket rajzoltathatunk úgy is, hogy egy NxN-es pontrács minden (x, y) pontját egyenként megvizsgáljuk, vajon páros-e az INT((S*x/N)^2+(S*y/N)^2) érték? Ha igen, akkor kigyújtjuk a pontot, ellenkező esetben nem. Az S paraméter változtatásával hogyan módosul az ábra?

12.57

Az előző feladatban C_1 és C_2 konstansokat hozzáadva S*x/N, illetve S*y/N kifejezésekhez egy eltolás jellegű transzformációt hajtunk végre. Keressünk további összefüggéseket N, S és C_1 illetve C_2 között!

12.58

Az előző feladatokban a négyzetösszeget a párosságán kívül vizsgálhatjuk aszerint is, hogy mondjuk osztható-e hárommal (vagy néggyel, öttel stb.), és eszerint rajzolunk pontokat vagy sem. Ilyenkor az előző ábráknak érdekes „ritkítása" adódik.

12.59

Sokkal látványosabb, ha például az öttel való oszthatóságnál öt esetet különböztetünk meg: az öt lehetséges maradék szerint választunk színeket a pontok számára. Valószínűleg sokkal több szabályosságot fogunk felfedezni!

12.60

Ha az előzőkben az x és y koordinátákra vonatkozó más függvények (például xy, avagy x+y, esetleg x^2*y^2) értékei alapján döntünk, még sokkal meghökkentőbb formák adódhatnak.

12.61

Bizonyos alakzatokat (kör, ellipszis, szabályos sokszögek stb.) nagyon könnyű nagyítani. Írjunk ilyen programokat!

12.62

Próbáljunk tetszőleges síkidomot nagyítani egy tetszőleges pontból!

12.3 Függvényábrázolás

Ez az alfejezet több lépésben jut el az „automatikus normáláshoz", ami a egyáltalán nem egyszerű feladat. A feladatok megoldása után olyan programok birtokába juthatunk, amelyekkel szinte bármely függvény grafikonjának bármely részletét (tetszés szerinti méretben) ábrázolni tudjuk.

A matematikával igen szoros kapcsolat ugyanakkor az egyváltozós valós függvények grafikonjának a derékszögű koordinátarendszerben történő ábrázolása, ami bizonyára segít a függvény fogalmának elmélyítésében. Egy további nehéz témakörhöz is segítséget próbálunk nyújtani: a függvények transzformációjához, ami sok tekintetben rokon a 12.2. résszel. Reményeink szerint itt is gyümölcsöző a koordináta-geometriai közelítéses tárgyalás.

Példát mutatunk paraméteres alakban adott görbék előállítására is; a görbét ilyenkor úgy tekinthetjük, mint egy mozgó pont pályát, azaz egy adott időintervallum minden értékéhez hozzárendeljük valamilyen módon a sík egy-egy pontját. Ezt két, az adott $(t_1;t_2)$ intervallumon értelmezett egyváltozós valós függvény, x és y segítségével tehetjük meg: a mozgó pont a t időpontban a sík (x(t); y(t)) koordinátájú pontjában tartózkodik. (Lásd például a középiskolás matematika könyvekben az egyes paraméteres vektoregyenletével foglalkozó részeket!) Egyes feladatoknál a pálya egyenlete adott, így a paraméterezést magunknak kell elvégeznünk.

Néhány görbét egyszerűbb polárkoordinátás egyenlete alapján megrajzolnunk. A polárkoordináta-rendszerben a sík pontjait egy rögzített félegyenes végpontjától (origó) mért távolsággal (r) és az origóból az adott pontba húzott szakasz és a félegyenes által bezárt (természetesen előjeles) szöggel (s) jellemezhetjük. Így például ha egy pont polárkoordinátái (10;-20), akkor ez azt jelenti, hogy ez a pont a rögzített félegyenes végpontjából induló, azzal negatív (az óramutató járásával megegyező) irányban 20 fokos szöget bezáró, 10 egység hosszúságú szakasz (másik) végpontja.

Teljesen természetes, hogy a fentiekkel párhuzamosan okvetlenül érdemes megismerkedni az olyan (matematikai) segédprogramok használatával, mint a DERIVE, Graphical Calculus, *CAD rendszerek, illetve talán a MAPLE és a MATHEMATICA.

A két módszer (programozás és programhasználat) ideálisan erősíti egymást.

12.63

Ábrázoljuk az [5;10] intervallumon az

$$x \rightarrow x/5+1$$

függvény grafikonját oly módon, hogy az értelmezési tartomány pontjain (a képernyő felbontása által megengedett lépésközzel) végighaladva minden ponthoz számítsuk ki a függvényértéket, majd a ábrázoljuk az ennek megfelelő pon-

tot! Használjunk a képernyő vízszintes és függőleges felosztásánál azonos mértékegységeket, az origót helyezzük a képernyő bal alsó sarkába!

12.64

Ábrázoljuk a [2;3] intervallumon az

$$x \rightarrow 5*x-5$$

függvény grafikonját oly módon, hogy az értelmezési tartomány pontjain (a képernyő felbontása által megengedett lépésközzel) végighaladva minden ponthoz számítsuk ki a függvényértéket, majd ábrázoljuk az ennek megfelelő pontot! Használjunk a képernyő vízszintes és függőleges felosztásánál azonos mértékegységeket, az origót helyezzük a képernyő bal alsó sarkába!

12.65

Az előző feladatokban szereplő két függvény egymás inverze, grafikonjaik (a matematikából ismert összefüggés alapján) a koordinátarendszer y=x egyenletű egyenesére vonatkozóan szimmetrikusan helyezkednek el. Ezzel ellentétben a képernyőn az első esetben egy („folytonos") szakasz, míg a második esetben egy (egymástól „izolált" pontokból álló) pontsor adódik. Adjunk magyarázatot erre! Vizsgáljuk meg, hogy milyen módosítással kapunk a második esetben is szakaszt a képernyőn!

12.66

Ábrázoljuk a [2;5] intervallumon az

$$x \rightarrow 5*x+1$$

függvény grafikonját! Használjunk olyan egységeket a koordinátatengelyeken, amelyekkel a képernyőt vízszintes és függőleges irányban egyaránt kihasználjuk (azaz: igazítsuk a grafikont mindkét irányban a képernyő méretéhez, akár annak árán is, hogy a két tengelyen különböző egységeket kell alkalmaznunk)!

12.67

Ábrázoljuk a [-2;5] intervallumon az

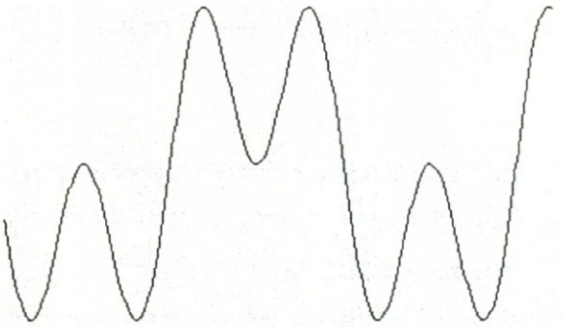
$$x \rightarrow 2*x+1$$

függvény grafikonját! Rajzoljuk meg a koordinátatengelyeket is, és jelöljük rajtuk az egységeket!

12.68

Az f(x) függvény grafikonját az értelmezési tartomány egy [a;b] intervallumán szeretnénk ábrázolni, ezért a képernyőpontok vízszintes koordinátáit meg kell feleltetnünk az [a;b] intervallum pontjainak. Milyen összefüggést kell használnunk ehhez, ha azt akarjuk, hogy az intervallum (vízszintesen) pontosan lefedje a képernyőt? Készítsünk olyan programot, amely az intervallum kezdő- és végpontjának értékétől függően végrehajtja ezt a megfeleltetést!

12.69



Tekintsük az előző feladatban szereplő függvény [a;b] intervallumban felvett értékeit! Azt szeretnénk, ha a grafikon függőleges irányban is alkalmazkodna a képernyő méretéhez (azaz: rá is férne, s azt teljes magasságában kihasználná), ezért a függvény [a;b]-n felvett értékeit és a képernyőpontok függőleges koordinátáit is meg kell feleltetnünk egymásnak. Milyen összefüggést kell használnunk ehhez? Készítsünk programot, amely egy adott függvény esetén az intervallum kezdő és végpontjának értékétől függően végrehajtja ezt a megfeleltetést!

12.70

Az előző feladatokban alkalmazott módszerrel a grafikon mindkét irányban pontosan lefedi a képernyőt. Ez általában azt eredményezi, hogy az ábra torz lesz, mert a vízszintes és a függőleges tengelyen más-más egységek szerint mérünk. Bizonyos esetekben szükség van arra, hogy egy függvény grafikonját olyan koordinátarendszerben ábrázoljuk, amelynek mindkét tengelyén ugyanazt a léptéket használjuk (ennek nyilvánvalóan csak akkor van értelme, ha az ábrázolt intervallum és az ott felvett függvényértékeket magába foglaló legszűkebb intervallum azonos nagyságrendű). Ilyenkor ahhoz, hogy a grafikon kérdéses része a képre férjen, arányosan változtatnunk kell az egyik tengelyen a léptéket, amíg mindkettő rá nem fér a képre. Készítsünk programot, amely egy adott függvény esetén az intervallum kezdő- és végpontjának értékétől függően végrehajtja ezt a megfeleltetést!

12.71

Az előző feladatokban készített programot egészítsük ki egy olyan – az adott f hozzárendelést megvalósító – szubrutinnal (eljárással), amely a vízszintes képernyőtengely pontjainak megfelelő [a;b]-beli pontokra kiszámítja a függvény értékeit, s a képernyőn kigyújtja az ezt ábrázoló képpontot!

12.72

Az előző feladatban készített program segítségével a függvény grafikonjának a képernyőre eső részletét látjuk. Fejlesszük tovább programunkat úgy, hogy a kurzormozgató gombok segítségével ezt az „ablakot" (mind a négy irányban) mozgatni tudjuk a függvény grafikonján!

12.73

Fejlesszük tovább programunkat úgy, hogy a koordinátatengelyeket – amennyiben az ábrázolt tartományba esnek – is rajzolja ki a gép! Mivel ezek adott

esetben zavaróan is hathatnak, tegyük lehetővé, hogy egy gombnyomásra ki-be kapcsolgathassuk őket!

12.74

Programunk az x → f(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Adott egy (nullától különböző) c állandó. Módosítsuk a függvényértékeket kiszámító szubrutint (eljárást) úgy, hogy az x → f(x+c) függvény grafikonját kapjuk! Hogyan valósítható meg ezek után egyszerűen az x → f(x–c) eset?

12.75

Programunk az x → f(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Adott egy (nullától különböző) c állandó. Módosítsuk a függvényértékeket kiszámító eljárást úgy, hogy az x → f(x)+c függvény grafikonját kapjuk! Hogyan valósítható meg ezek után egyszerűen az x → f(x)–c eset?

12.76

Programunk az x → f(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Módosítsuk a függvényértékeket kiszámító szubrutint (eljárást) úgy, hogy az x → f(–x) függvény grafikonját kapjuk!

12.77

Programunk az x → f(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Módosítsuk a függvényértékeket kiszámító szubrutint (eljárást) úgy, hogy az x → –f(x) függvény grafikonját kapjuk!

12.78

Programunk az x → f(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Adott egy (0-tól és 1-től különböző) c állandó. Módosítsuk a függvényértékeket kiszámító szubrutint úgy, hogy az x → f(c*x) függvény grafikonját kapjuk! Hogyan valósítható meg ezek után egyszerűen az x → f(x/c) eset?

12.79

Programunk az x → f(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Adott egy (0-tól és 1-től különböző) c állandó. Módosítsuk a függvényértékeket kiszámító szubrutint úgy, hogy az x → c*f(x) függvény grafikonját kapjuk! Hogyan valósítható meg ezek után egyszerűen az x → f(x)/c eset?

12.80

Programunk egyik szubrutinja az x → f(x), egy másik szubrutinja pedig az x→g(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Készítsünk

olyan harmadik szubrutint, amelynek segítségével ábrázolható az x→f(x)+g(x) függvény grafikonja!

12.81

Módosítsuk úgy, hogy az x → f(x)–g(x) esetre is alkalmazható legyen! Mire kell ez utóbbi esetben ügyelnünk?

12.82

Programunk egyik szubrutinja az x → f(x), egy másik szubrutinja pedig az x→g(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Készítsünk olyan harmadik szubrutint, amelynek segítségével ábrázolható az x→f(x)*g(x) függvény grafikonja!

12.83

Módosítsuk úgy, hogy az x → f(x)/g(x) esetre is alkalmazható legyen! Mire kell ez utóbbi esetben ügyelnünk?

12.84

Programunk egyik szubrutinja az x → f(x), egy másik szubrutinja pedig az x→g(x) függvény grafikonjának a képernyőre eső részét ábrázolja. Készítsünk harmadik szubrutint, amelynek segítségével ábrázolható az x→f(g(x)) összetett függvény grafikonja! Ügyeljünk az összetett függvény létezésének feltételére!

12.85

Nevezzünk „jó"-nak minden olyan függvényt, amely a gépünk által ismert függvényekből és konstansokból az összeadás, a kivonás, a szorzás, az osztás és az összetett függvény képzése segítségével előállítható, és az őt definiáló képlet y=f(x) alakban, szöveges változó formájában, INPUT utasítással bevihető a gépbe! Készítsünk olyan programot, amely bekéri a képletet, az „ablakot" meghatározó [a;b], illetve [c;d] intervallumokat (ez utóbbi hiányában maga számítja ki a képen függőlegesen ábrázolandó tartomány határait), majd elvégzi a grafikon képre rajzolását! Tegyük lehetővé – amennyiben a látható tartományba esnek – a koordinátatengelyek ábrázolását az egyes tengelyeken az egységek feltüntetését, tetszés szerinti feliratok elhelyezését (s természetesen mindezek kibe kapcsolgatását)!

12.86

Az előző feladatban készített program csak grafikonpontokat szolgáltat, noha a grafikon általában folytonos vonal(ak)ból áll. Módosítsuk a programot úgy, hogy – választásunk szerint – az eredeti módon, pontokból felépített grafikont vagy a „szomszédos" pontok által meghatározott töröttvonalat jelezze ki!

12.87

Egy függvényt úgy is ábrázolhatunk, hogy a grafikonpontokat összekötjük a vízszintes tengellyel, más szóval bevonalkázzuk a „görbe alatti" (negatív értékek esetén természetesen a „görbe feletti") területet. Készítsünk olyan programot, amely ilyen grafikonokat (is) tud rajzolni!

12.88

Az ábrázolt függvény grafikonját kifejezőbbé, plasztikusabbá tehetjük azáltal, ha a grafikon pontjaiból kiindulva egy-egy (rövidke) függőleges szakaszt húzunk, amelynek nagysága a függvénygörbe ottani meredekségével arányos (ha a meredekség pozitív, akkor lefelé, ha pedig negatív, akkor felfelé húzzuk a szakaszt). Készítsünk olyan programot, amely ilyen grafikonokat (is) tud rajzolni!

12.89

Egy egyismeretlenes egyenlet (vagy egyenlőtlenség) lényegileg nem más, mint két egyváltozós függvény a megfelelő relációs jellel (jelekkel) összekapcsolva. Ha ezeket a függvényeket ugyanabban a koordinátarendszerben ábrázoljuk, a grafikonokról leolvasható, hogy hol teljesül(nek) a kérdéses reláció(k). Készítsünk programot, amely alkalmas egyenletek, egyenlőtlenségek grafikus megoldására!

12.90

Ábrázoljuk az

$$x(t) = r*\sin(t),$$
$$y(t) = r*\cos(t)$$

paraméteres alakban adott kört a $0<t\leq 6.28$ intervallumban! Igazítsuk a lépésközt a képernyő felbontóképességéhez és a gép sebességéhez!

12.91

Ábrázoljuk az

$$x(t) = a*\sin(t),$$
$$y(t) = b*\cos(t)$$

paraméteres alakban adott ellipszist!

12.92

Nagyon látványosak a fizikából ismert Lissajous görbék:

$$x(t) = \sin(a*t),$$
$$y(t) = \cos(b*t).$$

Érdemes többféle a, b paraméterpárral futtatni.

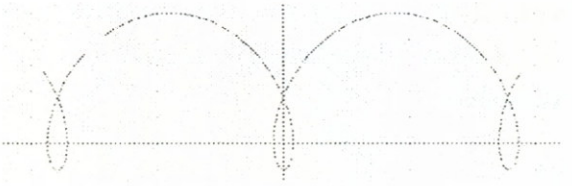
12.93

Ábrázoljuk az

$x(t) = a*t - b*\sin(t)$,

$y(t) = a - b*\cos(t)$

paraméteres alakban adott görbét (ciklois)!



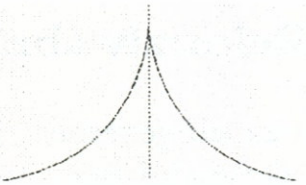
12.94

Ábrázoljuk az

$x(t) = a * (\log(\tan(t/2)) + \cos(t))$,

$y(t) = a * \sin(t)$

paraméteres alakban adott görbét (traktrix)!



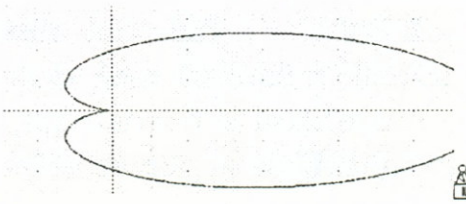
12.95

Ábrázoljuk az

$x = \cos(t*(1+\cos t))$,

$y = a*\sin(t*(1+\cos t))$

görbét (kardioid)!



12.96

Ábrázoljuk az

$x = a*(\cos t)^3$,

$y = a*(\sin t)^3$

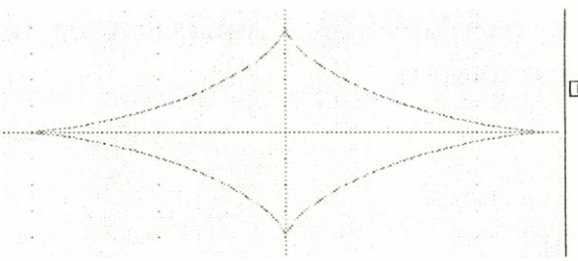
függvényt (asztrois)!

12.97

Ábrázoljuk az (r;s) polárkoordináta-rendszerben az

$r = a*s$

archimédeszi spirálist!



12.98

Ábrázoljuk az (r;s) polárkoordináta-rendszerben az

$r = a/s$

hiperbolikus spirálist!

12.99

Ábrázoljuk az (r;s) polárkoordináta-rendszerben az

$r = a*\exp(b*s)$

logaritmikus spirálist!

12.100

Ábrázoljuk az (r;s) polárkoordináta-rendszerben az

$$r = a*(1+\cos(s))$$

kardioidot!

12.101

Ábrázoljuk az (r;s) polárkoordináta-rendszerben az

$$r = a*\sin(3*s)$$

háromlevelű rozettát!

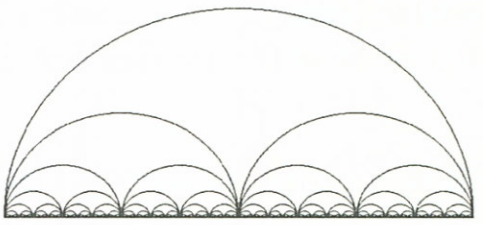
12.4 Rekurzív ábrák

Ebben az alfejezetben ábrasorozatokkal foglalkozunk. Ezeket rekurzív módon fogjuk definiálni, azaz minden sorozatnak megadjuk az első elemét, továbbá azt a szabályt, amelynek segítségével az n. ábrából megkaphatjuk az (n+1). elemét. Az aktuális n értéket a rekurzió szintjének vagy mélységének nevezzük.

Felhívjuk a figyelmet arra, hogy a kitűzött feladatok megoldására szolgáló algoritmusok közül a rekurzívak **általában** nagyságrendekkel egyszerűbbek, mint a nem rekurzívak (ciklussal magadottak). Fiatalabbaknak célszerű Logo-ban (teknőcgrafika) megoldani a feladatokat, de a BASIC, vagy Pascal nyelven írott változatok tanulságosabbak – persze nehezebbek is.

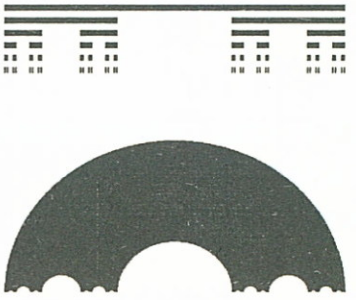
12.102

A 12.1. részben ciklusokkal oldottuk meg az alábbi „egymásba skatulyázott" félköröket (téglalapokat). Csináljuk meg ugyanezt rekurzívan is.

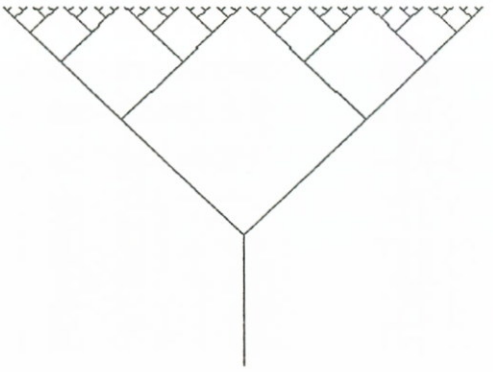


12.103

Hagyjuk el egy egységnyi hosszúságú szakaszból a középső harmadát; ugyancsak hagyjuk el az így megmaradt szakaszok középső harmadát s így tovább! Készítsünk programot, amely a képernyőre rajzolja az N. rekurzív szint ábráját! (Cantor, vagy triadikus halmaz.)



12.104



Készítsünk N-edrendű „egyenes” bináris fát rajzoló programot! A fa „törzse” (1. ága) egy függőleges szakasz, továbbá minden (nem csúcsban végződő) ága a gyökértől távolabbi végén két, az eredeti ág irányával jobbra és balra egyaránt 45-45 fokos szöget bezáró, fele olyan hosszú szakaszban folytatódik (az elsőrendű tehát egy Y-hoz hasonló alakzat lesz, a másodrendű egy olyan Y-hoz hasonló, amelynek mindkét szárát egy-egy újabb Y alkotja, ... A hatodrendű pl. így néz ki:

12.105

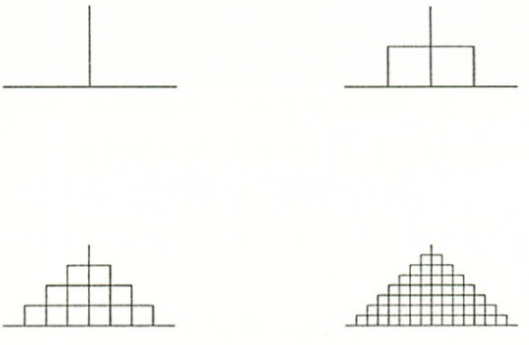
Teknőcgrafikát használva az ábrát úgy célszerű megrajzolni, hogy a rekurzív eljárás végén a teknőc (kurzor) ugyanolyan állapotban (hely, irány) legyen, mint az eljárás elején. A nemrekurzív változat lényegesen bonyolultabb. Jelölje ebben a balra fordulásokat 1, a jobbra fordulásokat 0 számjegy! Így például egy konkrét pontba eljutást az 11010 számsorozat jellemezhet. Egy ilyen pontból a következőképpen juthatunk tovább:

- ha még nem vagyunk ág végén, akkor menjünk tovább balra (az új ponthoz tartozó számsorozat: 110101),
- ha ág végén vagyunk, akkor lépjünk vissza az utolsó balrafordulóshoz, s ebből a pontból kell bal helyett jobbra menni (az új ponthoz tartozó számsorozat: 1100).

12.106

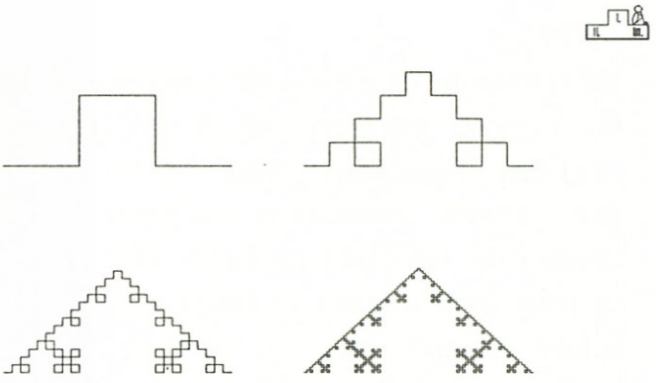
Az aktuális szakasz hosszát a számsorozat elemei számából, irányát pedig a balra-, illetve jobbrafordulások számából határozhatjuk meg.

12.107



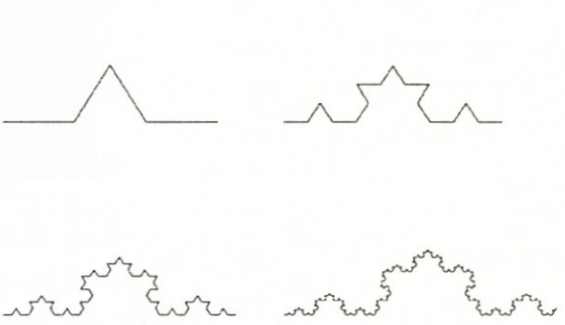
Egy, a csúcsán álló négyzet oldalai és átlói által meghatározott derékszögű háromszögekbe az átlókkal párhuzamos oldalú, maximális területű négyzeteket írunk, az így keletkező újabb derékszögű háromszögekbe ugyanígy négyzeteket rajzolunk, és így tovább. Az ábrán a rekurzív ábrasorozat 4. szintjét látjuk. Készítsünk programot, amely a képernyőre rajzolja az N. rekurzív szint ábráját!

12.108



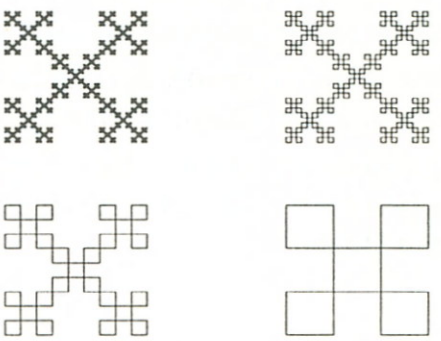
Az ábrák egy rekurzív sorozat első négy elemét mutatják (minden szakasz középső harmadát a rá, mint alapra emelt négyzet másik három oldalával helyettesítjük). Készítsünk programot, amely megrajzolja a képernyőre az N. rekurzív szint ábráját! (Négyzetes Koch-görbe.)

12.109



Az ábrák egy rekurzív sorozat első négy elemét mutatják (minden szakasz középső harmadát a rá, mint alapra emelt szabályos háromszög másik két oldalával váltjuk ki). Készítsünk programot, amely megrajzolja a képernyőre az N. rekurzív szint ábráját! (Koch görbe.)

12.110



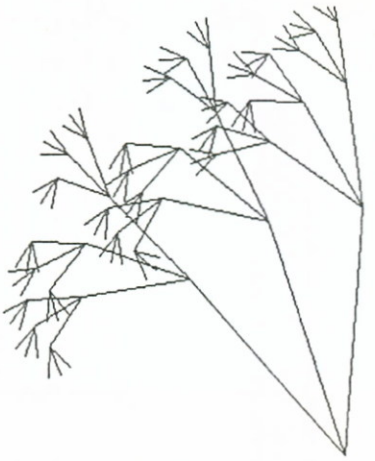
Az ábrák egy rekurzív sorozat első négy elemét mutatják (minden szakaszt az első szint ábrájával helyettesítünk, felváltva ki-, illetve befelé fordítva; a szakaszok egyenlőek, a szögek 120 fokosak). Készítsünk programot, amely megrajzolja a képernyőre az N. rekurzív szint ábráját!

12.111

Készítsünk N-edrendű „egyenes", szintenként K ágú fát rajzoló programot! Az ilyen fa „törzse" (1. ága) egy függőleges szakasz, továbbá minden (nem csúcsban végződő) ága a gyökértől távolabbi végén K, az eredeti ág egyenesére szimmetrikusan elhelyezkedő, egymással 180/(n+2) fokos szöget bezáró, fele olyan hosszú szakaszban folytatódik.

12.112

Készítsünk N-edrendű „szélfútta" bináris fát rajzoló programot! Az ilyen fa ugyanúgy építendő fel, mint az „egyenes", csak az új ágak az eredeti iránnyal balra és jobbra nem azonos mértékben hajlanak el, hanem pl. balra 50, jobbra pedig 30 fokos szöggel.

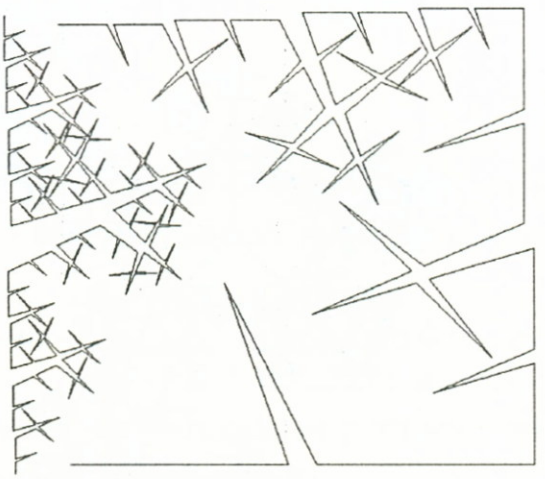


12.113

Készítsünk N-edrendű „szélfútta", szintenként K ágú fát rajzoló programot! Az ilyen fa ugyanúgy építendő fel, mint az „egyenes", csak az új ágak egymással nem ugyanazt a szöget zárják be, hanem valamilyen monoton (pl. balról jobbra növekvő), a fa egészére jellemző rend szerint változnak.

12.114

Egy rekurzív ábrasorozat első négy ábráját láthatjuk. Készítsünk programot, amely a képernyőre rajzolja az N. rekurzív szint ábráját!



12.115

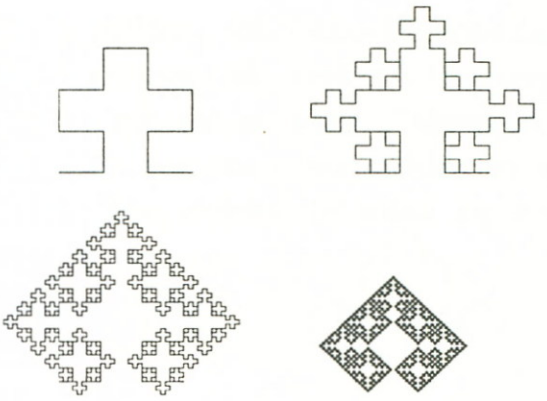
Húzzuk meg egy „talpán álló" szabályos háromszög középvonalait, az így keletkező kis háromszögek közül minden „talpán állónak" ugyancsak húzzuk meg a középvonalait s így tovább! Készítsünk programot, amely a képernyőre rajzolja az N. rekurzív szint ábráját!

12.116

Három ilyen kisebb háromszög van, tehát a rekurzív eljárás háromszor hívja meg önmagát. E három esetet megkaphatjuk egymásból 120 fokos forgatásokkal, ezért a hívást – megfelelő elmozdulásokkal – egy háromszor lefutó ciklusba tehetjük.

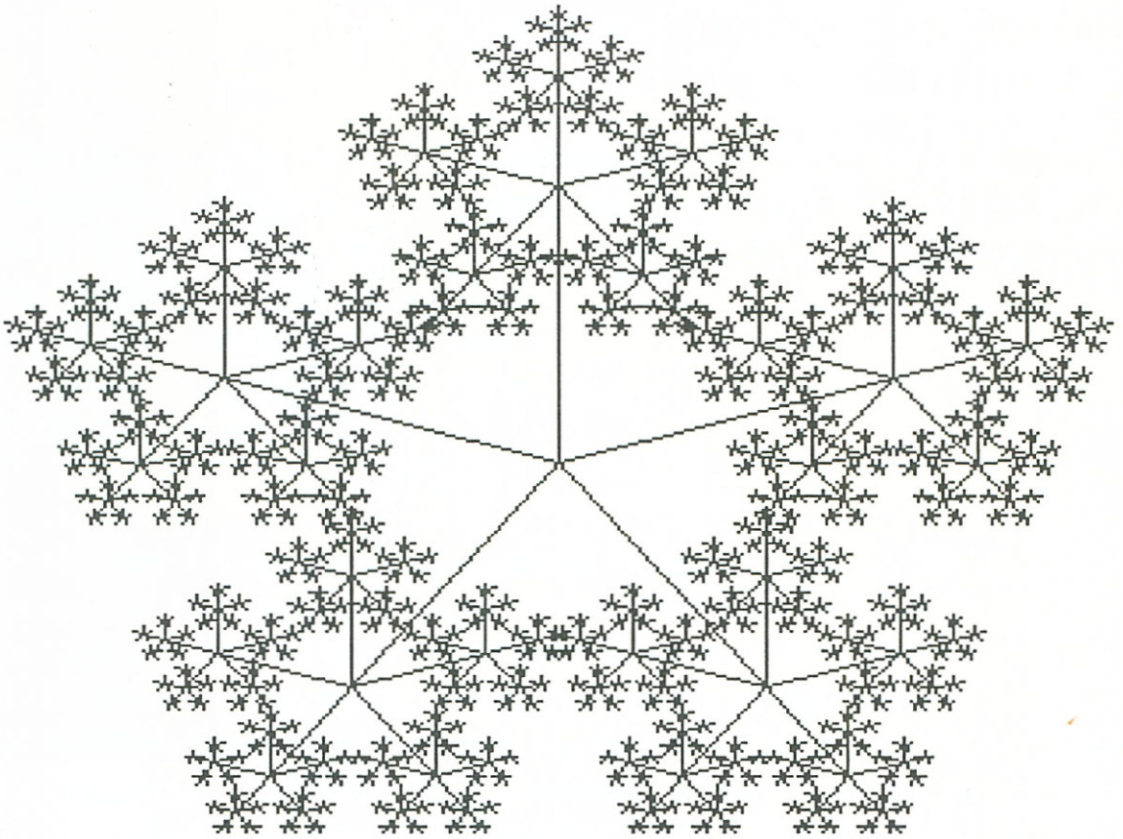
12.117

Az alábbi fraktál generálása nagyon hasonlít ahhoz, ahogyan az 12.115. és 12.116. ábrákat nyertük. Mi lehet vajon a képzési szabály?



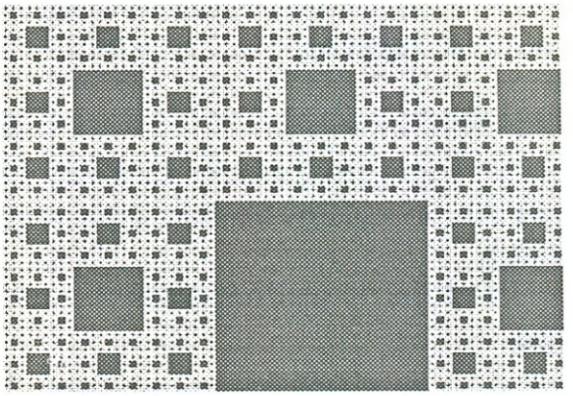
12.118

A Sierpinski-görbét definiáló rekurzív ábráját alább láthatjuk (a rekurzió a rajzokról egyszerűen leolvasható). Készítsünk programot, amely megrajzolja a képernyőre az N. rekurzív szint ábráját!



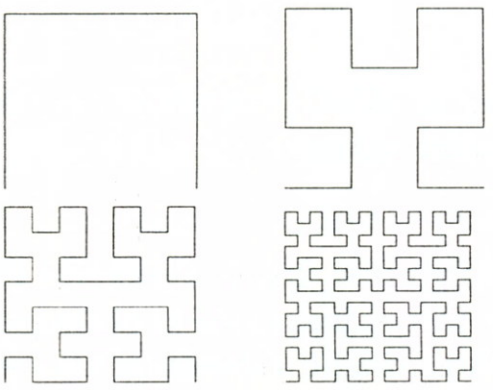
12.119

Az ún. Sierpinski-csipke (Cantor-terítő) a Cantor-halmaz síkbeli megfelelője, ahhoz hasonlóan származtatható: egy egységnégyzetből elhagyjuk annak középső harmadát, az így maradt 8 kis négyzet mindegyikének a középső harmadát, ... Készítsünk programot, amely a képernyőre rajzolja az N. rekurzív szint ábráját!



12.120

Az alábbi görbét Hilbertről nevezték el.



12.5 Térbeli ábrázolás

A térbeli ábrázolás számítógéppel azt jelenti, hogy a monitor kétdimenziós képernyőjén háromdimenziósnak tűnő ábrákat készítünk. Hasonló problémáról van szó, amikor papírra szeretnénk rajzolni.

Alapvetően kétfajta ábrázolás szerepel feladatainkban: az axonometrikus és a perspektivikus ábrázolás. Szinte minden feladat megoldható mindkét módon, a feladatok zöme axonometrikus ábrázolásra készült. Ha a feladatot perspektivikusan célszerű megoldani, akkor ezt a feladat szöveget tartalmazza.

Amikor a monitor síkbeli képernyőjén térbeli koordinátarendszert ábrázolunk, akkor az az általános ábrázolás, amikor az xy síkra (mely vízszintes) rálátunk, s szembe velünk a függőleges z tengely van.

12.121

Írjunk programot, amely ábrázolja a 3 térbeli koordináta-tengelyt, és adott intervallumban beolvasott P(x, y, z) pontokat! Az x tengely zárjon be a vízszintessel 30, az y tengely 45 fokot, a z tengely legyen függőleges!

12.122

Írjuk meg az előző programot úgy, hogy az x, y tengely elhelyezkedése bemenő paraméter legyen!

12.123

Írjunk programot, amely az előző feladatban megadott koordináta-rendszerben adott élhosszúságú kockát vagy téglatestet ábrázol! (Csak a látható éleket ábrázoljuk!)

12.124

Írjuk meg az előző programot úgy, hogy a láthatatlan éleket is ábrázolja szaggatott vonallal!

12.125

Írjuk meg az előző programot úgy, hogy a látható lapokat a program különböző színűekre színezze!

12.126

Írjunk programot, amely a monitor képernyőjének koordinátarendszerében tetszőleges háromdimenziós koordinátarendszert ábrázol úgy, hogy bemenő paraméternek az origót, valamint a három tengely egy-egy pontját adjuk meg!

12.127

Módosítsuk az előző programot úgy, hogy a térbeli koordinátarendszerben a program adott intervallumba eső pontot is tudjon ábrázolni!

12.128

Írjunk programot, amely a fejezet első feladatában megadott koordináta-rendszerben ábrázol

1. adott háromszög alapú hasábot,
2. adott négyszög alapú hasábot,
3. adott háromszög alapú gúlát,
4. adott négyszög alapú gúlát!

12.129

Módosítsuk az előző programot úgy, hogy a látható lapokat különböző színűekre színezzük!

12.130

Ábrázoljunk egy adott kockát perspektivikusan! A horizont legyen a képernyő felső, illetve alsó széle!

12.131 

Ábrázoljunk a képernyőn gömböt!

12.132 

Oldjuk meg az előző feladatot úgy, hogy a gömb nem-látható felén ne rajzoljunk hosszúsági és szélességi köröket!

12.133 

Készítsünk programot, amely térbeli koordinátarendszerben ábrázol adott sugarú és magasságú

1. hengert,
2. kúpot!

12.134 

Ábrázoljunk térbeli koordináta-rendszerben forgási hiperboloidot!

12.135 

Rajzoljunk a képernyő síkjába adott négyzetet, majd függőleges, illetve vízszintes tengely körül forgassuk el és 10 fokonként perspektivikusan ábrázoljuk az elforgatott négyzetet!

12.136 

Írjunk programot, amely a képernyő síkjába rajzol adott sugarú kört, majd függőleges, illetve vízszintes tengely körül elforgatja, és 10 fokonként perspektivikusan ábrázolja az elforgatott kört!

12.137 

Készítsünk programot, amely térbeli koordináta-rendszerben ábrázolt kockát adott vektorral eltolva is ábrázol!

12.138 

Módosítsuk az előző programot úgy, hogy az azonos síkban levő lapokat azonos színnel színezze ki!

12.139 

Tükrözzünk adott pontra középpontosan térbeli koordinátarendszerben ábrázolt kockát!

12.140 

Írjuk meg az előző programot úgy, hogy az azonos síkban levő lapokat a program azonos színnel színezze ki!

12.141 

Tükrözzünk adott síkra térbeli koordináta-rendszerben ábrázolt kockát!

12.142 

Módosítsuk az előző programot úgy, hogy az azonos síkban levő lapokat a program azonos színnel színezze ki!

12.143 

Forgassuk el adott tengely körül térbeli koordináta-rendszerben ábrázolt kockát!

12.144 

Írjunk programot, amely térbeli koordináta-rendszerben ábrázolt kockát adott pontra, adott mértékben kicsinyít, illetve nagyít!

12.145 

Alakítsuk át az előző programot úgy, hogy az azonos síkban levő lapokat a program azonos színnel színezze ki!

12.146 

Készítsünk programot a perspektivikus ábrázolás szemléltetésére! Legyen a horizont a képernyő vízszintes felezőegyenese! Osszuk a képernyőt

- vízszintesen,
- függőlegesen,
- vízszintesen és függőlegesen is

négy részre, és ábrázoljunk mindegyik mezőben egy-egy kockát!

12.147 

Írjunk programot, amely kúpfelületet ábrázol! Rajzolja meg a program a kúpnak a tengellyel adott alfa szöget bezáró síkmetszeteit!

12.148 

Írjunk programot, amely tetszőleges, látható csúcsaival megadott poliédert ábrázol, majd azt

- adott vektorral eltolja,
- adott szöggel függőleges tengely körül elforgatja,
- adott szöggel vízszintes tengely körül elforgatja,
- adott mértékben kicsinyíti,
- adott mértékben nagyítja!

12.149 

Alakítsuk át az előző feladat megoldását úgy, hogy a program a transzformált poliéder csúcspontjait tárolja, és így újabb transzformációt végezhet vele!

12.150 

Készítsünk programot térbeli függvények ábrázolására! Definiáljunk egy xy síkrészt (pl. –15-től +15 ig)! Számítsa ki a program az így adott sík környezetébe eső függvényértékeket úgy, hogy konstans x-ek mellett kiszámolja a f(c, y); konstans y-ok mellett az f(x, c) értékeket, majd ábrázolja a görbehálót!

12.151 

Írjunk programot, amely egy térben ábrázolt kockának az

- xy síkkal párhuzamos,
- xz síkkal párhuzamos,
- yz síkkal párhuzamos

síkmetszetét megrajzolja és beszínezi!

12.152 

Rajzoljuk meg és színezzük be az xy síkon álló hengernek vagy kúpnak az

- xy síkkal párhuzamos,
- xz síkkal párhuzamos,
- yz síkkal párhuzamos

síkmetszetét!

12.153 

Írjunk programot, amely térben ábrázolt kockának tetszőleges csúcsából indulva adott kisebb kockát kivág, s a maradék testet látható éleivel ábrázolja!

12.6 Hang

A különböző számítógépek hanggenerálása nagyon eltérő, függ a számítógép kiépítettségétől, hangkártyával való ellátottságától. A hang megszólaltatásának módja gépenként más és más. A legtöbb esetben a hangfrekvenciát és a hangerőt is változtathatjuk. Napjainkban egyre több iskola rendelkezik multimédia lejátszására alkalmas számítógépekkel. Ezen gépeken általában a hangrögzítés, feldolgozás és lejátszás egyaránt megoldható. Természetesen ez nem csak a rendelkezésre álló hardveren, hanem a szoftveren is múlik.

A feladatoknak nem mindegyike oldható meg minden gépen, valamint lesz olyan feladat, amelyik az egyik gépen könnyű feladatnak számít, a másikon nehéznek.

Mindenképpen fontos, hogy a feladat megoldásának elkezdése előtt alaposan tanulmányozzuk az adott gép lehetőségeit.

Megegyezhetünk abban, hogy a kisoktáv jelölésére a kisbetűket (c, d, e, ...), az egyvonalas oktáv jelölésére az aposztrófot (c', d', e', ..), a kétvonalas oktáv jelölésére a dupla aposztrófot (c", d", e", ...) használjuk.

Megegyezhetünk abban is, hogy kottát rajzoló feladatok esetén azt violinkulcsban írjuk. (Természetesen megírhatók ezek a programok másfajta beolvasással, más kulcsban is.)

12.154 

Írjunk programot, amely a „c" billentyű benyomására megszólaltatja a c' hangot (egyvonalas c) egy másodpercig!

12.155 

Írjunk programot, amely a „c" billentyű benyomására megszólaltatja a c' hangot addig, amíg a billentyűt nyomva tartjuk!

12.156 

Írjunk programot, amely a megfelelő billentyűk lenyomásakor megszólaltatja a c', d', e', f', g', a', h' hangok valamelyikét addig, amíg a billentyűt nyomva tartjuk!

12.157 

Írjuk át az előző programot úgy, hogy két oktáv megszólaltatására legyen képes a c, d, e, f, g, a, h, C, D, E, F, G, A, H billentyűk által!

12.158 

Írjuk át az előző programot úgy, hogy három oktáv megszólaltatására legyen képes (kis oktáv, egyvonalas, kétvonalas oktáv)!

12.159 

Írjunk programot a következő ritmusképletek bemutatására: éles, nyújtott, szinkópa. A hangmagasságokat karaktersorozatként olvassuk be (pl. szinkópa: cec)!

12.160 

Írjunk programot, amely dallamok tárolását segíti! Két tömbben tároljuk a hangmagasságokat és hanghosszúságokat. A tárolt dallamot játssza le programunk a kívánt sebességgel!

12.161 

Írjuk meg az előző feladat megoldását úgy, hogy előjegyzéses hangokat is tudjon a gép megszólaltatni, és a hangterjedelem legalább 3 oktáv legyen!

12.162 

Írjunk programot, amely dallamok tárolását segíti! Két tömbben tároljuk a hangmagasságokat és hanghosszúságokat. A tárolt dallamot játssza le programunk kívánt hangon kezdve, a kívánt hangnemben (transzponálás)!

12.163 

Írjuk meg az előző feladat megoldását úgy, hogy a program a dallamot nemcsak tetszőleges hangnembe transzponálja és eljátssza, hanem a transzponált dallamot ki is írja!

12.164 

Írjunk programot, amely a maximum 3 oktáv terjedelemben tárolt dallamot a megadott dó-nak megfelelően szolmizációs hangokkal leírja!

12.165 

Írjuk át az előző programot úgy, hogy a képernyőn mindig csak egy szolmizációs hang jelenjen meg, a hanghosszúságnak megfelelő időtartamig! Lehessen választani, hogy közben szóljon-e a megfelelő hang!

12.166 

Írjunk programot, amely a max. 3 oktáv terjedelemben tárolt dallamot a menüből kiválasztott tempójelzésnek (pl. allegro) megfelelő sebességgel játssza le!

12.167 

Írjunk programot, amely hangközök bemutatására szolgál! A menüből kiválasztott hangköz (pl. kis szekund) és a megadott hang alapján megszólaltatja a megfelelő hangközt, valamint kiírja az eredményt! (pl. c és kis szekund eredményeképp megszólaltatja és kiírja: c - desz.)

12.168 

Írjunk programot, amely hangközök felismerését gyakoroltatja! A véletlenszerűen megszólaló hangközt a képernyőn megjelenő hangközökből a megfelelő kiválasztásával kell felismerni.

12.169 

Írjunk programot, amely hangközök felismerését gyakoroltatja! A véletlenszerűen megjelenő kérdésre (pl. c' - felső kisterce) kell a felhasználónak válaszolni.

12.170 

Írjunk programot, amely hangok felismerését gyakoroltatja. A véletlenszerűen megszólaló hangot (melynek neve is megjelenik, pl. G) követő hangot kell felismerni, és válaszként megadni.

12.171 

Írjunk programot, amely a maximum 3 oktáv terjedelemben tárolt dallamot

A. a kezdőhangra,

B. a záróhangra,

C. megadott tengelyhangra tükrözi.

12.172 

Írjunk programot, amely a maximum 3 oktáv terjedelemben tárolt dallamot rákfordítással megfordítja!

12.173 

Írjunk programot, amely a maximum 3 oktáv terjedelemben tárolt dallamot az előző két program felhasználásával tükrözi és rákfordítja!

12.174 

Írjunk programot, amely az előző három program valamelyikével generált dallamot az eredetivel együtt két szólamban megszólaltatja!

12.175 

Írjunk programot, amely maximum 3 oktáv terjedelemben tárolt kétszólamú dallamot megszólaltat!

12.176 

Írjunk programot különböző zörejek, zajok előállítására! (Például: kopogás, lépések, autókürt, motorzaj, szél, madárcsicsergés, köhögés stb.)

12.177 

Írjunk programot, amely menüből kiválasztott két zajt, zörejt egyszerre megszólaltat!

12.178 

Írjunk programot, amely tetszőleges maximum 2 oktáv terjedelmű dallamnak megrajzolja a kottáját (ütemvonalak nélkül)!

12.179 

Írjunk programot, amely tetszőleges, maximum 2 oktáv terjedelmű, 4 ütem hosszú dallamnak megrajzolja a kottáját (ütemvonalakkal, violinkulccsal, ütemjelzéssel, előjegyzéssel együtt)!

12.180 

Írjunk programot, amely tetszőleges, maximum 2 oktáv terjedelmű, 4 ütem hosszú szolmizációs hangokkal megadott dallamnak megrajzolja kottáját adott dó esetén!

12.181 

Írjuk át az előző két programot úgy, hogy a dallamot le is játssza a gép, és a kotta alatt nyíl mutassa, hogy hol tartunk éppen!

12.182 

Írjunk programot, amely tetszőleges, maximum 2 oktáv terjedelmű, 4 ütem hosszú szolmizációs hangokkal megadott dallamot megad abc-s hangokkal adott dó esetén!

12.183 

Írjunk programot, amely modális hangsorok felismerését gyakoroltatja! A memóriában tárolt véletlenszerű sorrendben eljátszott dallamról kell a felhasználónak felismerni, milyen hangsorban íródott.

12.184 

Írjunk programot, amely dallamot generál véletlenszerűen! Paraméterek legyenek:

- hosszúság,
- hangterjedelem,
- a dallamban milyen ritmusértékek szerepelhetnek,
- a dallamban milyen szolmizációs hangok szerepelhetnek,
- a dó helye,
- befejező hang.

12.185 

Írjuk meg az előző feladat megoldását úgy, hogy visszatérő témát is tartalmazzon a generált dallam! Kérjük a felhasználótól a témát szolmizálva!

12.186 

Írjunk programot, amely pentaton dallamot generál véletlenszerűen C - dóban. Paraméterek legyenek a hosszúság, a hangterjedelem, a dallamban milyen ritmusértékek szerepelhetnek.

12.187 

Írjunk programot egy beolvasott ritmussor ütembe rendezésére!

12.188 

Írjunk programot, amely a klaviatúrát a zongorabillentyűkhöz hasonlóan használja, a billentyűzeten lejátszott dallamot megszólaltatja!

12.189 

Írjuk meg az előző feladat megoldását úgy, hogy a játszott dallamot (ritmus nélkül) le is kottázza a gép a képernyőre!

12.190 

Írjunk programot, amely egyszerű dallamokat tárol, majd kettőt lejátszik és megkérdezi: egyforma – hasonló – vagy különböző-e!

12.191 

Írjunk programot, amely megadott hangközökhöz megkérdezi az enharmónikus hangközt!

12.192 

Írjunk programot, amely gyakoroltatja egy hangköz oktávhoz történő kiegészítését!

12.193 

Írjunk programot, amely beolvasott dallam esetén megadja a dallam fogástáblázatát furulyán!

13. FEJEZET

JÁTÉKOS-ÉRDEKES FELADATOK

A SZÁMÍTÁSTECHNIKA ALAPJAI
OPERÁCIÓS RENDSZER
SZÖVEGSZERKESZTÉS
ADATBÁZIS- ÉS TÁBLÁZATKEZELÉS

13. JÁTÉKOS-ÉRDEKES FELADATOK

Ezek a feladatok elsősorban táborokra, iskolanapra, vetélkedőkre ajánlottak, de esetenként egy-egy óra, illetve házi feladat színesebbé tételére is szolgálhatnak. Elsősorban mintának szántuk, hogy további, hasonló rejtvények, játékok készítésére ösztönözze az Olvasót.

Érdemes néha – gyakorlati munka keretében – gyerekeknek is kiadni ilyen jellegű feladatok készítését. Sokat tanulhatnak, amíg egy rejtvényes érdekes feladat megszületik. A kellemes, szórakoztató jellegű feladat csábító, így később a gyerekek – de még a felnőttek is – szívesen nyúlnak egy-egy számítástechnikai feladat után.

13.1. A számítástechnika alapjai

A számítástechnika tanításának céljai között a következők (is) szerepelnek:

- Ismerjék meg a tanulók a számítógéppel kapcsolatos alapfogalmakat.
- Keltsük fel a tanulók érdeklődését, és ezzel szorgalmazzuk az egyéni számítástechnikai önképzést.
- Váljanak a számítástechnikai alapismeretek az általános műveltség részévé.

A követelmények között pedig:

- Ismerjék a számítógép felépítését, a leglényegesebb elemek elnevezését, funkcióit, kapcsolásukat.
- Ismerjék a hardver és a szoftver fogalmát.
- Ismerjék a perifériák fajtáit és feladatait.
- Ismerjék a számítógépes adatábrázolási módokat.
- Ismerjék a számítástechnika fejlődésének főbb állomásait.
- Ismerjék Neumann János nevét, a tárolt program lényegét.

A minimum követelmény a 8. évfolyam végén:

- A számítógép főbb egységeinek felismerése funkciói.

Ez alapján három részre oszlik a tananyag. (Számítástechnikai eszközök, Informatikai alapfogalmak, A számítástechnika története)

Új szavakat, kifejezéseket tanulhatnak a feladatokon keresztül, illetve a tanult fogalmakat, ismereteiket alkalmazhatják a megoldás során.

13.1

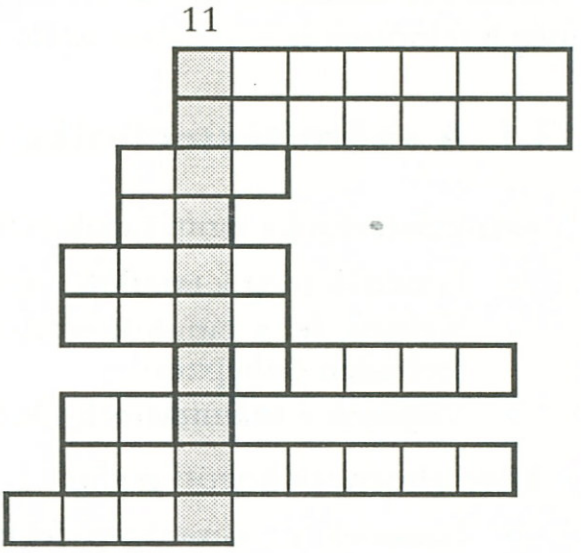
[Futólépésben] Az ábra betűi egy sakktáblán helyezkednek el. Ezek a betűk a futó lépésének szabályai szerint, helyes sorrendben összeolvasva a számítógép egy fontos részét adják megfejtésül. (A kezdőbetű félkövér.)

	Ű		T	
Z		Y		N
	E		E	
T		I		L
	B		L	

13.2

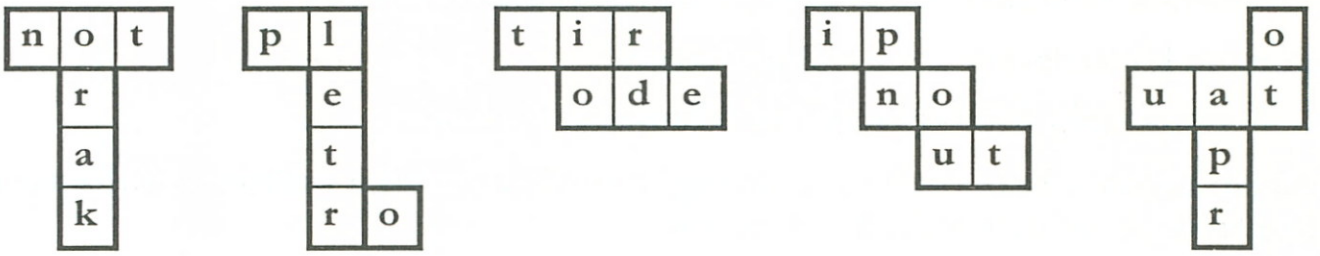
[Rejtvény] Fejtsd meg a következő rejtvényt!

1. Szoftver
2. A gyárakban számítógéppel vezérlik
3. Disk Operating System
4. Személyi számítógép
5. Összeköti a számítógép egyes részeit
6. Mágneslemez röviden
7. .... szerkesztő program
8. Ebben van a számítógép nagy része
9. Printer
10. A képernyőn levő nyilat is mozgathatod vele



13.3

[A kocka hálója] Amelyik hálóból nem lehet kockát alkotni, az azon levő betűkből egy számítástechnikával kapcsolatos értelmes szó alkotható. Ha ezt a szót meg is tudod magyarázni, igazán ügyes vagy!



13.4

[Nyelvi játékok]

1. Húzd alá, amelyik nem illik a többi közé!

   monitor, nyomtató, toll, billentyűzet, egér

2. Úgy kell csoportosítani a szavakat, hogy egy gyűjtőfogalmat, s ennek 3-3 elemét kell megkeresni!

   Mágnesszalag, logikai művelet, egér, adatbeviteli eszköz, vagy, CD, scanner, információhordozó, fényceruza, floppy, tagadás, és

3. Mi a közös ezekben a szavakban?

   tasztatúra, klaviatúra, keyboard

4. Négy számítógéppel kapcsolatos szó betűi vannak összekeverve! Melyek ezek a szavak? Az összekeverés logikája melyikben tér el a többitől?

   A. i o o m n r t
      a o ó m ny t t
      i e ű e b l l n ty z t
      e o l p r t t

   B. a u o a a t m t
      a i ó e r m m
      o e o p r c s s z r
      o e m d m

5. Plusz egy szó. Több összetett szó első tagját leválasztottuk és betűrend szerinti sorrendbe raktuk. Próbáljuk meg kitalálni a szavakat! A kezdőbetűkből összeolvasott szó a számítástechnikában nagyon fontos fogalom!

   adat-, egér-, in-, menü-, mikro-, óra-, rész-

   _____rendszer

   _____kezelés

   _____processzor

   _____mű

   _____probléma

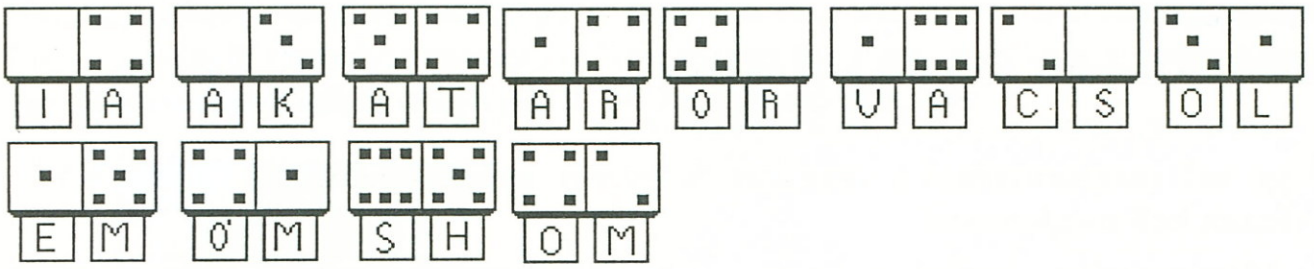
   _____put

   _____bank

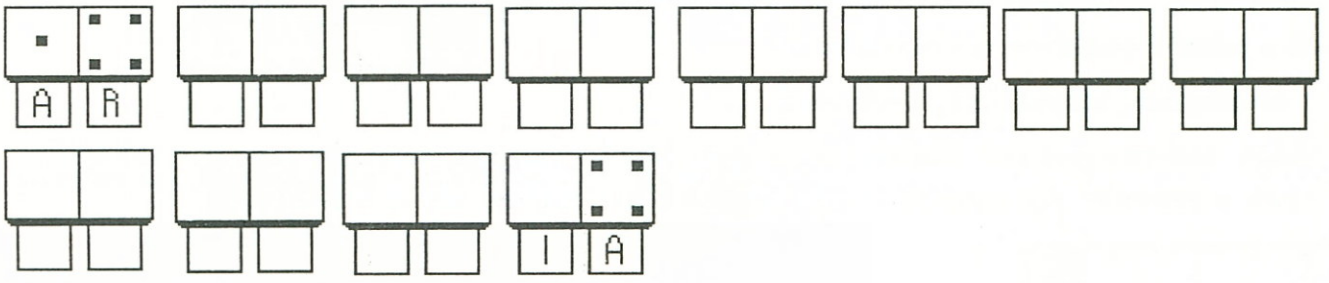
13.5

[Szódominó] Egy számítástechnikával kapcsolatos mondat fejthető meg ebben a feladványban, amelyet a dominójáték szabályai szerint lehet megoldani. Könnyítésül az első és az utolsó két-két betűt beírtuk az ábrába (szavakra neked kell bontanod).

Ezek a dominók:



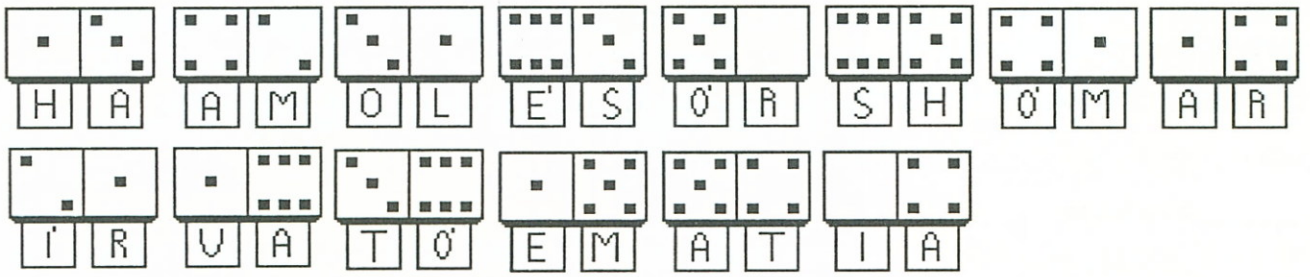
Itt próbálgathatsz:



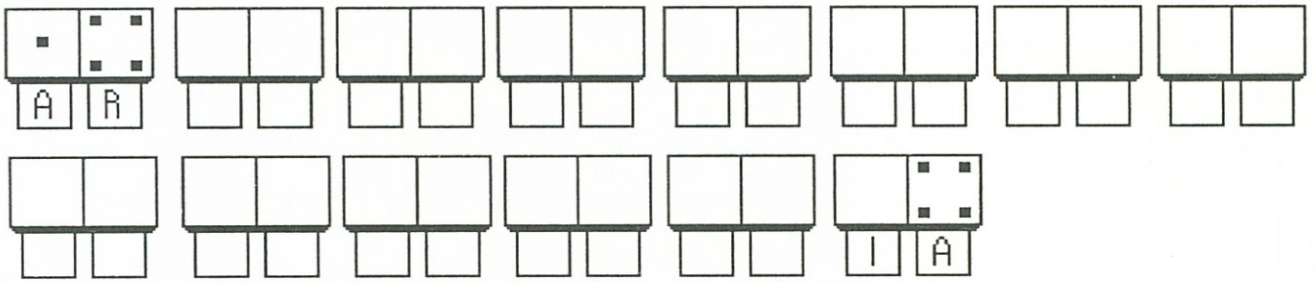
13.6

[Szódominó] Egy számítástechnikával kapcsolatos mondat fejthető meg ebben a feladványban, amelyet a dominójáték szabályai szerint lehet megoldani. Könnyítésül az első és az utolsó két-két betűt beírtuk az ábrába (szavakra neked kell bontanod).

Ezek a dominók:

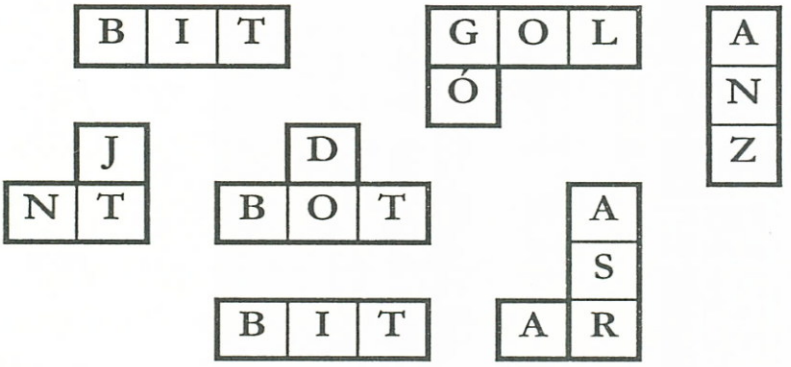


Itt próbálgathatsz:



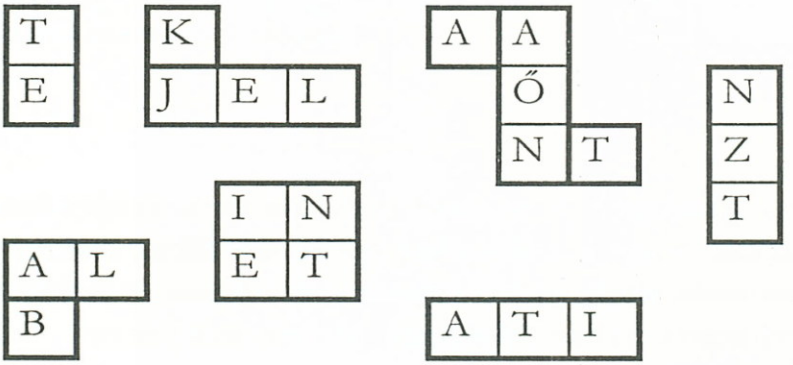
13.7

[Rakosgat] Ha a részeket összerakjuk egy téglalappá, illetve négyzetté, a kapott szöveg egy értelmes mondat lesz!



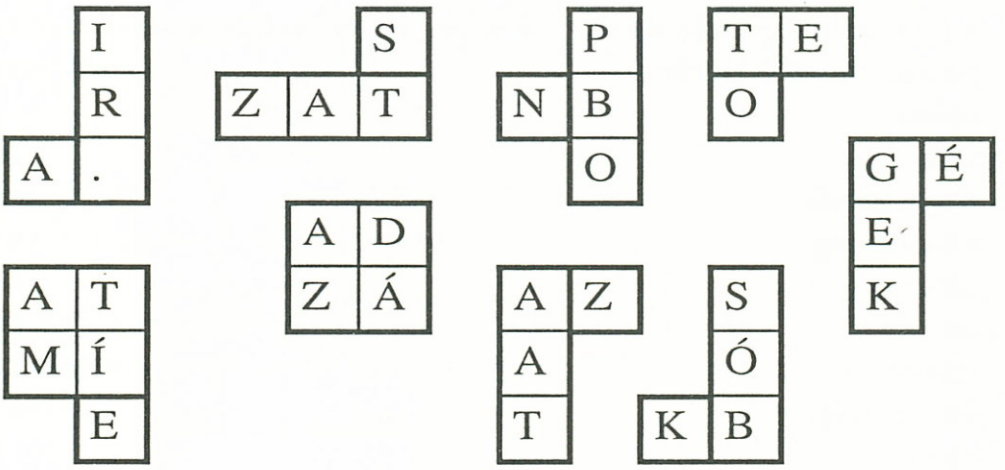
13.8

[Rakosgat] Ha a részeket összerakjuk egy téglalappá, illetve négyzetté, a kapott szöveg egy értelmes mondat lesz!



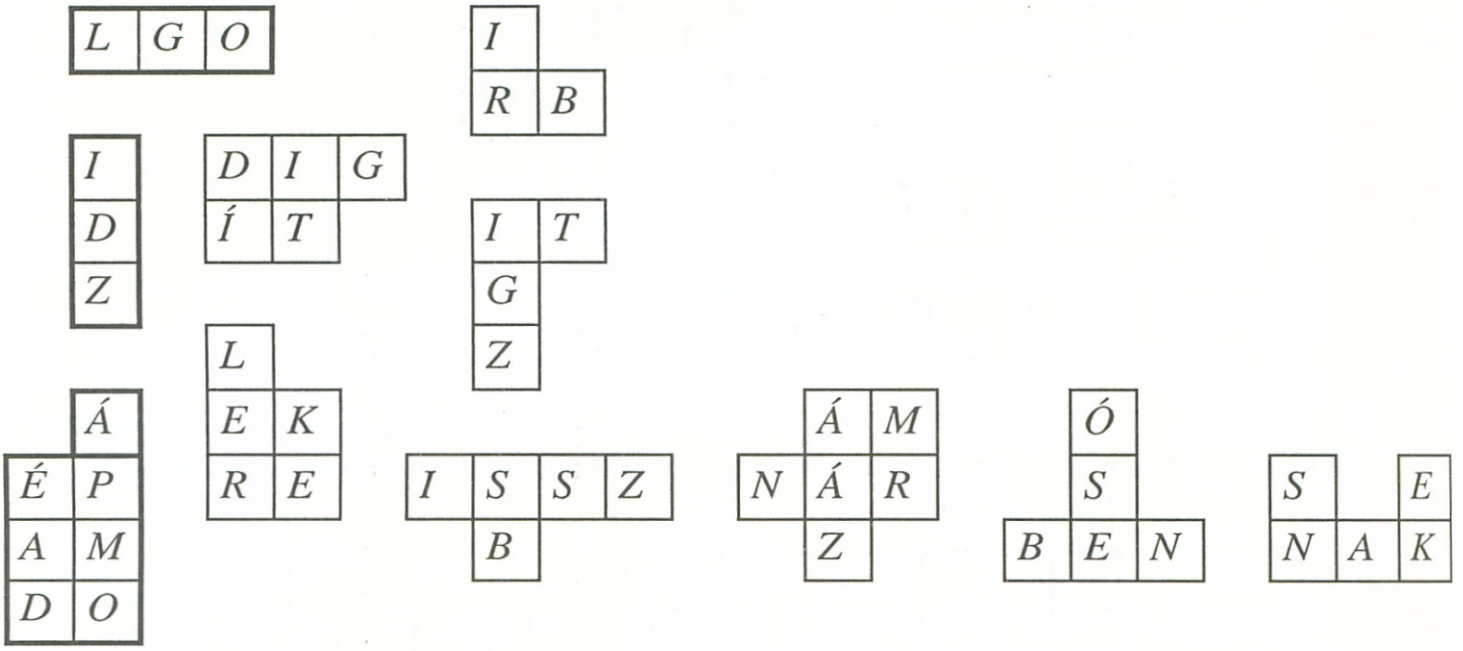
13.9

[Rakosgat] Ha a részeket összerakjuk egy téglalappá, illetve négyzetté, a kapott szöveg egy értelmes mondat lesz!



13.10

[Rakosgat] Ha a részeket összerakjuk egy téglalappá, illetve négyzetté, a kapott szöveg egy értelmes mondat lesz!



13.11

[A gyarapodó szókincs gazdagít] A nyelvet gondolataink kifejezésére használjuk. Ha pedig azt akarjuk, hogy közlésünket ne értsék félre, szükségünk van a gondolatainkat legpontosabban kifejező szavak sokaságára. A számítástechnika történetben is találkozhatunk néhány ilyennel. Melyik mit jelent?

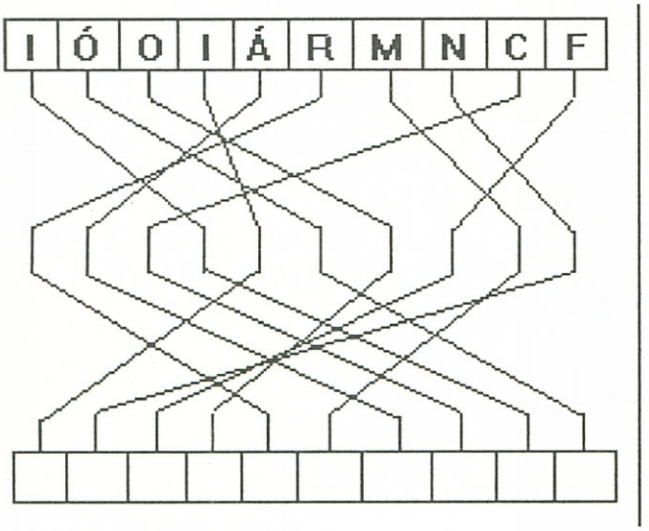
1. szorobán
   - A. régi nemesi rang
   - B. japán számolást segítő eszköz
   - C. középkori fegyver
2. algoritmus
   - A. a feladat megoldásához szükséges lépések felsorolása
   - B. programozási nyelv
   - C. ütem
3. differencia
   - A. fogaskerék
   - B. különbség
   - C. ülésezés
4. analitikus
   - A. írástudó
   - B. északi nép
   - C. elemző

5. digitális
    A. diszkrét értékeken alapuló
    B. újszerű
    C. hangköz
6. univerzális
    A. egyetemes, általános
    B. tételes
    C. mindenre képes
7. memória
    A. áramköri elem
    B. a halál állapota
    C. emlékezet
8. automata
    A. billentyű
    B. önműködő
    C. paradicsomi állapot
9. szinkron
    A. félvezető anyag
    B. időzített
    C. egy időben történő
10. parallelizmus
    A. tudományág
    B. párhuzamosság, hasonlóság
    C. fizetség
11. integrálódik
    A. megtisztul
    B. egységesül
    C. képes felfogni
12. aszinkron
    A. nem egyidejű
    B. hasonlít
    C. kemény anyag
13. orientálás
    A. utazás
    B. keleties
    C. irányítás
14. asszociatív
    A. összekötő, egyesítő
    B. egybehangzó
    C. rendszerező
15. analóg

A. korszerűtlen
B. megfelelő
C. szükségszerű

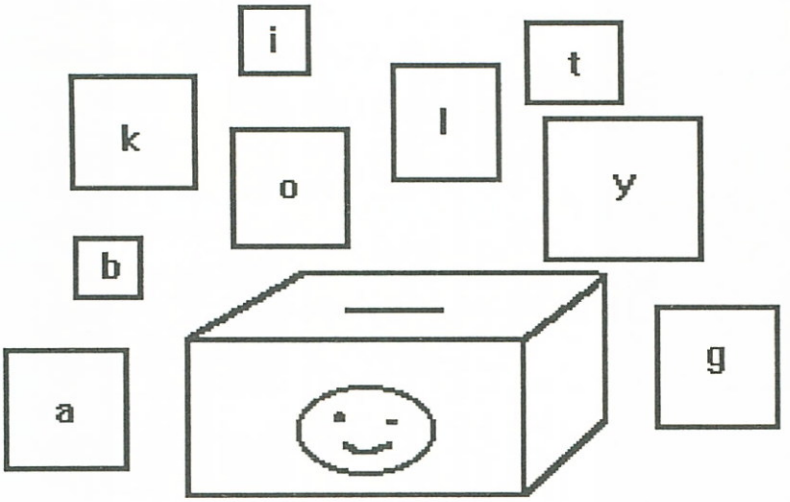
13.12

[Betűcsúszda] A ceruzát balról jobbra haladva csúsztassuk le egymás után a betűket a vonalak mentén az alsó sorba! Így az értelmetlen betűhalmazból pillanatok alatt egy értelmes szó alakul ki. Mit jelent ez a kifejezés?



13.13

[Szemmérték] Állapítsuk meg, melyik fér be a nyíláson! Ezeken levő betűkből egy értelmes szó rakható ki. Mit jelent?



13.14

[Számok és betűk] Próbáljuk megjegyezni 3 perc alatt, hogy melyik betű melyik számhoz tartozik! Ezután a számok mellé írjuk oda a megfelelő betűket! Az nyer, aki leggyorsabban és a legkevesebb hibával írja le a szavakat.

b - 1 ; d - 2 ; e - 3 ; i - 4 ; j - 5 ; k - 6 ; l - 7 ; ó - 8 ; t - 9

a/ 5 ; 3 ; 7 ; a szó: _____

b/ 1 ; 4 ; 9 ; a szó:_____

c/ 6 ; 8 ; 2 ; a szó:_____

*/ 9 ; 4 ; 7 ; 2; 3 ; a szó:_____

13.15

Találjuk ki a szavak kezdőbetűjét a vízszintes sorokban! Ha sikerült, egy híres ember nevét tudhatjuk meg!

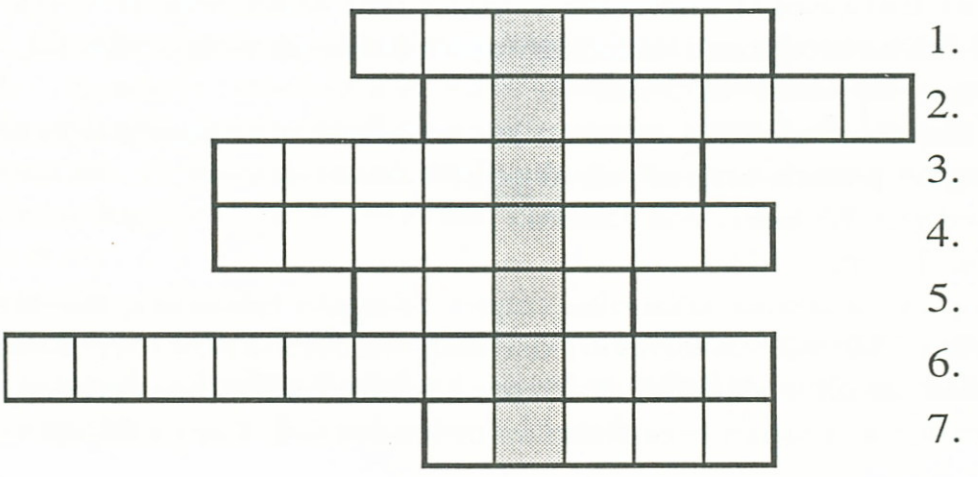
_rocesszor
_utomata
_canner
_ímbusz
_dat
_emez

13.16

Keresztrejtvény

1. Ilyennek kell lennie a számolás eredményének
2. A számítógépnek is kell
3. Számolást segítő eszköz
4. Önműködő
5. Amit a számítógép feldolgozhat
6. Valamikor ez is kellett a számítógépbe
7. Az első elektronikus számítógép

A besatírozott oszlopból megtudhatjuk a számítógép atyjának nevét!



13.17

Utazz velünk a kérdéseken keresztül az időben! (A válaszlehetőségek után megadjuk a következő kérdés számát!)

1. Mi volt az első számológép neve?
   kalkulátor :7
   abakusz : 10
   golyós számolótábla :18
2. Ő is szerkesztett egy eszközt a Leibniz-kereket, amelyet 250 éven keresztül változatlanul alkalmaztak, és egyes gépekben még ma is használnak. Ez egy négy alapműveletes gép, kézi forgató meghajtással, mozgatható beállítóművel. Próbáld másképpen a 9-est!
3. Tévedés! De tanítója férfi volt; korának egyik jól ismert matematikusa, Augustus do Morgan. Még egyszer a 30-as!
4. Hurrá! Latinul a digitus ujj, amelyből a számjegy angol elnevezése a digit ered, és erre vezethető vissza a mi digitális kifejezésünk is. Lépj a 14-re!
5. A XX. század nagy technikai fejlődése nagyrészt az elektronikának köszönhető. Mindez kihatott a számítástechnika fejlődésére is. Pl. a MARK-I-et Aiken 1939-1944 között készítette el – ez Babbage elgondolásával egyezően, központi vezérlésű, elektromechanikus analitikus számítógép volt. Szerinted mennyi vezeték kellett hozzá?
   760 ezer km : 15
   800 km : 31
6. Ő az ötletadó volt. Joseph Marie Jacquard 1810-ben elkészítette a lyukkártya-vezérelt szövőgépet, amely ötletet adott Babbage-nak az elemeire bontott számítási utasítások lyukkártyán való gépbe juttatására. Próbáld még egyszer a 23-at!
7. Nem. A kalkulátor a négy alapművelet végzésére szolgáló egyszerű számológép. Az abakusz sínekbe helyezett apró kövekből áll. A kövecske latin neve calculus, innen származik a kalkulátor szó is. Menj vissza az 1-re!
8. Így volt, pedig ma ez is soknak tűnik. A számlálóbiztosok olyan lyukkártyákkal járták az otthonokat, melyekre a válaszokat lyukasztással vitték fel. A kártyákat egy elektromágnese számláló-berendezéssel dolgozták fel, mindössze négy hét alatt. Új kérdés az 5-ön!
9. Sokan foglalkoztak még mechanikus számoló-automaták fejlesztésével. Ki alkotta meg az „Aritmométer" nevű gépet?
   Gottfried Wilhelm von Leibniz : 2
   C. X. Thomas : 16
10. Így van! Az abakusz számolás céljára szolgáló táblácska, sínekbe helyezett apró kövekből áll. Kb. 3000 éves, őshazája valószínűsíthetően Kína és Egyiptom. Az abakuszok családjába tartozónak tekinthető a kínai eredetű szuan-pan és ennek vállfajai: a japán szorobán; az orosz szcsoti. Lépj a 21-re!

11. Ahogy Babbage a számítógép ősatyja, úgy ki tekinthető a számítógép atyjának?

Neumann János : 32

Kozma László : 25

12. Nem. Falcon, a műszerész, lemezkéket használt szövőszék vezérlésére, ettől kezdve egy szövő több szövőszéket is tudott kezelni. Nézd meg újra a 23-at!

13. Bármily furcsának tűnik, igaz! Babbage el nem készült gépére ADA BYRON (1816-1852) írt programokat, így az első programozónak őt kell tekinteni. Zsenialitását mutatja, hogy ezek szinte mind helyesek. Tiszteletére róla nevezték el a nagyszámítógépekre kifejlesztett Ada programnyelvet. Tovább a 27-re!

14. Wilhelm Schickard, a tübingeni egyetem tanára 1623-ban építette meg mechanikus számológépét. Gépének elkészítéséről levélben értesítette Keplert. Vázlatai alapján Kepler elkészítette a fogaslécekkel működő berendezést. Vajon milyen műveletek elvégzésére volt képes ez a szerkezet?

összeadás, kivonás : 22

mind a négy alapművelet : 29

15. Azért ez túlzás! De 760 ezer áramköri eleme volt. Válassz újra az 5-ben!

16. Rendben! C. X. Thomas 1820-ban továbbfejlesztette Leibniz gépét, és megépítette a négy alapművelet elvégzésére alkalmas „Aritmométer" nevű gépet, amely a mai asztali számítógépek ősének tekinthető. A 23. következik!

17. Az ENIAC-ot a lövedékek gyors röppályaszámítása végett építették meg 1943-1945 között. Ez az első tisztán elektronikus számítógép. Milyen hosszú volt?

3 m :24

30 m : 28

18. A tévedésed csak kicsi. A golyós számolótábla az abakusz némileg módosított európai formája. Vissza az 1-re!

19. Csak nem ez a népszámlálás, hanem az előző tartott ennyi ideig. Vissza a 27-re!

20. Gratulálok! Charles Babbage 1833-ban kigondolta fő művét, az ANALITIC ENGINE - t, amely mechanikus működésű, univerzális, külső programozású, digitális számítógép. A tároló rész sajnos nem készült el. Lépj a 30-ra!

21. Fontos számítási „segédeszközünkhöz", az ujj latin nevéhez is kapcsolódnak ma használatos fogalmak. A kettő közül melyik?

digitális : 4

komputer : 26

22. Tévedés! Húsz évvel később Blaise Pascal gépe a szorzást, osztást nem tudta elvégezni. De mivel Schickard gépéből egyetlen példány sem maradt fenn, Pascal gépe mintapéldány lett. Próbáld újra a 14-est!

23. Mindenesetre egy ilyen karos – fogaskerekes géppel napokon keresztül osztani – szorozni meglehetősen nagy fizikai leterhelést jelentett. A következő lépcsőfokot az automatikus vezérlés megteremtése jelentette. Kit tekinthetünk a következők közül a számítógép ősatyjának?

Joseph Marie Jacquard : 6
Falcon :12
Charles Babbage :20

24. Nem, de 3 m magas volt. Vissza a 17-re!
25. Nem. Kozma László villamosmérnök 1939-ben elkészítette igen gyorsan működő jelfogós gépét, amely mind a négy alapművelet elvégzésére alkalmas volt. Lépj vissza a 11-re!
26. Hát!! Ez is latin eredetű szó. A computer jelentése: összevágni, rovásfára felróni. A komputer szó tehát a számok rögzítésére szolgáló egykori ősi módszerre utal. Újra a 21-re!
27. Hermann Hollerith (1860-1929) ismerte fel először, hogy a számítógép alapvető feladata az adatfeldolgozás. Rendezőgépet dolgozott ki, mellyel 1890-ben az USA-ban a 11. népszámlálás adatait dolgozta fel, 63 millió személyről. Mennyi ideig tartott ez?

négy hétig: 8
hét évig : 19

28. Hihetetlen, de 30 m hosszú, 3 m magas és 1 m széles volt, és 30 tonnát nyomott. Ugorj a 11-re!
29. Jó!! Mind a négy alapművelet elvégzésére alkalmas volt. Számológépének alapja a tízállapotú forgókerék. Sajnos gépének egyetlen példánya sem maradt fenn. Tovább a 9-re!
30. Ebben a korszakban nemcsak a gépek konstruálásával foglalkoztak, hanem megjelentek a programok előhírnökei is. Vajon férfi vagy nő volt az első programozó?

férfi : 3
nő : 13

31. Talált! 800 km hosszú vezetékköteget tartalmazott. Mehetsz tovább a 17-re!
32. Igen. Neumann János dolgozta ki a tárolt programú digitális számítógépek felépítésének elveit, melyek lényegében véve ma is érvényesek (Neumann-elvek).

A végére értél! Remélem, e kis ízelítőtől kedvet kaptál a számítástechnika történetének további tanulmányozásához!!

13.18

1. Kb. hány éves az abakusz?

1 :1 millió 2 :kb. 3000 X : 200

2. Hány év alatt készítette el Jost Bürgi svájci órásmester az első logaritmus táblázatot?

1 : 20 év 2 : 3 év X : 8 év

3. Ki készítette az első „szériában gyártott" számológépet?
1 : Leibniz 2 : Blaise Pascal X : Schickard

4. Hány darabot készített?
1 : 7 db 2 : 30 db X : 100 db

5. Szerinted a francia forradalom ideje alatt hány számjegyes logaritmus táblázatot tudtak készíteni?
1 : 5 2 : 30 X : 19

6. Elkészítéséhez hány számoló szolgát alkalmaztak?
1 : 80 2 : 10 X : 200

7. Kit nevezhetünk az első programozónak?
1 : Ada Byron 2 : Pascal X : Jacquard

8. Mennyi ideig tartott a 10. amerikai népszámlálás?
1 : 3 év 2 : 7 év X : 5 év

9. És a 11.?
1 : 4 hét 2 : 1 év X: 5 év

10. Milyen eszközt használtak a 11. népszámláláson a számlálóbiztosok?
1 : adatlap 2 : lyukkártya X : táblázatok

11 Az ENIAC, azaz Electronic Numerical Integrator and Calculator hány kapcsoló elemet tartalmazhatott?
1 : 300 2 : 6000 X : 1200

12. Az ENIAC 20 szót tudott tárolni és az EDVAC?
1 : 100 2 : 1000 X : 2000

13. Milyen géppel ekvivalens a Kalmár-gép?
1 : Turing-gép 2 : Mark-I X : Z1

+1 A felírt programozási nyelvek közül melyik a legalkalmasabb logikai feladatok megoldására?
1 : Logo 2 : Pascal X : PROLOG

13.2. Operációs rendszer

Az operációs rendszer tanításának feladatai között szerepelnek:

- a számítástechnikai ismeretek legyenek az általános műveltség részei,
- a tanulás-tanítás folyamatában ki kell alakítani a tanulókban a fogalmak helyes, szakszerű értelmezését, tudatos alkalmazását.

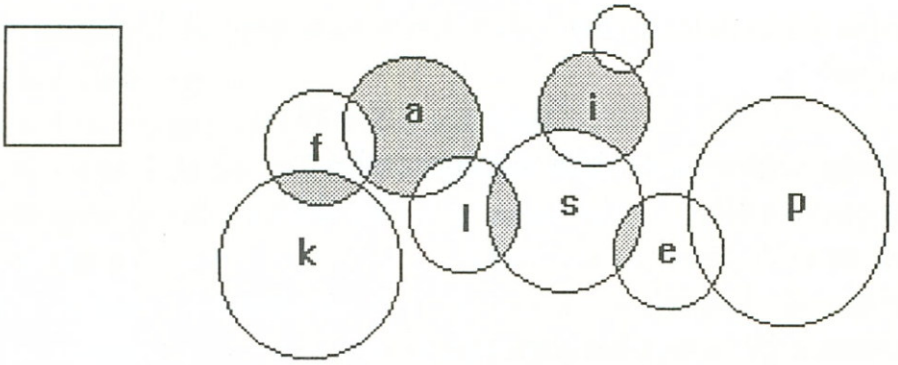
Minimum követelmény a 8. osztály végén:

- Az alapfunkciók céljainak ismerete...

Ezek megvalósításához a következő feladatok is hozzájárulhatnak.

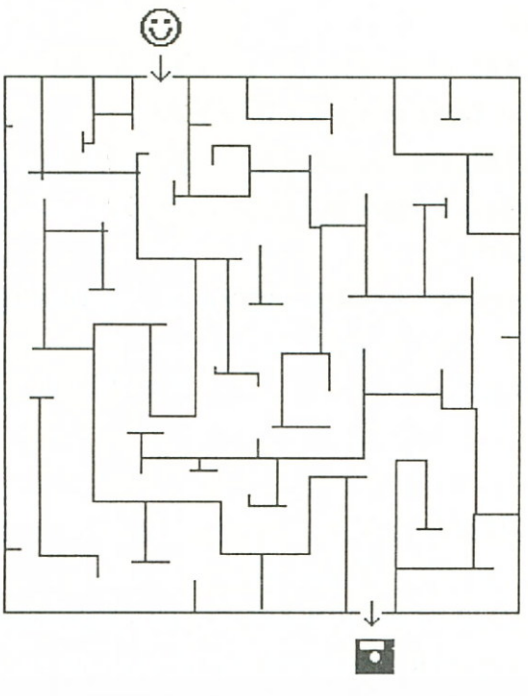
13.19

Állapítsuk meg, mely körök illenek pontosan a négyzetbe! A rajtuk levő betűkből értelmes szó rakható ki!



13.20

Juss keresztül a labirintuson a floppy-hoz!

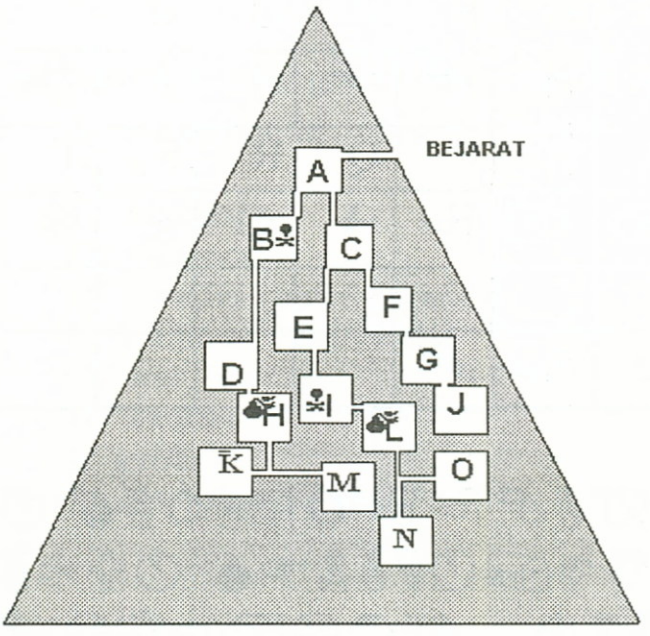


13.21

Az ábra egy piramis rejtett kamráit mutatja. A kamrákat folyosók kötik össze. Egy kutató nagybetűkkel jelölte vázlatrajzán az egyes kamrákat. „Titkosírással" rögzítette, hogy a bejárattól milyen módon juthat el az N-nel jelölt kincstárterembe:

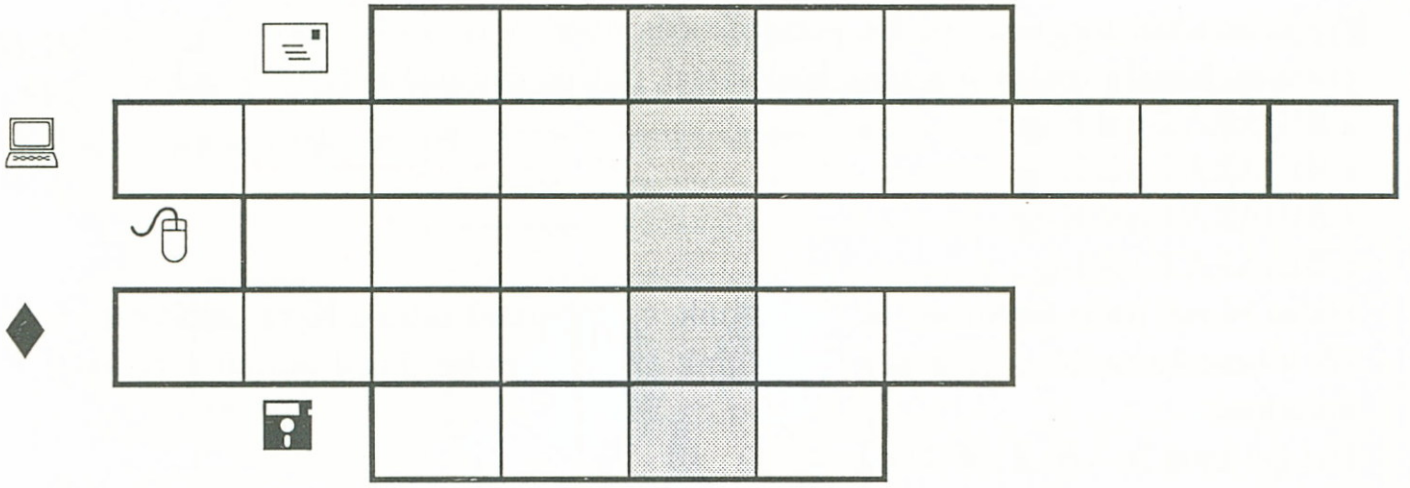
A\C\E\I\L\N

A vázlatrajzot megtekintve kis gondolkodás után Ti is rájöhettek az egyszerű rejtjelre. Ezután írjátok le a fenti kódolással, milyen kamrákon át juthatunk el:
a BEJÁRAT-tól F-ig: :.....
a BEJÁRAT-tól H-ig:.....
a BEJÁRAT-tól K-ig:.....
a BEJÁRAT-tól J-ig:.....
Ha az N-nel jelölt teremben tartózkodunk, s át akarunk jutni a K-val jelölt másik kincseskamrába, vissza kell másznunk az A terembe, majd onnan a K-ba juthatunk.
Ezt így írjuk le: \A\B\D\H\K
Írjuk le jelekkel, hogy juthatunk el:
E-ből D-be:.....
M-ből O-ba:.....
I-ből J-be:.....



13.22

Ügyeskedjük be az ábrába az itt látható dolgok nevét úgy, hogy a satírozott oszlopban egy veszélyes dolog neve alakuljon ki végső megfejtésként! Lehet-e köze az ábrák valamelyikéhez?



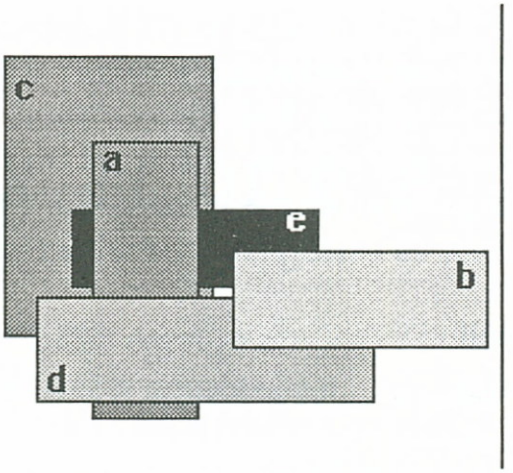
13.23

Fejtsük meg a lenti szöveget, melyet az alábbi kódtáblázattal kódoltunk!

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

13.24

Milyen sorrendben szedhetjük le a következő téglalapokat?



13.25

Kössük össze az idegen szavakat a megfelelő megadott meghatározással!

1. pause	A/ végrehajt, kivitelez
2. abort	B/ fa
3. root	C/ útvonal
4. file	D/ parancs
5. retry	E/ tévedés (téves parancs)
6. tree	F/ leállítani a műveletet
7. path	G/ irat, dosszié
8. fail	H/ szünet
9. execute	I/ gyökér
10. command	J/ újra megpróbálni

13.3 Szövegszerkesztés

A tanítás feladatai között szerepel:

- A tanítás-tanulás folyamatában ki kell alakítani a tanulókban a fogalmak helyes, szakszerű értelmezését, tudatos alkalmazását.
- Nevelje alapos, pontos, megfontolt munkára a tanulókat.

Fejlesztési követelmény:

- Margók, betűtípus, keresés, csere, másolás, áthelyezés.

Minimális teljesítmény a 8. osztály végén:

- A kész dokumentumon végzett módosítás ...

Napjainkban a személyi számítógépek majdnem 60%-a csak valamilyen szövegszerkesztési feladat ellátására használatos. Ezen gépeken kívül több cég gyárt olyan számítógépeket, amelyek speciálisan csak szövegszerkesztésre alkalmasak. A

szövegszerkesztők használatához néhány fontos alapfogalmat ismerni kell. Ezeknek a köznyelvbe való beépülését is elősegítik a fejezet feladatai.

13.26

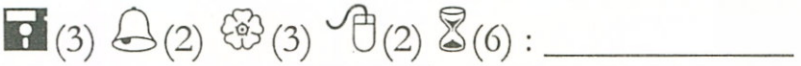
A számsor néhány tagja (valamilyen logika alapján) a többitől eltérő alakú. Egészítsük ki a sort húszig!

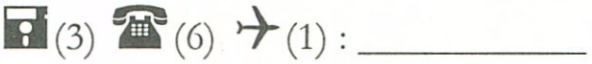
1 2 3 4 5 *6* 7 8 9 10 11 *12* 13 14 ...

13.27

Találjuk ki képek segítségével!

Jelölés:  (4) jelentése ó, mert az olló 4. betűje ó.

 (3) (2) (3) (2) (6) : _____

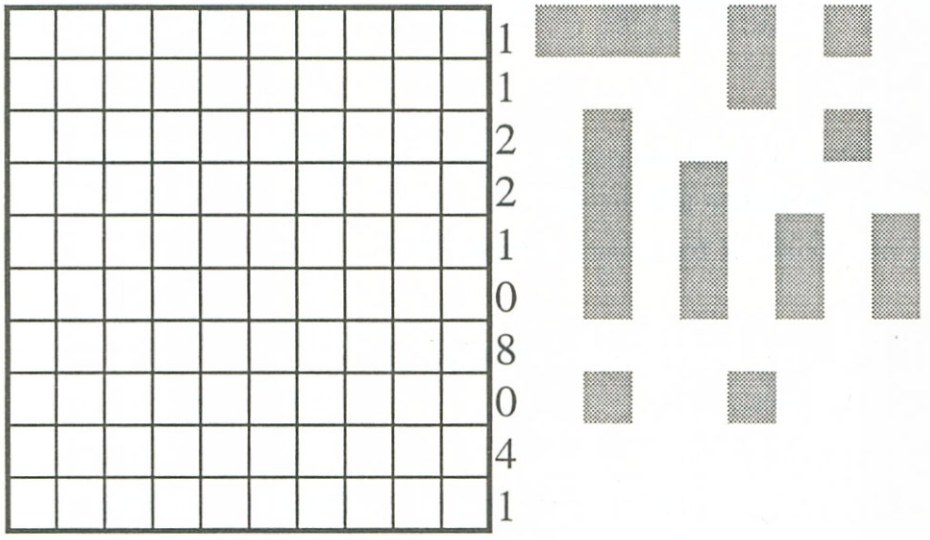
 (3) (6) (1) : _____

 (1) (7) (1) (3) (1) (5) : _____

 (1) (3) (1) (4) (4) : _____

13.28

Bizonyára számodra sem ismeretlen játék a torpedó, bár lehet, hogy egyedül még sohasem játszottad. A lényeg: helyezd el mind a tíz hajót az ábrában úgy, hogy azok sem oldalukkal, sem sarkukkal nem érintkezhetnek egymással! A „tenger" mellett látható számok azt jelzik, hogy az adott sorban vagy oszlopban hány négyzetet foglal el a hajó vagy hajórész. Ha ezzel elkészültél, a következő ábrában jelöld meg azokat a négyzeteket, amelyeken hajók találhatók. Az így kapott betűket összeolvasva a számítógépes szövegfeldolgozás fontos lépését kapod megfejtésül!



V	I	A	H	T	Z	E	I	E	R
H	E	F	S	B	N	L	E	S	C
I	L	E	Y	A	I	G	K	S	I
T	E	Y	S	X	Z	K	I	P	X
E	N	Y	Í	H	E	S	S	V	C
G	H	J	U	L	O	S	A	C	X
R	Á	D	S	E	E	L	L	E	N
C	A	S	D	F	G	J	H	T	R
Õ	A	S	R	A	D	Z	H	N	É
I	N	O	S	A	R	B	E	I	T

13.29

A számozott mondatok egy-egy fogalomnak – a magyar nyelv értelmező szótára, illetve számítástechnika szakkönyvek szerinti – meghatározását adják, de magának a fogalomnak a megjelölése nélkül. A fogalmak megnevezései a meghatározások után – természetesen más sorrendben – egy-egy betűvel jelölve sorakoznak. A feladat az, hogy a megfelelő meghatározáshoz – minél gyorsabban – kiválasszuk a megfelelő fogalmat, és annak betűjelét felírjuk!

1. _____-nak nevezünk egy teljes, összetartozó szövegállományt.
2. A _____ a nyelvnek meghatározott hangulatú és jelentésű, viszonylag önálló egysége.
3. Az írásművek nagyobb önálló egysége a _____.
4. A szövegszerkesztés legkisebb egysége a _____, amely betű, számjegy vagy írásjel lehet.
5. A _____ többsoros, formázható, két ENTER leütése közötti szövegrész.
6. A _____ a beszédnek az a legkisebb, rendszerint többszavas egysége, amelylyel valamit kijelentünk, kívánunk vagy kérdezünk.

A - szó
B - mondat
C - fejezet
D - karakter
E - bekezdés
F - dokumentum

13.30

A feladat célja a függőleges irányú regresszióról való leszoktatás. Lényege, hogy egyszeri elolvasással végezze el tanuló a feladatot. Semmiképpen sem szabad visszatérnie a már elolvasott szövegre, és úgy keresni a megoldást, azaz eldönteni, hogy az egyes szakaszok után következő, a-d betűkkel jelzett 4 kifejezés közül melyik illik a felette álló szöveg kipontozott helyére. Egy-egy számozott mondat vagy szöveg egy-egy külön feladat.

A gyakorlat elvégzéséhez két papírlap szükséges. Az egyikkel a lap alját takarja le, vagyis azokat a gyakorlatokat, amelyek még nem kerültek sorra. A másikkal – azt fentről lefelé tolva – gátolja meg, hogy szeme önkéntelenül visszaugorjon a már elolvasott szövegre. A négy kifejezést, melyekből a megfelelőt kiválasztja, letakarni nem kell.

Feladat:

A. Írott szövegeink – még külalakjuk is – énünket tükrözik. _____ érvényre jutása, hatása elsősorban attól függ, hogyan vannak azok felöltöztetve, formába öntve.

a/ városunk b/ gondolataink c/ akaratunk d/ szívünk

B. A stílus arra való, hogy valamit _____ közöljünk, és ne ködösítsünk. Az utóbbi célra legfeljebb politikusok és diplomaták használják fel. Ezért olyan köntörfalazó az ő nyelvük.

a/ erősen b/ feketén c/ világosan d/ szilárdan

C. Ne akarjuk műveltségünket idegen szavak halmozásával fitogtatni még akkor se, ha valójában beépültek szókincsünkbe. Csak akkor használjuk őket, ha olyan jelentéstöbbletet hordoznak, amelyek _____ megfelelőjükből hiányoznak, vagy az általuk létrehozott hangulatot szándékosan akarjuk előidézni.

a/ magyar b/ kevés c/ nem d/ jó

D. Tudnunk kell, hogy mit akarunk elmondani! Nem is csak úgy nagyjából, abban a reményben, hogy _____ közben majd úgyis minden kitisztul. Ez így ritkán sikerül. Minél inkább tudjuk, hogy mit akarunk, annál világosabbá válik a szöveg.

a/ nap b/ járás c/ vezetés d/ írás

E. A stílus akkor jó, ha fennakadás nélkül olvashatjuk. Ha egy szövegben botorkálni kell, vagy ha félreérthető, akkor a stílus rossz. Ugyanez a helyzet akkor, ha _____ is el kell olvasni egy-egy mondatát, amíg végre megértjük.

a/ néha b/ háromszor c/ utoljára d/ egyszer

F. Olyan szöveget olvasni, melyben nyolc _____ át egyetlen új bekezdés sincs, kínszenvedés. Az olvasó szinte kifullad tőle. Ez ellen jók a bekezdések.

a/ betűn b/ könyvön c/ oldalon d/ soron

13.31

A tanuló nyelvi fantáziáját, képzelőerejét és gondolkodásának rugalmasságát egyaránt próbára teszi a következő gyakorlat. Részletet olvashat a szövegszerkesztéssel kapcsolatban, azonban a szöveg csak úgy válik érthetővé, ha kiegészíti a hiányos szedést. Minden pont helyére egy betűt kell írnia, így válik a szöveg érthetővé. Könnyebben megy a munka, ha nem az egyes szavakat próbálja kitalálni, hanem figyelembe veszi a szövegkörnyezetet.

Feladat:

Egy f_nto_ elv_rá_ a szöv_gr_szekk_l el_ége_he_ő m_vel_tek_e vo_atk_zik. A sze_kesz_őnek tu_nia k_ll egy ki_el_lt szöve_részt tö_ö_ni, áth_lye_ni, át_ás_lni. Az u_ó_bi két mű_ele_et legt_bb ese_ben az ún. zseb (scrap) hasz_álat_val ol_ják meg. Az át_elye_ésnél a zs_bbe át_er_l a szö_eg, az ere_eti hely_ről törl_dik, míg a má_olás_ál az er_deti is me_marad. Ez_tán az eltá_olt sz_veg tetsz_le_es számú al_alo_mal be_llesz_hető a zs_bből a dok_men_umba. A zseb tar_alm_t egy új_bb zse_be hely_zett szöv_g t_rli ki.

13.4 Adatbázis- és táblázatkezelés

A tananyagban szerepelnek:

- Egyszerű keresési feladatok.
- Kapcsolatok keresése táblázatokban levő adatok között.

Minimális teljesítmény a 8. évfolyam végén:

- Csoportosítás, rendszerezés.

Az e témában szereplő feladatokban, játékokban egyrészt a kapcsolódó fogalmak ismeretére építünk, másrészt csoportosítást, rendszerezést, fő jellemző tulajdonságok kiválasztását gyakorolhatjuk. A játékok felhasználhatók bármely tanórán pihentetőül már alsó tagozatos kortól, persze a korosztálynak megfelelően.

13.32

Bizony nem könnyű az út a starttól a célig, de neked biztosan sikerül megtalálnod! A játék egyszerű: a bal felső sarokból indulva, valamennyi mezőt egyszer érintve (rálépve) el kell jutni a jobb alsó sarokig úgy, hogy közben mindig csak a nyilak által megszabott irányban haladhatsz, nem feltétlenül a szomszédba. (Ránézésre talán nem látszik, melyik lesz az első mező, amelyre lépned kell, de abban biztos lehetsz, hogy a baloldali oszlopban kell maradnod, hiszen a startmezőben levő nyíl lefelé mutat.) Ha a sorrendet sikerült meghatároznod, már csak ugyanezt kell tenned az alsó ábrában levő betűkkel is. És máris kialakul a számítógép egyik fontos funkciója.

Start ↓	→	↓	←
→	↑	↓	←
→	→	↑	↑
→	→	←	Cél

Start ↓	L	O	D
F	E	G	T
A	D	L	A
Z	Ó	O	Cél

13.33

Ez már komolyabb előképzettséget igényel a tanulóktól! A feladat az, hogy a szövegben a megfelelő helyre a megfelelő fogalom kerüljön, azonban több szó van megadva, mint amennyire szükség van.

Rendszerint élőlények, vagy tárgyak, vagy fogalmak _____ait gyűjtjük össze az adatállományban. S egyetlen élőlény, vagy egyetlen tárgy, vagy egyetlen fogalom adatai mindig összetartoznak (pl. személynél a neve, telefonszáma, születési helye, anyja neve stb.), és ezért ezek együtt egy _____ban kezelendők az állományon belül. Az adatállomány minden rekordja _____kből áll, mert a

rekord mezőiben vannak az illető élőlény, vagy tárgy, vagy fogalom adatai (pl. a személy neve). Több, valamiképpen összetartozó _____ alkotja az _____t. Ezekből választhatsz: vonal, adat, eszköz, rekord, adatállomány, kód, adatbázis, mező.

13.34

Játsszunk titkosírást a következő táblázat segítségével!
Én így írnám le:

A3|F1|B2|F5|G1|A3||A1||C1|A1|C4|B1|E4|D3|B3||
LEGYÉL A BARÁTOM

Ide mit írtam?
-

C1|A1|C4|B1|E4|D3|B3||A1||D4|G5|B1|B3|E2|E4|E3|B2|G 1|A4||

Írjatok üzeneteket és válaszoljatok egymásnak!

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

MEGOLDÁSI JAVASLATOK

7. FEJEZET

SZAKTÁRGYI FELADATOK

7.1

Meg kell állapítanunk, hogy van-e a számnak valódi osztója. Ne felejtsük el: elegendő a szám négyzetgyökéig vizsgálódnunk! Sok időt takaríthatunk meg azzal is, ha – amennyiben a szám nem páros – csak a páratlan osztókat keressük!

7.6

Vegyük észre, hogy minden osztónak van egy (tőle általában különböző) párja is!

7.9

Vegyük észre, hogy a négyzetszámok prímtényezős felbontásában minden kitevő páros!

7.10

Egy kevésbé ügyes megoldás: az osztókat meghatározzuk, majd mindegyikről egyenként eldöntjük, hogy prím-e vagy sem. Sokat segít, ha észrevesszük: bármelyik osztó osztói (ha vannak) csak a többi közül kerülhetnek ki, így – ha ezek között nem találunk – prímosztóval van dolgunk.

7.11

Ezen azt értjük, hogy az adott prímszámnak (maximálisan) hányadik hatványával osztható az adott szám. Más szóval – és ez egy lehetséges út az algoritmus elkészítéséhez –: ha a számot a prímszámmal osztjuk, a kapott hányados továbbra is osztható lesz-e ezzel a prímszámmal (s ha igen, ezt hányszor tudjuk egymás után elvégezni)?

7.17

Csak azokat a számokat érdemes továbbvizsgálni, amelyek osztóinak összege egyik számnál sem nagyobb!

7.19

Használjuk fel, hogy A és B közös osztói osztói a különbségüknek is!

7.22

A (p, p+2) számpár vizsgálatából (melyik volt prím) következtethetünk a következő megvizsgálandó számpárra!

7.26
Pitagoraszi számhármasnak azt a három pozitív egész számot nevezzük, amelyek valamelyikének négyzete megegyezik a másik kettő négyzetösszegével, más szóval derékszögű háromszög szerkeszthető belőlük.

7.27
A megoldás: $(...(A_1*x+A_2)*x+...+A_n)$

7.32
Ismétléses permutáció: számoljunk el tízes számrendszerben 0000000000-tól 9999999999-ig!
Ismétlés nélküli permutáció: hagyjuk el ebből azokat a számokat, amelyekben egy számjegy kétszer fordul elő!
Elképzelhető az utóbbi feladat megoldására a visszalépéses keresés (backtrack) alkalmazása is.

7.37
Ügyeljünk a túlcsordulásra! A számítási sorrend nem közömbös!

7.38
Gondoljuk meg, milyen összefüggések igazak a Pascal-háromszögre:
- minden elem a fölötte levő két elem összege,
- minden elem megkapható a tőle balra levőből egy szorzás és egy osztás segítségével.

7.39
A kezdőknek a binomiális együtthatók segítségével való megoldást javasoljuk, az edzettebbek próbálkozhatnak rekurzióval!

7.49
Keressük meg azokat a pontpárokat, amelyeken áthaladó egyenes úgy vágja a síkot két félsíkra, hogy ezen két félsík egyike (belsejében vagy határán) tartalmazza az összes pontot!

7.57
Egy szakaszt kapunk.

7.59
Az $y(0)>0$ közelítő értékből kiindulva az

$$y(i+1):= (y(i)+N/y(i))/2$$

képlet segítségével egyre pontosabb közelítést kaphatunk.

7.61
Az

$$y(i+1):= y(i)+(N/y(i)/y(i)-y(i))/3$$

sorozat N köbgyökéhez tart.

7.63

A 4. módszer nem adja meg minden esetben a megoldást. Gondoljunk pl. az f'(x(I))=0 esetre! Részletesebben megtalálható például Bacskay Zoltán: Közelítő számítások c. középiskolai szakköri füzetében (Tankönyvkiadó, 1972.).

7.68

Próbáljuk formalizálni azt, ahogyan kézzel végezzük az alapműveleteket! A kézi számolás minden részeredményének is feleltessünk meg egy-egy változót!

7.81

A grafikonok felrajzolásához egy általánosan is használható keretprogram készítését ajánljuk! Legyen benne egy y=f(x) típusú sor, amelyikben a függvényt adjuk meg! A képernyőn kívüli pontokat ne is kísérelje meg felrajzolni, hogy elkerüljük a hibajelzést! Ezt tovább bővíthetjük azzal, hogy a kívánt értelmezési tartományon belül a program számítsa ki a függvényértékek minimumát és maximumát, és úgy nyújtsa vagy zsugorítsa a grafikont, hogy minden pont a képernyőre kerülhessen, és azt a lehető legjobban használja is ki!

7.88

A láncszemek számát vegyük párosnak, és csak a lánc egyik felére figyeljünk! Ilyenkor a középső szemet egyik oldalról vízszintesen húzza a szomszédja, a másik oldalról pedig akkora erővel, hogy annak függőleges vetülete egyensúlyozza ki a nehézségi erőt. Kiindulási adatnak vegyük fel a fal és a lánc csatlakozásának szögét!

7.89

Dolgozzunk változó időtartamokkal, és figyeljük külön-külön az összes láncszem mozgását! Minden láncszem lecsúszása után ki tudjuk számítani az egyes szemekre ható erőket, ebből a gyorsulást, a sebességet és az új helyet. Ha ez kész, vehetjük a következő időtartamot.

7.90

A probléma: minden egyes kilövellésnél az üzemanyag megmaradt részét is gyorsítani kell!

7.95

A sebességvektor irányába történő gyorsítás távolodást jelent, s ettől csökken a sebesség. Ezzel körpályáról ellipszispályára térhetünk át. Erről újabb körpályára úgy térhetünk át, ha egy ponton újra gyorsítunk.

7.139

Az egyes vércsoportok a következő genotípusokhoz tartoznak:

A: aa, a0
B: bb, b0
AB: ab
0: 00

7.159

A genotípus – fenotípus megfeleltetés:

genotípus	fenotípus
aa a0 0a	A
bb b0 0b	B
ab ba	AB
00	0

7.165

A következő változások hatását kell számítani és ábrázolni:

$$N' = b * N - d * N,$$

illetve b helyébe b0-k*N, d helyébe d0+l*n kerül.

N a populáció létszáma, N' az egy időegység alatti változás, b a születési, d a halálozási ráta, amelyek logisztikus növekedésnél a létszámtól függnek.

7.172

A ragadozó- (R), illetve a zsákmánypopuláció (Z) változását a következő egyenletek írják le:

$$R' = (B_1*Z-D_1) * R ,$$
$$Z' = (B_2-D_2*R) * Z ,$$

ahol B_i, D_i a megfelelő populáció születési, illetve halálozási rátája.

7.179

Ha a szimuláció során A-t és annak B szomszédot választottunk, A helyére kerüljön C, B helyére semmi! Ha C-t választottunk és van üres szomszédja, akkor C helyére kerüljön A, az üres helyre pedig B! Minden más esetben csak cseréljük ki a két mező tartalmát!

7.192

Az összefüggőség-vizsgálathoz induljunk ki egy atomból, jelöljük meg azokat, amelyekkel kötése van, jelöljük meg azokat, amelyeknek ezekkel kötése van, ...! Ha így az összes atomot megjelöltük, akkor összefüggő szerkezetről van szó. A 6. részfeladat megoldásánál hagyjuk el a molekulából az 1 kötéssel rendelkező atomokat! Ezután a maradékban újra keletkezhettek 1 kötésű molekulák, ezeket is hagyjuk el! Mindezt addig ismételjük, amíg találunk 1 kötésű molekulát! E folyamat végén éppen a keresettek maradtak meg, ha voltak egyáltalán.

7.202

Bonyolult megoldani, hogy melyik utas hol száll ki. Elképzelhető például egy annyi elemű vektor, ahány utas lehet a liftben, s egyes elemei az utasok célemeletét tartalmazzák. Újabb problémát okoz, ha a lift útközben megáll, s a kiszállók helyére újabb utasok szállnak be. Gondoljuk meg azt is, ha több helyről hívják, akkor hova menjen a lift!

7.205

Tároljuk az autók koordinátáit, valamint az út egyes helyeinek állapotát (van ott autó vagy nincs)! Így egyrészt könnyen kezelhetjük az autókat, másrészt egyszerű vizsgálni, hogy van-e előtte autó vagy sem.

7.208

Egyszerűbb a feladat, ha adott egy menetrend a gyors-, illetve a személyvonatokról, amely garantáltan helyes (ilyen pld. bármelyik MÁV-menetrend), s csak a tehervonatokat kell úgy indítani, hogy ez ne ütközzön a menetrenddel! Bonyolultabb esetben a személyvonatok menetrendjét is módosítani kell!

7.209

Nézzük meg minden vagonra azt, hogy melyik irányba kell küldeni! Ennek alapján állítsuk be valamelyik szerelvénybe! Ha egy szerelvény összeállt, s van hozzá mozdony is, akkor indítsuk el! Az azonos irányba továbbítandó kocsik összekapcsolásakor már eleve vegyük figyelembe, hogy a szerelvény elejére a legtávolabb menő, a végére pedig a leghamarabb leakasztandó kocsi kerül!

7.210

A mozgólépcsőknek feleljen meg egy-egy vektor (0, 1 vagy 2 ember állhat minden lépcsőfokon)! Egy időegység alatt a lépcső 1 lépcsőfoknyit mozduljon el (a vektorok léptetése), s ehhez igazítsunk minden más történést!

7.212

Egyszerű esetben (2 kifutópálya) az egyik kifutópályán csak leszállás, a másikon csak felszállás legyen! Ha van harmadik is, akkor azt felváltva használhatjuk fel-, illetve leszállásra. Elképzelhető azonban az is, hogy ha éppen nincs felszálló gép, akkor az összes pályát leszállásra használjuk.

7.215

Ötletet a fizika alfejezet harmadik típusú feladatainál találunk.

7.216

Ötletet a fizika alfejezet harmadik típusú feladatainál találunk.

7.241

Ezt a feladatot szándékosan nem fogalmaztuk pontosan, tekintettel a megoldók különböző fokú felkészültségére. A probléma körvonalazása viszont remélhetőleg mindenkinek elegendő ahhoz, hogy ki-ki idejének és fantáziájának függvényében belevágjon ebbe az érdekes „játékba", s számos új paraméterrel is bővítve, érdekes szimulációs programokat készítsen.

7.251

Az izotermatérképen az azonos hőmérsékletű pontokat kötik össze vonalak. Látványos egy olyan program, amely az így kapott vonalak közötti minden második sávot fehérre, minden másodikat feketére fest.

7.254

A felhős területek nagy fényességűek a többi ponthoz viszonyítva, hiszen a felhő a ráeső napfény nagyon nagy részét veri vissza.

7.264

Bontsunk rövid időtartamokra egy periódust, és számítsuk ki a pillanatnyi teljesítmény grafikonja alatti területet!

7.265

Tegyük fel, hogy hővezetéssel adja le az energiát, ezért a hőmérséklete közelítőleg arányos a teljesítménnyel!

7.272

A tartó lehajlása egy tetszőleges x pontban

y =F*(2h^3- 3xh^2+x^3) /(6EI),

ahol h a tartó hossza, y a lehajlás, E és I állandók, x a szabad végtől való távolság.

7.276

A sebesség függ a viszkozitástól, a sűrűségtől, a nyomáskülönbségtől, a súrlódási tényezőtől, a csőátmérőtől.

8. FEJEZET

AZ ISKOLAI ÉLET SZERVEZÉSE

8.1
Olyanra gondolunk, amely automatikusan szétosztja a felmenő rendszer szerint az így eloszthatő órákat. Kiírja ezután, kinek hány órája van, és ezt a továbbiakban is mindig megteszi. Nyomon követi a túlórák számának alakulását.

8.2
Kikeresi, hogy az adott tantárgyat oktatók közül kinek van a legkevesebb órája, ki tanít párhuzamos osztályban stb.

8.5
Igyekezzünk minél memóriatakarékosabb megoldást választani!

8.6
A program fejleszthető úgy, hogy óracserével történő helyettesítés esetén, úgy választja ki a tanárt, hogy tőle könnyen vissza lehessen venni az órát. Esetleg meg is mondja, melyik nap melyik órájában.

8.9
Egyszerűbb esetben csak egy szakmai előadó beosztását készíttessük el a géppel!

8.10
Első lépésként készítsünk egy terembeosztást nyilvántartó programot!

8.12
A program az igazgatók munkáját segítheti, különösen nagyobb létszámú tantestület esetén.

8.13
A program két részből áll. Először kiszámítja, hogy az adott fizetésemelés milyen adósávba esik, majd az ezzel a százalékkal csökkentett értékeket rendezi az adott arányban.

8.16
Első lépésként javasoljuk egy olyan program elkészítését, amely nyilvántartja, ki hányszor ügyelt már az év folyamán. A következő lépésként a számítógép elsősorban ez alapján jelöli ki az éppen szükséges felügyelőket.

8.17
A feladat megoldható valamilyen adatbáziskezelő felhasználói program segítségével is.

8.18
Figyeljünk azokra a szokásos esetekre – bukás, csoportbontás, átmenet másik csoportba, évközi kimaradás stb. – amelyek gyakran előfordulhatnak az iskolában. Legyen nagyon áttekinthető és egyszerű a kezelés, hiszen bizonyára nem számítástechnikai szakember fogja használni!

8.19
Célszerű a programot alkalmassá tenni a következő feladat kiszolgálására is.

8.22
Itt a nyilvántartáson túl elsősorban a rendszeres családlátogatások nyilvántartására, azok esedékességére való figyelemfelhívás megoldására gondoltunk.

8.23
Ez az egyik olyan program, amelyet a tanulók írhatnak, használhatnak. Így egyszerű feladaton keresztül ismerhetik meg a számítógép gazdasági felhasználását.

8.24
Javasoljuk, hogy a számítógép nyomtasson levelet a szülőknek, amelyben az iskola értesíti őket a fizetendő díjakról. Természetesen jó, ha ugyanez a program a kollégiumi díjak megállapításában és ismertetésében is tud segíteni.

8.32
Gondoljunk arra, hogy a feladat megoldásához egy folyamatosan karbantartható (új eszközök, új kísérletek) adatbázis célszerű lenne.

8.36
Adjunk meg feltételeket! Például meghatározott feleletszám után mindenkinek legyen egy megadott minimális számú jegye. A felelő kiválasztása azonban mindig véletlenszerű maradjon!

8.37
Úgy gondoljuk, hogy e program segítségével olyanok is tudnak ilyen feladatlapot készíteni, akik egyébként nem tudnak programot írni.

8.40
Tulajdonképpen a „tömbök" bevezetésére szolgáló feladat, csak tartalma miatt ismertetjük itt is.

8.41

Alapesetben arra gondoltunk, hogy előreülteti a szemüvegeseket és az alacsonyakat, külön a fiúkat és a lányokat. Célszerű azonban lehetővé tenni más feltételek megszabását is. Egy lényeges javítása a programnak, ha lehetővé tesszük, hogy a tanulók választhassanak padtársat.

8.42

A lakhelyeket egy képzeletbeli térkép pontjainak koordinátáival adjuk meg. Két tanuló között a távolság függ attól, hogy telefonálhatnak-e egymásnak, milyen távolságra laknak busszal, vonattal stb. A riadólánc hosszát is vegyük figyelembe!

8.49

Tételezzük fel, hogy a számítógép állandóan üzemel. A tanuló rendszeres interaktív kapcsolatot tart a géppel. A gép a beépített óra által vezérelve, állandóan számon tartja a tanuló napi teendőit. Például: reggel ébreszti, figyelmezteti, mit kell aznap iskolába vinnie; jelzi, ha el kell indulnia; délután beosztja a házi feladat elkészítését; időben szól, ha egy kedvenc tv-műsor kezdődik stb. Nehéz, de megoldható, igen izgalmas feladat. A tanulók nagy lelkesedéssel dolgoznak rajta.

8.51

A tanulmányi mozgalom olyan iskolai verseny, amelyben az osztályok igen sok szempont alapján versenyeznek egymással. A különböző szempontok más-más súllyal veendők figyelembe. Általában külön hirdetnek összesített, illetve területenkénti győzteseket.

8.52

Ha a csapatok száma páratlan, akkor vegyünk fel egy fiktív csapatot, s ha valaki ezzel játszana, akkor szabadnapos. Páros csapatszám esetén egy lehetséges sorsolás:

Csapatszám	Mérkőzések					
	1.	2.	3.	...	N-2.	N-1.
1	2	3	4	...	N-1	N
2	2	N	N-1	...	4	3
3	4	3	2	...	6	5
4	6	5	4	...	8	7
.						
.						
N/2+1	N	N-1	N-2	...	3	2
N/2+2	3	2	N	...	5	4
N/2+3	5	4	3	...	7	6
.						
.						
N	N-1	N-2	N-3	...	2	N

Cseréljük 1-re azokat a táblázatelemeket, amelyek a fenti kiírás szerint a két ellenfelet egyazon csapattal jelölik ki, így megkapjuk az egyik lehetséges versenykiírást!

8.55

A program továbbfejleszthető úgy, hogy a diákok választhatnak szobatársat. Készíthetünk hozzá olyan kiegészítést, amely szobanévsorokat nyomtat az ajtókra, amellyel kereshetünk valakit név szerint, osztály szerint stb.

9. FEJEZET

GÉPKÖZELI FELADATOK

9.1
A megoldás a Horner-elrendezés:
(...(Kimenet(8)*2+Kimenet(7))*2+...+Kimenet(1))
vagy ciklusban X=X+X+Kimenet(I)

9.2
Az eredmény: (X AND 32)/32, illetve (X & 32)/32.

9.6
A Pascal az AND és az OR, illetve a C az & és a | műveletet bitenként végzi el. (X OR 3), (X AND 252) illetve (X | 3), (X & 252).

9.7
A Pascal-beli `IF`, `CASE`, illetve a C-beli `SWITCH`, -, `AND` felhasználásával, vagy T[3]=2; T[2]=3; X=T[X].

9.9
A számot inkrementáljuk, utolsó két bináris helyiértékét vágjuk le az AND, illetve az & művelettel!

9.10
Használjuk fel az előző két feladat eredményét!

9.11
A szükséges bitkombinációk egyeznek az előző feladatéval, csak a sorrend pont fordított.

9.15
Lehetséges megoldás:
Sárga1=Sárga2=Kimenet[2]; Piros1=Kimenet[1]; Piros2=Kimenet[3]; Zöld1=Kimenet[3] ÉS NEM Kimenet[2]; Zöld2=Kimenet[1] ÉS NEM Kimenet[2].

9.16
A módosítás: Kimenet(1)=NEM Kimenet(3).

9.17
Használjuk a PC hangszórójának időzítőjét, vagy karórával hitelesítsük egy ciklus futási idejét! Ha van lehetőség rá, mindkét eljárást próbáljuk ki több számítógépen is!

Pascal megoldás:

```
Procedure Idozit(tartam: integer);
  Var
     ido, allapot: byte;      {tartam [mms]}
  Begin
    Port[$61]:=$01;            {idôzítô engedélyezve}
    Port[$43]:=$b0;            {üzemmód, LSB-MSB
sorrend}
     For ido:=0 to tartam-1 Do Begin
       Port[$42]:=$a9;         {$04a9=1193 -> 1 ms}
       Port[$42]:=$04;
       allapot:=$30;
       While (allapot=$30) Do Begin
         Port[$43]:=$E8;       (ha lejár,$B0-ra vált}
         allapot:=Port[$42];
       End;
     End;
  End;
```

C megoldás:

```
#include <dos.h>
  void Idozit (unsigned int tartam)      /*tartam
[ms]*/
{
   unsigned int ido, allapot;
   outp (0x61, 0x1);        /*idôzítô engedélyezve*/
   outp (0x43, 0xb0);       /*üzemmód, LSB-MSB sorrend*/
   for (ido=0; ido < tartam; ido++){
      outp (0x42, 0xa9);     /*0x04a9=1193 -> 1ms*/
      outp (0x42, 0x04);
      allapot=0x30;
      while (allapot == 0x30) {
                             /*ha lejár, 0xb0-ra vált*/
        outp (0x43, 0xe8);
        allapot=inp (0x42);
      }
   }
   return;
}
```

9.18

Az időzítő ciklusban figyeljük, van-e lenyomott billentyű! Ha van, dolgozzuk fel, és próbálgatással állapítsuk meg, mennyivel kell változtatni a ciklusváltozó értékét! Az előző feladathoz csatolt program lehetőséget ad az időzítéssel párhuzamosan más tevékenység elvégzésére, de azt az időzítő lejárta előtt be kell fejezni.

9.19

A sötétről világosra váltások számát érdemes számolni, hiszen különben egy nagyméretű munkadarab elhaladását többször is számlálnánk.

9.20

Stopperrel hitelesítsük egy ciklus futásának idejét; használjuk a 9.17-es feladat időzítőjét!

C megoldás:

```
#include <dos.h>
unsigned int Idomero()
{
   unsigned int i;
   i=0;
   while (kbhit () == 0) {
      outp (0x61, 0x1);
      outp (0x43, 0xb0);
      outp (0x42, 0x96);
      outp (0x42, 0x04);
      outp (0x43, 0xe8);
      while (inp (0x42) == 0x30) outp(0x43, 0xe8);
      i++;
   }
   return (i);
}
```

9.21

A két sebesség az első 1/2 cm, illetve a második 1/2 cm átlagsebessége. Egyenletesen gyorsuló (lassuló) mozgás esetén az ilyen sebességű pontok éppen 1/2 cm-re vannak egymástól.

9.23

Az ingára tegyünk olyan széles árnyékvetőt, hogy a pálya alsó pontjában a fotokapun való áthaladás sebességét már elég pontosan tudjuk mérni, de az áthaladás alatt a sebesség még ne változzék jelentősen! Az áthaladás idejéből és a zászlócska hosszából a sebesség kiszámítható, az egymás utáni áthaladások között eltelt idő a fél lengésidő.

9.25

Figyeljünk egy áthaladás esetén a kapuba való belépés és a kapuból való kilépés pillanatára! Állapítsuk meg, honnan jött és merre ment a látogató, hiszen akár vissza is fordulhat a kapuban!

A program végig is követheti a két érzékelő minden lehetséges állapotát az áthaladás során. Ez a nehezebb megoldás, de érdemes egyszer megpróbálni, hogy hogyan feleltethető meg a valós rendszer „állapottere" a program állapotainak. Itt azt is figyelembe kell venni, ha a látogató a kapuban visszafordul (akár többször is), hol teszi azt.

9.28

A program indításkor vegyen mintát a bemenetről, és ehhez hasonlítsa az állapotokat!

9.29

Itt nem használhatjuk a 9.17-es feladatban leírt időzítőt, mert azt már a hang keltésére lefoglaltuk.

9.32

Ha csak az egyik motor van bekapcsolva, a jármű a másik lánctalp közepe körül fordul, tehát a középpontja halad előre.

9.39

C megoldás:

```
#include <dos.h>
void Abetu()
{
   unsigned int sor, oszlop;
   _AX=0x0003;          /*00 üzemmódváltás, 03 üzemmód*/
   __int__(0x10);       /*interrupt-hívás*/
   for (sor=0; sor < 24; sor++){
      for (oszlop=0; oszlop < 80; oszlop++){
         _AH=0x02;      /*kurzorpozicionálás*/
         _DH=sor;
         _DL=oszlop;
         _BX=0x0000;    /*0-dik képernyô*/
         __int__(0x10);
         _AH=0x09;      /*karakterírás*/
         _AL=0x41;      /*A betű*/
         _BL=0x01;      /*attributum*/
         _BH=0;         /*lapszám*/
         _CX=0x0001;    /*1-szer*/
         __int__(0x10);
      }
   }
   while (getch () != 13);    /*tovább = ENTER*/
}
```

9.42

C megoldás:

```
#include <dos.h>
void Pozi (unsigned int s, unsigned int o)
{
   _AH=0x02;
   _DH=s;
   _DL=o;
   _BH=0;
   __int__ (0x10);
}
void Ir (char k, char a)
{
   _AH=0x09;
   _AL=k;
   _CX=1;
   _BH=0;
   _BL=a;
   __int__ (0x10);
}
void Tukroz ()
{
   unsigned int sor, oszlop;
   char kar[2], attr[2];
   for (sor=0; sor < 24; sor++){
      for (oszlop=0; oszlop < 40; oszlop++){
         Pozi (sor, oszlop);
         _AH=0x08;           /*az 1. karakter olvasása*/
         __int__ (0x10);
         kar[0]=_AL;         /*az 1. karakter mentése*/
         attr[0]=_AH;
         Pozi (sor, 79-oszlop);
         _AH=0x08;           /*a 2. karakter olvasása*/
         __int__ (0x10);
         kar[1]=_AL;         /*a 2. karakter mentése*/
         attr[1]=_AH;
         Ir (kar[0], attr[0]);
         Pozi (sor, oszlop);
         Ir (kar[1], attr[1]);
      }
   }
   while (getch () != 13);     /*tovább = ENTER*/
}
```

9.44

Változtassuk meg a karakterek attribútumát magának a karakternek a változtatása nélkül!

9.46
Egyik regiszterből toljuk, illetve forgassuk a másikba, majd töltsük vissza az eredetibe!

9.48
Használjuk. a 9.42 feladat megoldásának rutinjait!

9.58
Egy memóriatartományt kezeljünk úgy, mint a képernyőt, ez tartalmazza a teljes grafikont! Mindig az aktuális területről töltsük fel a képernyőt! A nagyítást elvégezhetjük a sorok és oszlopok kétszerezésével, illetve a korábbi elemek átlagának beszúrásával. A kicsinyítésre is kétféle lehetőségünk van: minden második sor, illetve oszlop kihagyása, vagy pedig 2x2-es részek helyettesítése az átlagukkal.

9.60
Keressük a képernyő oldalegyeneseivel való metszéspontokat! Gondoljuk meg, mikor nincs biztosan(!) a képernyőn az egyenes egyetlen pontja sem!

9.61
Közelítsük sokszöggel, vagy használjuk a kör különböző definícióit, vagy közelítsük pontonként!

9.64
Használjunk veremmemóriát vagy írjunk rekurzív programot!

9.66
Egyszerű, ha azonos méretű részeket cserélgetünk, gyorsabb és helytakarékosabb, ha csak a változó részt írjuk újra.

9.67
Azonos hullámhosszú, időben szinuszosan változó kitérésű színuszhullámokat kell a képernyőn előállítanunk az előző feladathoz hasonlóan.

9.69
A program akkor látványos, ha a pont mozgatása közben a szakasz is mozog.

9.77
A biteket hármasával léptessük egy regiszterbe!

9.87
Minden számjegy bináris helyiértéke eggyel nő, azaz sifteljük a számot balra, az utolsó bitet állítsuk 0-ra!

9.88
10=2+8, tehát adjuk össze a szám kétszeresét és nyolcszorosát! Még jobb eljárást írhatunk annak alapján, hogy 10=2*(4+1).

9.89
Vonjunk ki először 10000-et, amíg csak lehet. Ennyiszer van meg benne a 10000. Ezt a számot szorozzuk meg 10-zel, s adjuk hozzá, hogy az osztandó maradékából hányszor lehet kivonni 1000-et. Így megkapjuk, hogy hányszor van meg benne az 1000. Ezt folytassuk egészen 10-ig!

9.91
Gondoljunk arra, hogy az ember nem gép, és nem tudja az időtartamokat pontosan kimérni, tehát engedjünk meg néhány százalékos szabálytalanságot is! Megpróbálhatjuk a jelek időtartamának tárolását is, és csak utólag, az összes időtartam ismeretében, az egyéni sebesség figyelembevételével dekódoljuk őket.

9.94
B=(A+1)^2, C=2*A+1, A=X négyzetgyökének egészrésze.

9.98
Az 1407 alakú oktális vagy decimális számot írja ki.

9.99
A feltételregiszter CARRY-bitje 1 lesz, ha CL nem osztója DL-nek, s 0, ha igen.

9.100
68000 nem fér el 2 bájton, végtelen ciklus MOV AL, 5 miatt.

9.101
BX-be rosszul tölt, SBB előtt CARRY=?, BX nem a maradék.

9.102
DX-et binárisan kiírja.

9.103
DX a beírt szám binárisan.

9.104
Az első DEC SP utasítás címe. AX=szegmens, SI=offset

10. FEJEZET

PROGRAMELEMZÉS

10.1
Tetszőleges A, B, C értékre kiírja, hogy EGYENLOK.

10.2
Z = X * Y értékét határozza meg.

10.3
Z-ben X-nek Y-adik hatványát állítja elő.

10.4
A = INT(SQR(X)) értékét határozza meg, kivéve ha X négyzetszám, mert akkor A = SQR(X)-1. (C=2*A+1, B=(A+1)^2)

10.5
C = A / 10 értékét határozza meg.

10.6
Megadja A és B legnagyobb közös osztóját.

10.7
Felcseréli X és Y értékét.

10.8
Csökkenő sorrendbe rendez 3 számot.

10.9
A beírt szóközöket átlépi, vesszővel elválasztott szavakat ír ki. A beírás végét pont jelzi.

10.10
A programrészlet szintaktikusan hibás. Az `IF` utasítás vége a `THEN` alapszót követő első utasítás utáni pontosvessző. Ezért az `ELSE` alapszóra a Pascal fordító hibát jelez.

10.11
Két db 1-est ír ki és az ALFA eljárás már nem hívja meg magát.

10.12
A 10.9 feladat egyszerűbb, helyes megoldása. Szavakat ír ki egymás alá, amelyeket „ , ” választ el, s az utolsó szót „ . ” követi. A beírt szóközöket figyelmen kívül hagyja.

10.13
A HOSSZ-ban a SOR sorozat periódushosszát kapjuk meg.

10.14
A 10.13 feladat alapján. Ha nem sikerül, akkor kódold valamilyen nyelven és futtasd le!

10.15
Ha nem fejtetted meg, mégis futtasd le!

10.16
Próbáld a10.15 feladat alapján! Ha úgy sem sikerül, mégis futtasd le!

10.17
Például: A=3, B=2.

10.18
Tetszőleges X esetén Y legyen egyszer páros, egyszer páratlan! Pl. Y=3

10.19
Legyen egyszer N értéke hibás (<1 vagy >100), másodszor pedig N=2, SOROZAT(1)<SOROZAT(2)>=100. A második feltételnek eleget tevő tesztadatokhoz kell még: N=2 100>SOROZAT(1)>=SOROZAT(2).

10.20
A szükséges tesztadatok (az egyes kiíratásokhoz szükséges újabb feltételek):
Az első ciklus miatt: A(1) ≠0, A(2)=0,
A második ciklus miatt: A(N) ≠0, A(N-1)=0,
A harmadik ciklus miatt: A(2)=0, A(3) ≠0,
A negyedik ciklus miatt: A(3) ≠0, A(4)=0,
E értéke: N=5.

10.76
Leállási feltétel módosításával csökkentsd a ciklus lépésszámát!

10.77
A sok kivonás helyett alkalmazzunk osztást.

10.78
A sorozat részekre osztásával csökkenthető a lépésszám.

10.80
Használjuk ki a mátrix szimmetriáját!

10.81
Kihasználható, hogy a mátrix rendezett.

10.82
Pl. a ciklusok összevonásával

10.84
Alkalmazzunk rekurziót!

10.87
Nem kell minden csúcsot kigyűjteni, az összevonható ciklusokat vonjuk össze!

10.92
Hagyjuk el a felesleges műveleteket és feltételeket!

10.93
Nem kell minden pontot minden ponttal összehasonlítani.

10.94
A kivételes eset kiküszöbölésének módszerével.

10.99
Ciklusok szétválasztásának módszerével.

10.130
Ne számoljuk ki többször ugyanazt!

10.131
Tömb bevezetésével, ciklusösszevonással.

10.132
A ciklusból kiemelhető műveletet emeljük ki! Alkalmazzunk ciklusösszevonást!

10.133
A feltétel egyszerűsíthető, ha használjuk az ismert halmazelméleti azonosságokat.

10.134
Ha találtunk egy, a feltételnek megfelelő részsorozatot, akkor elég az első elemét elhagyni, a következőt hozzávenni, s vizsgálni, hogy ez jó lesz-e!

11. FEJEZET

JÁTÉKOK

11.2

- Könnyű észrevenni, hogy megkülönböztetett jelentősége van a középső négyzetnek.
- Nem teljesen kezdő játékosoknál leggyakoribb a döntetlen!

11.3

Sokat segíthet a szimmetria megfigyelése. Ezáltal egyrészt csökken a lényegesen különböző esetek száma, másrészt a válaszlépések is jobban megfigyelhetők.

11.8

Meglepő módon a másodiknak van védekező stratégiája: valamilyen rendszer szerint az ellenfél előző lépésének közelébe téve nem tud veszíteni!

11.11

Elég könnyű olyan részleges stratégiát választani, amely csak néhány dolgot vizsgál, és az eredménye már kielégítő. Az általános változat kifejezetten nehéz.

11.15

Meglepő módon elég jól működik egy kizárólag védekező stratégia! Ilymódon ez a feladat könnyebb, mint a legtöbb síkbeli megfelelője!

11.17

Négy (2*8-as) sávra osztva a táblát, mindegyikben el kell végezni az

1 2 3 4 5 6 7 8
5 6 7 8 1 2 3 4

számozást. Ennek segítségével már könnyű megfogalmazni, mit válaszoljon a második ellenfele minden lépésére. Tehát határozottan nyerő stratégia létezik a második számára!

11.24

Ne felejtsünk el védekezni se: az ellenfél sorában se engedjünk páratlan követ (mert ezeket azonnal nyerőre változtathatja)!

11.25

Gyakorlottak szinte csak döntetlent csinálnak: a végtelenségig húzogathatnak.

11.30
Először rajzoljuk át a táblát: egy négyzetrácson sokkal áttekinthetőbbek az utak, ha minden él egyenes szakasz. Ezután a pontok ügyes színezése (két színnel) segít a megoldásban.

11.33
Első ötletként vegyük észre, hogy mikor érdemes a tábla szélére (és főleg a sarkokba) rakni? Mikor érdemes a szélétől eggyel beljebb rakni?

11.37
Fontos védekezni is! Amit látszólag feleslegesen a másik útjába teszünk, az később még nagyon jól jöhet az építésben is!

11.41
Akkor kezd igazán izgalmassá válni, ha már nem tudják egymás négyzeteit megakadályozni. Ilyenkor nem érdektelen, hogy mit áldozunk, illetve milyen áldozatot kényszerítünk ki.

11.51
Kulcskérdés, hogyan tudjuk az egyes kövek kapcsolatait: „szomszédságukat", távolságukat, összefüggésük mértékét stb. kódolni?

11.59
A második játékosnak van nyerő stratégiája.

11.66
Ha a választott szám n, a második játékos akkor és csak akkor nyerhet, ha $n=13k-6$ vagy $n=13k$, ahol $k>1$ egész.

11.90
A kezdőnek van nyerő stratégiája.

11.91
Segítséget nyújt Varga Tamás: Osztójátékok című cikke. Lásd Élő matematika, 161-178. oldal!

11.92
Nyerő stratégiája annak van, aki páros számból tud kivonni.

11.93
A kezdő akkor és csakis akkor tudja elérni, hogy biztosan nyerjen, ha n páratlan.

11.99
Az 1. esetben a kezdő, a 2. esetben a második játékos nyerhet.

11.103
A második játékosnak van nyerő stratégiája.

11.106
Könnyítésül eláruljuk, hogy 1 és 100 közötti számhoz hét kérdés kell, míg 1 és 1000 között már tíz kérdés szükséges.

11.107
Egyetlen vitás pont van, amikor a kérdezett szám pontosan felezi a még szóba jöhető intervallumot. Ilyenkor mindkét válasz egyformán kellemetlen, de a későbbiek kedvéért valamelyiket választva egyértelművé kell tenni a dolgot.

11.108
A stratégia korrekt követése azt jelenti, hogy a pontos felezések következetesek: ilyenkor vagy mindig „kisebb" a válasz, vagy mindig az ellenkezője.

11.118
Legelőször – rögzítve egy pontot – figyeljük meg a tőle azonos távolságra lévő pontok helyét! Ezután vizsgáljuk, mely pontok jellemezhetők ugyanazon távolságpárokkal stb.

11.123
Itt megint segít, ha a különböző tippszámok utáni lehetőségeket (azok számát) elemezzük.

11.132
Térbeli játéknál az is szép teljesítmény, ha a játéktér (a képernyőn) valóban informál – nemcsak egy csupasz szám a válasz. Különösen didaktikus, ha több fokozatú válasz is lehetséges.

11.138
Most a felezős stratégia annyival bonyolultabb az eddigieknél, hogy nem adódik egy természetes (áttekinthető) rendezés a lehetséges esetekre. Szerencsére nem is túl nehéz készíteni ilyent..

11.149
Először próbáljuk ezt a közös összeget meghatározni, majd utána próbáljuk kitölteni a rácsot! A feladat szerepelt egy korábbi fejezetben is, most természetesen más módszert kell keresni. Pontosan ezért legyen N páratlan!!

11.153
Vegyük észre, hogy a megoldás során sokszor kell (az eredetihez és egymáshoz egyaránt) hasonló részfeladatokat végrehajtani! Jó lenne valami rekurzív megoldást produkálni!

11.154

Alapvető, hogy páros, illetve páratlan számú korong esetén más-más kezdés célszerű!

11.179

Érdemes jól végiggondolni, hiszen itt többfajta igazság mellett lehet „meggyőzően" érvelni. A valóság pedig első hallásra alig hihető!

12. FEJEZET

GRAFIKA ÉS HANG

12.20

Célszerű lehet még a rajzolás vezérlésére pl. a

```
Q W E   avagy   U I O        ↖ ↑ ↗
A   S           J   L        ←   →
Z X C           N M ,        ↙ ↓ ↘
```

billentyűket választani.

12.21

Ez a feladat kifejezetten gyakoroltatja a szubrutinok (eljárások) használatát, illetve elegáns menürendszer is készíthető!
Pl. a T első benyomásakor „megjegyzi" a gép a téglalap bal alsó csúcsát, második benyomásakor a jobb felső csúcsát; és ennek alapján rajzolja a téglalapot.

12.23

Ezzel a feladattal oldhatjuk meg karakterek generálását, piktogramok kezelését.

12.44

Minél laposabb téglalapból indulunk ki, annál könnyebb rájönni.

12.46

Ha a gépünk nem képes ezt könnyen (egy utasítással) végrehajtani, segít egy kis (említett) trigonometria.

12.72

Vízszintes mozgatásnál először számítsuk ki az újonnan belépő függvényértéket, mert ez szükségessé teheti a függőleges irányú mozgatást is! A mozgatás BASIC-ben igen lassú, ajánljuk a gépi kódú megoldást!

12.73

Ne feledkezzünk meg a tengelyek törlése után a velük együtt kitörölt grafikonpontokról!

12.82

Az osztásnál a g függvény zérushelyeinek környezetében problémák adódhatnak; ezek kiküszöbölésére írjunk hibakezelő szubrutint!

12.83

Az osztásnál a g függvény zérushelyeinek környezetében problémák adódhatnak; ezek kiküszöbölésére írjunk hibakezelő szubrutint!

12.86
Legyünk óvatosak az értelmezési tartományba nem eső értékek környezetében! Például az $x \rightarrow 1/x$ függvény -0.1 és +0.1 pontokban felvett értékét ne kössük össze!

12.90
Ha a lépésköz túl kicsi, akkor az ábra szebb, de sokáig tart a rajzolás; túl nagy lépésköz esetén gyorsan készül az ábra, de nagyon durva lesz.

12.115
Három ilyen kisebb háromszög van, tehát a rekurzív eljárás háromszor hívja meg önmagát. E három esetet megkaphatjuk egymásból 120 fokos forgatásokkal, ezért a hívást – megfelelő elmozdulásokkal – egy háromszor lefutó ciklusba tehetjük.

12.120
Dimenzió: log4/log3

12.130
A kocka hozzánk legközelebb levő éle látszódjék rövidülés nélkül, ebből kiindulva egyenesek metszéspontjainak kiszámításával oldható meg a feladat.

12.131
Használjuk, ha már írtunk ilyet, az ellipszisrajzoló programunkat! 8-8 hosszúsági és szélességi kör megrajzolása célszerű.

12.134
Próbáljuk meg úgy, hogy két ellipszis (kör) megfelelő pontjait kötjük össze fáziseltolódással!

12.135
Egyszerűbb a feladat megoldása, ha először a forgástengelyre illeszkedik a négyzet egyik oldala, így ugyanis annak az oldalnak nem változik a hossza. A horizont legyen a képernyő felső széle!

12.136
Egyszerűbb a feladat megoldása, ha először a forgástengelyre illeszkedik a kör megfelelő átmérője. A horizont legyen a képernyő felső széle!

12.141
Legyen ez a sík a képernyő síkjára merőleges függőleges sík!

12.143
Legyen ez a tengely a z tengely!

12.144

Legyen ez a pont az origó!

12.150

Kipróbálásra javasolt függvény:

z=0.5*(1-cos(5*PI/15*x*y/15))

12.160

A hangmagasság tárolására a nagy és kis abc megfelelő betűit javasoljuk: c, d, e, ...C, D, E, ... A hanghosszúság jelölésére javasolt számok: nyolcad – 8, negyed – 4, fél – 2, egész – 1. Megszólaltatáskor vesszük a számok reciprokát. (pl. a „Bociboci tarka" kezdetű dal eleje így fest: C-E-C-E-G-G-... és 8-8-8-8-4-4-...) A szünet jelölésére használhatjuk a 0, a %, a @ karakter valamelyikét!

12.161

A három oktáv megkülönböztetésére javasoljuk: c, d, e, ... c', d', e', ... c", d", e", ... A felemelt hangokat jelölhetjük így: cisz – #c, disz – #d, fisz – #f, ... A leszállított hangokat pedig így: gesz – bg, ász – ba, bé – bh, ...

12.163

Például a „Boci-boci tarka ..." kezdetű dalt G-dúrba transzponálva a következőképpen írja ki:
g-h-g-h-d-d-g-h-g-h-d-d-G-#F-E...
8-8-8-8-4-4-8-8-8-8-4-4-8- 8-8...

12.164

Például a „Boci-boci tarka ..." kezdetű dalt C-dónál a következőképpen írja ki:
d-m-d-m-s-s-d-m-d-m-s-s-d'-t-l-s-f-l-s-f-m-r-d-d.

12.171

Pl. c-e-c-e-g-g a kezdőhangra tükrözve: c-ba-c-ba-f-f .

12.172

A rákfordítás a dallam „visszafelé játszása", azaz minden hang ugyanolyan tartamú és magasságú, csak éppen a sorrendjük fordított.

12.179

Az ütemvonal jelzésére pl. a „:" karaktert javasoljuk. Még szebb módszer, ha a program a dallam beolvasása előtt megkérdezi az ütemjelzést (pl. 2/4), s így a program maga teszi ki az ütemvonalakat.

12.183

Egy dallam több modális hangsorban is felfogható.

12.187

Előbb rajzolja ki a gép a beolvasott ritmussort, majd a beolvasott ütemjelzés alapján végezze el a feladatot (ha elvégezhető) és rajzolja ki a megoldást!

12.190

Két dallamot akkor mondunk hasonlónak, ha egyik a másiknak transzponáltja.

12.191

Pl. C – Cisz-hez a C – Desz-t.

12.193

A pl. 16 oszlopban, grafikusan ábrázolt fogások alatt jelenjen meg a képernyőn a ritmus is!

13. FEJEZET

JÁTÉKOS-ÉRDEKES FELADATOK

13.1
BILLENTYŰZET

13.2
11. PROCESSZOR

13.3
editor

13.4
1. toll
2. információhordozó: CD, mágnesszalag, floppy
   adatbeviteli eszköz: scanner, egér, fényceruza
   logikai művelet: tagadás, és, vagy
3. jelentésük billentyűzet
4.A. monitor, nyomtató, billentyűzet, plotter ; az eltérés a harmadikban van.
   B. automata, memória, processzor, modem; az eltérés a harmadikban van.
5. memória.

13.5
A ROM csak olvasható memória.

13.6
A RAM írható és olvasható memória.

13.7
A bit angol szó darabot jelent.

A	B	I	T	A
N	G	O	L	S
Z	Ó	D	A	R
A	B	O	T	J
E	L	E	N	T

13.8

A latin bini az kettőt jelent.

A	L	A	T	I	N
B	I	N	I	A	Z
K	E	T	T	Ő	T
J	E	L	E	N	T

13.9

Az adat a számítógépekben bitek sorozata.

A	Z	A	D	A	T
A	S	Z	Á	M	Í
T	Ó	G	É	P	E
K	B	E	N	B	I
T	E	K	S	O	R
O	Z	A	T	A	.

13.10

Digitális számítógépek bináris számrendszerben dolgoznak.

D	*I*	*G*	*I*	*T*	*Á*	*L*	*I*	*S*	*S*	*Z*	*Á*	*M*
Í	*T*	*Ó*	*G*	*É*	*P*	*E*	*K*	*B*	*I*	*N*	*Á*	*R*
I	*S*	*S*	*Z*	*Á*	*M*	*R*	*E*	*N*	*D*	*S*	*Z*	*E*
R	*B*	*E*	*N*	*D*	*O*	*L*	*G*	*O*	*Z*	*N*	*A*	*K*

13.11

1. B: japán számolást segítő eszköz
Az egyik első eszközként a kb. 3000 éves abakusz tette lehetővé az egyszerűbb matematikai műveletvégzést. Hasonló eszközt használnak még ma is a japánok, amelyet szorobánnak neveznek, és ma újra elterjedőben levő eszköz.
2. A: A feladat megoldásához szükséges lépések felsorolása
Élt a IX. században egy arab tudós, Al-Hvarizmi, aki egyik könyvében recept-szerűen összefoglalta korának legfontosabb számítási összefüggéseit. Latinosított nevéből (Algoritmi) ered az algoritmus szó.
3. B: különbség (latin eredetű)
4. C: elemző
Görög-latin eredetű, részekre bontást alkalmazó tudományos kutató módszer.
5. A: diszkrét értékeken alapuló

A digitális jelek csak meghatározott számértékeket vehetnek fel. A számítógép is csak a digitális jeleket érti, hogy az analóg jeleket fel tudja dolgozni átalakítóra van szükség.
6. A: egyetemes, általános
7. C: emlékezet
A számítógép programokkal adatokat dolgoz fel. A jelenlegi gépek „számoló egysége" egyszerre csak egy adatot, ill. egy utasítást tud kezelni. A többi adatot, utasítást és az addigi eredményeket a memória őrzi addig, amíg a gép be van kapcsolva.
8. B: önműködő
Görög-latin eredetű. Bonyolult műveleteket, műveletsorozatokat megadott program alapján önműködően elvégző berendezés.
9. C: egy időben történő – görög eredetű szó
10. B: párhuzamosság, hasonlóság – görög-latin eredetű szó.
11. B: egységesül
Görög eredetű. Egységesül, több rész egésszé összegeződik.
12. A: nem egyidejű
Görög eredetű. Nem egyidejű, nem együttható, nem egy időben működő.
13. C: irányítás
Latin-francia eredetű.
14. A: összekötő, egyesítő
Latin eredetű; egybekötő, összekötő, egyesítő, társító. Matematikában az elemek sorrendjétől, a csoportosítás módjától független.
15. B: megfelelő
Az analóg jelek két határérték között bármilyen értékűek lehetnek.

13.12
INFORMÁCIÓ

13.13
bit

13.14
a/ jel; b/ bit; c/ kód; */ tilde

13.15
Pascal

13.17
A legrövidebb útvonal: 1→10→21→4→14→29→9→16→23→2→30→13→27→8→5→31→17→28→11→32

13.18
x 2 x 1 2 1 1 2 1 2 2 x 1 x

13.19
file

13.21
a BEJÁRAT-tól F-ig: : A\C\F
a BEJÁRAT-tól H-ig: A\B\D\H
a BEJÁRAT-tól K-ig: A\B\D\H\K
a BEJÁRAT-tól J-ig: A\C\F\G\J
E-ből D-be: \A\B\D
M-ből O-ba: \A\C\E\I\L\O
I-ből J-be: \C\F\G\J

13.22
vírus

13.23
A SZÁMÍTÓGÉP-PROGRAM NEM KÍVÁNSÁGAID, HANEM UTASÍTÁSAID SZERINT MŰKÖDIK!

13.24
b-d-a-e-c

13.25
1-H, 2-F, 3-I, 4-G, 5-J, 6-B, 7-C, 8-E, 9-A, 10-D

13.27
margó, sor, sablon, hasáb

13.28
HELYESÍRÁS ELLENŐRZÉS

13.29
1 - F, 2 - A, 3 - C, 4 - D, 5 - E, 6 - B

13.30
1 - b, 2 - c, 3 - a, 4 - d, 5 - b, 6 - c

13.31
Egy fontos elvárás a szövegrészekkel elvégezhető műveletekre vonatkozik. A szerkesztőnek tudnia kell egy kijelölt szövegrészt törölni, áthelyezni, átmásolni. Az utóbbi két műveletet legtöbb esetben az ún. zseb (scrap) használatával oldják meg. Az áthelyezésnél a zsebbe átkerül a szöveg, az eredeti helyéről törlődik, míg a másolásnál az eredeti is megmarad. Ezután az eltárolt szöveg tetszőleges számú alkalommal beilleszthető a zsebből a dokumentumba. A zseb tartalmát egy újabb zsebbe helyezett szöveg törli ki.

13.32

ADATFELDOLGOZÓ

Start ↓	7	9	8
5	6	11	4
1	2	10	3
13	14	12	Cél

13.33

A hiányzó helyek sorrendjében: adat, rekord, mező, adatállomány, adatbázis.

13.34

BARÁTOM A SZÁMÍTÓGÉP

FELELŐS KIADÓ KOCSIS ANDRÁS SÁNDOR
A KOSSUTH KIADÓ RT. ELNÖK-VEZÉRIGAZGATÓJA
MŰSZAKI VEZETŐ KUN GÁBOR
A BORÍTÓ PÁNYI BÉLA MUNKÁJA
TERJEDELME 24 (A/5) ÍV
NYOMTA ÉS KÖTÖTTE A SZEKSZÁRDI NYOMDA KFT.
FELELŐS VEZETŐ VADÁSZ JÓZSEF IGAZGATÓ

PROGRAMOZÁSI FELADATOK II.

Kétkötetes feladatgyűjteményünk mindazoknak szól, akik számítástechnikát tanulnak a Nemzeti Alaptanterv szerint. Elsősorban azok forgathatják sikerrel, akik az átlagosnál többet foglalkoznak programozással, ezért lehet alapkönyve az olyan számítástechnikai versenyeken résztvevőknek, mint a Nemes Tihamér Országos Középiskolai Számítástechnikai Tanulmányi verseny, illetve programozásból szeretnének felvételizni egyetemeken, főiskolákon. A feladatgyűjtemény fejezetei olyan témakörökről szólnak, amelyek az általános és a középiskolai számítástechnikai tananyaghoz kapcsolódnak, de van közöttük elmélyültebb tudást igénylő problémamegoldás is. A feladatok nem kötődnek egyetlen programozási nyelvhez, de tekintve, hogy a közoktatásban legelterjedtebb a Pascal, ezért a nyelvorientált feladatok nagyrésze is ehhez kapcsolódik, bár néhány esetben ajánlott a Basic vagy a C nyelvű program is.

ISBN 963-09-3977-0

9 789630 939775